Implicit Safe Set Algorithm for Provably Safe Reinforcement Learning

WEIYE ZHAO, Carnegie Mellon University, United States
FEIHAN LI, Carnegie Mellon University, United States
TAIRAN HE, Carnegie Mellon University, United States
CHANGLIU LIU*, Carnegie Mellon University, United States

Deep reinforcement learning (DRL) has demonstrated remarkable performance in many continuous control tasks. However, a significant obstacle to the real-world application of DRL is the lack of safety guarantees. Although DRL agents can satisfy system safety in expectation through reward shaping, designing agents to consistently meet hard constraints (e.g., safety specifications) at every time step remains a formidable challenge. In contrast, existing work in the field of safe control provides guarantees on persistent satisfaction of hard safety constraints. However, these methods require explicit analytical system dynamics models to synthesize safe control, which are typically inaccessible in DRL settings. In this paper, we present a model-free safe control algorithm, the implicit safe set algorithm, for synthesizing safeguards for DRL agents that ensure provable safety throughout training. The proposed algorithm synthesizes a safety index (barrier certificate) and a subsequent safe control law solely by querying a black-box dynamic function (e.g., a digital twin simulator). Moreover, we theoretically prove that the implicit safe set algorithm guarantees *finite time convergence* to the safe set and *forward invariance* for both continuous-time and discrete-time systems. We validate the proposed algorithm on the state-of-the-art Safety Gym benchmark, where it achieves zero safety violations while gaining 95% ± 9% cumulative reward compared to state-of-the-art safe DRL methods. Furthermore, the resulting algorithm scales well to high-dimensional systems with parallel computing.

JAIR Associate Editor: Marek Petrik

JAIR Reference Format:
Weiye Zhao, Feihan Li, Tairan He, and Changliu Liu. 2025. Implicit Safe Set Algorithm for Provably Safe Reinforcement Learning. *Journal of Artificial Intelligence Research* 84, Article 25 (December 2025), 44 pages. DOI: 10.1613/jair.1.17286

1 Introduction

Deep reinforcement learning (DRL) has achieved impressive results in continuous control tasks (Fujimoto et al. 2025; Govinda et al. 2025; Schulman et al. 2017; Zhao, F. Li, Sun, Chen, et al. 2024), but its real-world deployment is hindered by the absence of hard safety guarantees. Ensuring safety in terms of persistently satisfying hard state constraints has long been recognized as a critical requirement in robotics (Zhao, He, Chen, et al. 2023; Zhao, F. Li, Sun, Wang, et al. 2024). For example, vehicles must avoid colliding with pedestrians; robotic arms should not strike walls; and aerial drones must steer clear of buildings. When robots learn skills via reinforcement learning, they need to sufficiently explore the environment to discover optimal policies. However, such exploration is not inherently safe. A common strategy to encourage safe behavior is to introduce penalties for visiting unsafe

*Corresponding Author.

Authors' Contact Information: Weiye Zhao, ORCID: 0000-0002-8426-5238, weiyezha@andrew.cmu.edu, Carnegie Mellon University, Pittsburgh, Pennsylvania, United States; Feihan Li, ORCID: 0000-0003-1770-4664, feihanl@andrew.cmu.edu, Carnegie Mellon University, Pittsburgh, Pennsylvania, United States; Tairan He, ORCID: 0000-0001-6415-3223, tairanh@andrew.cmu.edu, Carnegie Mellon University, Pittsburgh, Pennsylvania, United States; Changliu Liu, ORCID: 0000-0002-3767-5517, cliu6@andrew.cmu.edu, Carnegie Mellon University, Pittsburgh, Pennsylvania, United States.

DOI: 10.1613/jair.1.17286

states. While these *posterior* measures are widely adopted (Papini et al. 2022; Pirotta et al. 2013; Zhao, He, Chen, et al. 2023), they are often insufficient to ensure robust safety. Instead, *prior* measures are required to proactively prevent robots from entering unsafe states in the first place.

Constraining robot motion to persistently satisfy hard safety specifications in uncertain environments is an active area of research in the safe control community. Among the most widely used approaches are *energy function-based methods*. These methods (Ames et al. 2014; Gracia et al. 2013; Khatib 1986; C. Liu and Tomizuka 2014) define an energy function that assigns low energy to safe states and construct a control law that dissipates this energy. When the energy function and the control law are properly designed, the system remains within the safe set (i.e., exhibits *forward invariance*) and may also converge to safe states in finite time if it starts from an unsafe condition (i.e., demonstrates *finite-time convergence*). Initially developed for deterministic systems, these methods have been extended to handle stochastic and uncertain systems (Cheng et al. 2019; Cosner et al. 2023; Pandya et al. 2024; Taylor and Ames 2020), where uncertainties may arise from state measurements, the future evolution of the ego robot, or the behavior of obstacles. Furthermore, they have been integrated with robot learning frameworks to enhance safety assurance (Du et al. 2023; Fisac et al. 2018; H. Zhang et al. 2025). However, a key limitation of energy function-based methods is their reliance on *white-box analytical models* of the system dynamics (e.g., the Kinematic Bicycle Model (Kong et al. 2015)) for both offline construction of the energy function and online computation of the safe control signal.

This paper extends the class of *energy function-based methods* by proposing a model-free safe control approach that provides safety guarantees without requiring an analytical dynamics model. The proposed method is capable of safeguarding any robot learning algorithm by integrating safety constraints directly into the control synthesis. The key insight underlying our approach is that a safe control law can be synthesized without access to a white-box analytical dynamics model, provided that a black-box dynamics function is available—that is, a function that maps the current state and control input to the next state. Importantly, such black-box models are often accessible in real-world applications, for example, through high-fidelity digital twin simulators (Abou-Chakra et al. 2025; Das et al. 2022; M. Liu et al. 2021) or deep neural network dynamics models trained in a data-driven manner (F. Li et al. 2025; W. Zhang et al. 2021). While this paper does not explicitly analyze formal guarantees on the accuracy of the black-box dynamics, it focuses on developing a reliable approach to synthesize the safe control strategy using only the black-box dynamics models, which is essential when designing safety-critical learning-based control systems in practice.

The key questions that we want to answer are: 1) how to synthesize the energy function (in our case called the safety index) with black-box dynamics, so that there always exists a feasible control to dissipate the system energy at all potentially unsafe states; 2) how to efficiently synthesize the optimal safe control with black-box dynamics, such that the control input can both dissipate the system energy and achieve good task efficiency; and 3) how to formally prove that the synthesized energy function and the generated safe control can guarantee *forward invariance* and *finite time convergence* to the safe set. We propose a safety index design rule to address the first question, and a sample-efficient algorithm to perform black-box constrained optimization to address the second question. As for the third question, we show that under certain assumptions, the proposed safety index design rule together with the proposed black-box optimization algorithm can guarantee *forward invariance* and *finite time convergence* to a subset within the safe set. By combining these approaches with the (model-based) safe set algorithm (C. Liu and Tomizuka 2014), we propose the (model-free) implicit safe set algorithm (ISSA). We then use ISSA to safeguard deep reinforcement learning (DRL) agents.

A short version of this paper has been presented earlier (Zhao, He, and C. Liu 2021), where ISSA is proposed and evaluated with the assumption that the sampling time is almost zero (i.e., the system evolves almost like a continuous-time system). In reality, most control and simulation systems are implemented in a discrete-time manner with non-negligible sampling time. As a result, the controller may not be able to respond immediately to potential violations of safety. Therefore, assuring provable safety for discrete-time systems with non-negligible

sampling time is important. In this paper, we introduce a general version of the implicit safe set algorithm which explicitly takes the sampling time into consideration. In particular, a novel convergence trigger (CTrigger) algorithm is introduced. The key contributions of this paper are summarized below:

- We propose two techniques to enable safe control with black-box models: (offline) continuous-time and discrete-time safety index design rules for mobile robots in a 2D plane, and a (online) sample-efficient black-box optimization algorithm using adaptive momentum boundary approximation (AdamBA).
- We propose the implicit safe set algorithm (ISSA) using these two techniques, which guarantees to generate safe controls for all system states without knowing the explicit system dynamics. We show that ISSA can safeguard DRL agents to ensure zero safety violation during training in Safety Gym. Our code is available on Github.[1]
- We provide the theoretical guarantees that 1) ISSA ensures the *forward-invariance* safety for both continuous-time and discrete-time systems; and 2) *finite time convergence* safety when ISSA is combined with CTrigger for both continuous-time and discrete-time systems (i.e., with any sampling time step).

In the remainder of the paper, we first formulate the mathematical problem in Section 2. We then discuss related work about safe reinforcement and safe control in Section 3. In Section 4 and Section 5, we first introduce the implicit safe set algorithm, and provide theoretical results in continuous-time systems. We further extend the implicit safe set algorithm to the discrete-time systems and provide theoretical guarantees of *forward invariance* and *finite time convergence* in Section 6 and Section 7, respectively. Finally, we validate our proposed method in Safety Gym environments in Section 8.

2 Problem Formulation

Dynamics. Let $x_t \in \mathcal{X} \subset \mathbb{R}^{n_x}$ be the robot state at time step t, where n_x is the dimension of the state space $\mathcal{X}$; $u_t \in \mathcal{U} \subset \mathbb{R}^{n_u}$ be the control input to the robot at time step t, where n_u is the dimension of the control space $\mathcal{U}$. Denote the sampling time as dt. The system dynamics are defined as:

$$x_{t+1} = f(x_t, u_t), \tag{1}$$

where $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ is a function that maps the current robot state and control to the next robot state. For simplicity, this paper considers deterministic dynamics. The proposed method can be easily extended to the stochastic case through robust safe control (Emam et al. 2025), which will be left for future work. Moreover, it is assumed that we can only access an implicit black-box form of f, e.g., as an implicit digital twin simulator or a deep neural network model. Note that the word *implicit* refers to that we can only do point-wise evaluation of $f(x, u)$ without any explicit knowledge or analytical form of $f(x, u)$.

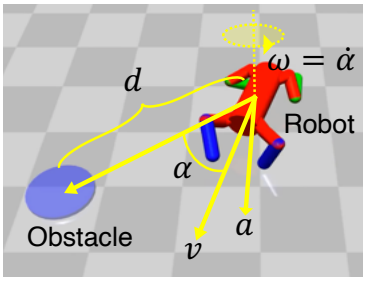


Fig. 1. Visualization of robot notations.

[1] https://github.com/intelligent-control-lab/Implicit_Safe_Set_Algorithm

Safety Specification. The safety specification requires that the system state should be constrained in a regular and closed subset in the state space, called the safe set $\mathcal{X}_S$. The safe set can be represented by the zero-sublevel set of a continuous and piecewise smooth function $\phi_0 : \mathbb{R}^{n_x} \to \mathbb{R}$, i.e., $\mathcal{X}_S = \{x \mid \phi_0(x) \leq 0\}$. $\mathcal{X}_S$ and ϕ_0 are directly specified by users. A state $x(t_0)$ is considered *safe* if it belongs to the safe set $\mathcal{X}_S$ at time t_0, notwithstanding the possibility that the trajectory $x(t)$ may exit $\mathcal{X}_S$ at some later time $t_1 > t_0$. The notion for the state, conditioned on the control policy, to result in a trajectory that always lies in the safe set is called *invariantly safe*, which is later captured by the notion of *forward invariance*. Hence the safe set $\mathcal{X}_S$ captures static safety and *forward invariance*, which usually happens within a subset of $\mathcal{X}_S$, captures dynamic safety. Nevertheless, the design of ϕ_0 is straightforward in most scenarios. For example, for collision avoidance, ϕ_0 can be designed as the negative closest distance between the robot and environmental obstacles.

Reward and Nominal Control. A robot learning controller generates the nominal control which is subject to modification by the safeguard. The learning controller aims to maximize rewards in an infinite-horizon deterministic Markov decision process (MDP). An MDP is specified by a tuple $(\mathcal{X}, \mathcal{U}, \gamma, r, f)$, where $r : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is the reward function, $0 \leq \gamma < 1$ is the discount factor, and f is the deterministic system dynamics defined in (1).

Safeguard Synthesis. Building on top of the nominal learning agent, the core problem of this paper is to synthesize a safeguard for the learning agent, which monitors and modifies the nominal control to ensure **forward invariance** in a subset of the safe set $\mathcal{X}_S$ and **finite time convergence** to that subset. *Forward invariance* of a set means that the robot state will never leave the set if it starts from the set. *Finite time convergence* to a set means that the robot state will return to the set in a finite amount of time if it is not initially in the set. The reason why we need to find a subset instead of directly enforcing *forward invariance* of $\mathcal{X}_S$ is that $\mathcal{X}_S$ may contain states that will inevitably go to the unsafe set no matter what the control input is. These states need to be penalized when we synthesize the energy function. For example, when a vehicle is moving toward an obstacle at high speed, it would be too late to stop. Even if the vehicle is safe now (if $\mathcal{X}_S$ only constrains the position), it will eventually collide with the obstacle (unsafe). Then we need to assign high energy values to these inevitably-unsafe states.

Collision Avoidance for Mobile Robots. In this paper, we are focused on collision avoidance for mobile robots in a 2D plane. The robot and the obstacles are all geometrically simplified as point-mass circles with bounded collision radius. The safety specification is defined as $\phi_0 = \max_i \phi_{0_i}$, where $\phi_{0_i} = d_{min} - d_i$, and d_i denotes the distance between the center point of the robot and the center point of the i-th obstacle (static or non-static), and the radius of both the obstacle and the robot are considered in d_{min} such that $d_{min} \geq$ obstacle radius + robot radius. Denote v, a and w as the relative velocity, relative acceleration and relative angular velocity of the robot with respect to the obstacle, respectively, as shown in fig. 1.[2] The remaining discussions are based on the following assumption.

Assumption 1 (Actuation Reachability). *1) The state space is bounded ($v \in [0, v_{max}]$), and the relative acceleration and angular velocity are bounded and both can achieve zeros, i.e., $w \in [w_{min}, w_{max}]$ for $w_{min} \leq 0 \leq w_{max}$ and $a \in [a_{min}, a_{max}]$ for $a_{min} \leq 0 \leq a_{max}$; 2) For all possible values of a and w, there always exists an input u such that the closed-loop low-level servo tracks that pair within the sampling period dt.*

These assumptions are easy to meet in practice. The bounds in the first assumption will be directly used to synthesize the safety index ϕ (will be introduced in Section 3). The second assumption enables us to turn the question on whether there exists a feasible safe control in $\mathcal{U}$ to the question on whether there exists a and w to

[2]Note that in our definition, $w = \dot{\alpha}$, where α is the angle between the robot's heading vector and the vector from the robot to the obstacle. The definition of w is different from the robot angular velocity in the world frame, where w is influenced by the 1) the robot's motion around the obstacle, which changes the vector from the robot to the obstacle and 2) the robot's self-rotation in world frame, which changes the robot's heading. Since the obstacle is treated as a circle, we do not consider its self-rotation in the world frame. In real world systems, a robot's self-rotation is bounded, and its motion around the obstacle is also bounded. Therefore, w is bounded.

influence ϕ. The requirement is satisfied by most wheeled or tracked mobile bases whose wheel/track torques can be commanded independently. It does not imply that the robot should be able to perform holonomic motion in Cartesian space; translational motion remains subject to the usual non-holonomic constraint. Our experiments use MuJoCo's unicycle and quadruped models whose low-level velocity controllers meet this reachability property.

3 Related Work

Safe Reinforcement Learning. Safe RL methods enforce safety through either soft safety constraints or hard safety constraints. For example, as for the safety specification of mobile robots, soft safety contraints limit a number of crashes per trajectory, whereas hard safety constraints require mobile robots should never crash into pedestrians. Typical safe RL methods for soft safety constraints include risk-sensitive safe RL (Garcıa and Fernández 2015; Noorani et al. 2025), Lagrangian methods (Ray et al. 2019), trust-region-based constrained policy optimization (Achiam et al. 2017; Zhao, Chen, Sun, F. Li, et al. 2024; Zhao, F. Li, Sun, Wang, et al. 2024). These methods are able to find policies that satisfy the safety constraint in expectation, but cannot ensure all visited states are safe. The methods that are more closely related to ours are safe RL methods with hard safety constraints. Safe RL methods with hard constraints can be divided into two categories: 1) safeguarding learning policies using control theories; and 2) safeguarding learning policies through a safety layer to encourage safe actions.

For the first category, safeguard methods are proposed based on 1) Control Barrier Function (CBF), 2) Hamilton-Jacobi reachability, and 3) Lyapunov method. Richard *et al.* (Cheng et al. 2019) propose a general safe RL framework, which combines a CBF-based safeguard with RL algorithms to guide the learning process by constraining the exploratory actions to the ones that will lead to safe future states. However, this method strongly relies on the control-affine structure of the dynamics system, which restricts its applicability to general non-control-affine unknown dynamics systems. Ferlez *et al.* (Ferlez et al. 2020) also propose a ShieldNN leveraging Control Barrier Function to design a safety filter neural network with safety guarantees. However, ShieldNN is specially designed for an environment with the kinematic bicycle model (KBM) (Kong et al. 2015) as the dynamical model, which cannot be directly applied to general problems. Besides CBF, reachability analysis is also adopted for the safeguard synthesis. Fisac *et al.* (Fisac et al. 2018) propose a general safety framework based on Hamilton-Jacobi reachability methods that works in conjunction with an arbitrary learning algorithm. However, these methods (Fisac et al. 2018; Pham et al. 2018) still rely heavily on the explicit analytic form of system dynamics to guarantee constraint satisfaction. In addition to CBF and reachability, Lyapunov methods are used to verify stability of a known system in control, which can be used to determine whether states and actions are safe or unsafe. Berkenkamp *et al.* (Berkenkamp et al. 2017) combine RL with Lyapunov analysis to ensure safe operation in discretized systems. Though provable safe control can be guaranteed under some Lipschitz continuity conditions, this method still requires explicit knowledge of the system dynamics (analytical form). In summary, control theory based safeguard methods provide good safety guarantees. However, those methods rely on the assumptions of 1) explicit knowledge of dynamics model or 2) control-affine dynamics model, which are hard to be satisfied in real world robotics applications.

The second category to make RL meet hard safety constraints is shield synthesis–style approaches, where a safety layer monitors the action proposed by the RL policy and corrects it, if necessary, to satisfy safety constraints. Such methods can guarantee safety online by projecting unsafe actions into a certified safe set before execution. Dalal *et al.* (Dalal et al. 2018) propose to learn the system dynamics directly from offline collected data, and then add a safety layer that analytically solves a control correction formulation at each state to ensure every state is safe. However, the closed-form solution relies on the assumption of the linearized dynamics model and cost function, which is not true for most dynamics systems. They also assume the set of safe control can be represented by a linearized half-space for all states, which does not hold for most of the discrete-time system (i.e., safe control may not exist for some states). Yinlam *et al.* (Chow et al. 2019) propose to project either the policy

parameters or the action to the set induced by linearized Lyapunov constraints, which still suffer from the same linear approximation error and non-control-affine systems as in (Dalal et al. 2018) and is not able to guarantee zero-violation. Bejarano *et al.* (Bejarano et al. 2025) integrate safety filters into the RL training process, allowing the controller to adapt to the presence of the filter and thereby improving the overall safety–performance trade-off and sample efficiency. However, their approach still fundamentally relies on an analytical dynamics model, which limits applicability in scenarios where such a model is unavailable or inaccurate. These works all fall under the general paradigm of shield synthesis in RL, where a pre-specified or learned model is used online to intercept and correct unsafe actions before execution. In contrast, our proposed method does not require the explicit form or control-affine form of system dynamics, and our method can be guaranteed to generate safe control as long as the safe set is regular (i.e., the set equal to the closure of its interior; for example, it is not a collection of singleton points) and contains forward invariant subsets.

Safe Control. Representative *energy function-based methods* for safe control include potential field methods (Khatib 1986), control barrier functions (CBF) (Ames et al. 2014), safe set algorithms (SSA) (C. Liu and Tomizuka 2014), sliding mode algorithms (Gracia et al. 2013), and a wide variety of bio-inspired algorithms (J. Zhang et al. 2017). The next step towards safe controller synthesis is to design a desired energy function offline, ensuring that 1) the low energy states are safe and 2) there always exists a feasible control input to dissipate the energy. SSA has introduced a rule-based approach (Z. Li et al. 2023) to synthesize the energy function as a continuous, piece-wise smooth scalar function on the system state space $\phi : \mathbb{R}^{n_x} \to \mathbb{R}$. And the energy function $\phi(x)$ is called a safety index in this approach. The general form of the safety index was proposed as $\phi = \phi_0^* + k_1\dot{\phi}_0 + \cdots + k_n\phi_0^{(n)}$ where 1) the roots of $1 + k_1 s + \ldots + k_n s^n = 0$ are all negative real (to ensure zero-overshooting of the original safety constraints); 2) the relative degree from $\phi_0^{(n)}$ to u is one (to avoid singularity); and 3) ϕ_0^* defines the same set as ϕ_0 (to nonlinear shape the gradient of ϕ at the boundary of the safe set). It is shown in (C. Liu and Tomizuka 2014) that if the control input is unbounded ($\mathcal{U} = \mathbb{R}^{n_u}$), then there always exist a control input that satisfies the constraint $\dot{\phi} \leq 0$ when $\phi = 0$; and if the control input always satisfies that constraint, then the set $\{x \mid \phi(x) \leq 0\} \cap \{x \mid \phi_0(x) \leq 0\}$ is forward invariant. In practice, the actual control signal is computed through a quadratic projection of the nominal control u^r to the control constraint:

$$u = \arg\min_{u \in \mathcal{U}} \|u - u^r\|^2 \text{ s.t. } \dot{\phi} \leq -\eta(\phi), \tag{2}$$

where $\dot{\phi} \leq -\eta(\phi)$ is a general form of the constraint; $\eta : \mathbb{R} \to \mathbb{R}$ is a non-decreasing function that $\eta(0) \geq 0$. For example, in CBF, $\eta(\phi)$ is designed to be $\lambda\phi$ for some positive scalar λ. In SSA, $\eta(\phi)$ is designed to be a positive constant when $\phi \geq 0$ and $-\infty$ when $\phi < 0$. Note there are two major differences between this paper and the existing results presented in (C. Liu and Tomizuka 2014). First, this paper considers constrained control space, which then requires careful selection of the parameters in ϕ to make sure the control constraint $\mathcal{U}_S(x) := \{u \in \mathcal{U} \mid \dot{\phi} \leq -\eta(\phi)\}$ is nonempty for states that $\phi \geq 0$. Secondly and most importantly, this paper considers general black-box dynamics, while the existing work considers analytical control-affine dynamics. For analytical control-affine dynamic models, $\mathcal{U}_S(x)$ is essentially a half-space, and the projection (2) can be efficiently computed by calling a quadratic programming solver. However, for black-box dynamics, this constraint is challenging to quantify.

Since real-world robot systems are always controlled in discrete-time with dynamics $x_{t+1} = f(x_t, u_t)$, we consider the discrete-time version of the set of safe control $\mathcal{U}_S^D(x) := \{u \in \mathcal{U} \mid \phi(f(x,u)) \leq \max\{\phi(x) - \eta, 0\}\}$ in the remaining discussions. By defining $\mathcal{L}(\phi) := \{x | \phi(x) \leq 0\}$, our ultimate goal is to ensure forward invariance and finite-time convergence with respect to the set $\mathcal{S} := \mathcal{X}_S \cap \mathcal{L}(\phi)$. Notice that the continuous-time system is a special case of a discrete-time system where the simulation time step (sampling time) is negligible, i.e. $dt \to 0$.

Additionally, $\mathcal{U}_S(x)$ and $\mathcal{U}_S^D(x)$ pose similar requirements for safe control, which require ϕ to decrease at the next time step when $\phi \geq 0$ at the current time step.

4 Implicit Safe Set Algorithm for Systems with Negligible Sampling Time

This section introduces the implicit safe set algorithm (ISSA), which is able to leverage *energy function-based methods* (SSA in particular) with black-box dynamics, and be used to safeguard any nominal policy. ISSA contains two parts: 1) a safety index synthesis rule to make sure $\mathcal{U}_S^D(x)$ is nonempty for all x, and 2) a sample-efficient black-box optimization method to solve the projection of the nominal control to $\mathcal{U}_S^D(x)$. With these two components, the overall pipeline of the implicit safe set algorithm is summarized as follows:

- **Offline:** Design the safety index $\phi(x)$ according to the safety index design rule.
- **Online:** Project nominal control into $\mathcal{U}_S^D(x)$ during online robot maneuvers.

4.1 Synthesize Safety Index for Continuous-time System

The safety index for collision avoidance in 2D plane will be synthesized without referring to the specific dynamic model, but under the following assumptions.

Assumption 2 (2D Collision Avoidance for continuous-time system). *1) The system time step $dt \to 0$. 2) At any given time, there can at most be one obstacle becoming safety-critical, such that $\phi_i \geq 0$ (Sparse Obstacle Environment).*

These assumptions are easy to meet in practice. The first assumption ensures that the discrete-time approximation error is negligible, i.e., the system can essentially be treated as a continuous-time system. The second assumption makes the safety index design rule applicable with multiple moving obstacles.

Following the rules in (C. Liu and Tomizuka 2014), we parameterize the safety index as $\phi = \max_i \phi_i$,

$$\phi_i = \sigma + d_{min}^n - d_i^n - k\dot{d}_i, \tag{3}$$

where all ϕ_i share the same set of tunable parameters $\sigma, n, k, \eta \in \mathbb{R}^+$. We illustrate the behavior of the safety index ϕ with respect to d and $\dot{d}$ under varying k and n in Figure 2. Our goal is to choose these parameters such that $\mathcal{U}_S^D(x)$ is always nonempty. Under the above assumptions, the safety index design rule is constructed below. The main idea of designing the safety index is to derive the sufficient condition of parameters of ϕ to ensure there exist safe controls for all states.

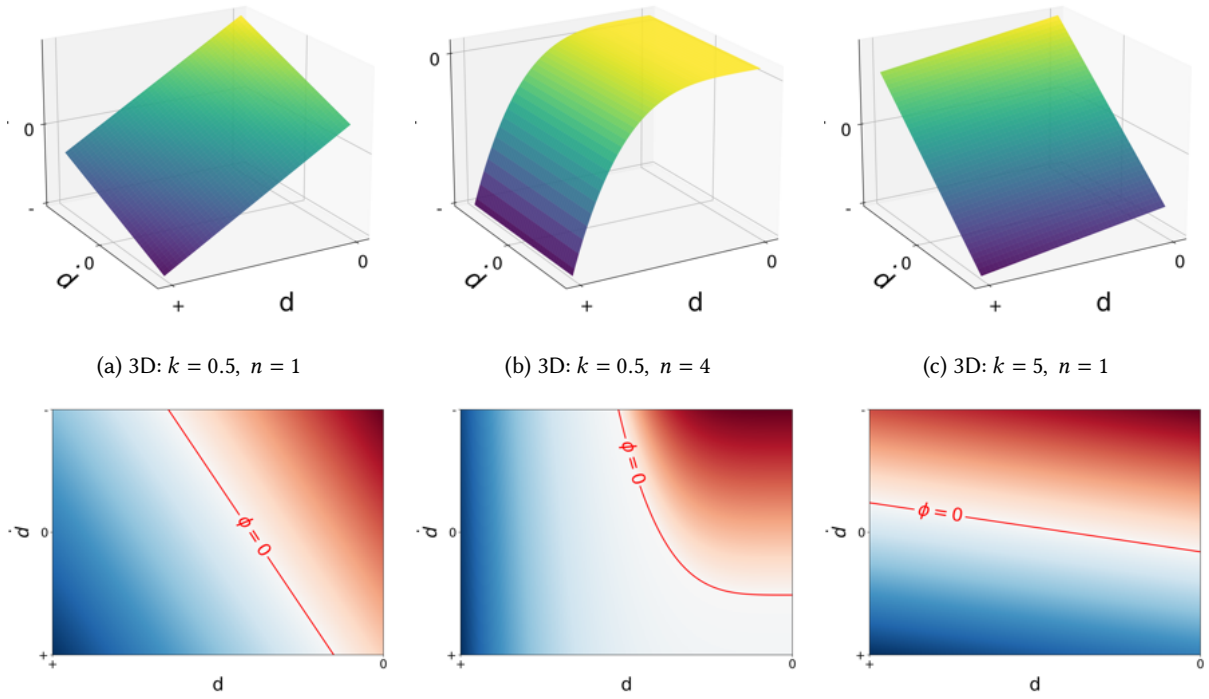
Safety Index Design Rule for Continuous-Time System: By setting $\eta = 0$, the parameters k, n, and σ should be chosen such that

$$\frac{n(\sigma + d_{min}^n + kv_{max})^{\frac{n-1}{n}}}{k} \leq \frac{-a_{min}}{v_{max}}, \tag{4}$$

where v_{max} is the maximum relative velocity that the robot can achieve in the obstacle frame. This design rule, together with the conditions in Assumption 1 and Assumption 2, guarantees that the safe control set $\mathcal{U}_S^D(x)$ is non-empty at any arbitrary state, which is the key to establishing Theorem 1. The proof of this non-emptiness is provided in Proposition 1.

4.2 Understanding of Safety Index Design Rule

Design philosophy. We treat n as a user-chosen shaping parameter for the safety index, while k is then selected to satisfy (4) so that a safe control set $\mathcal{U}_S^D(x)$ is non-empty for all states. We begin with the role of k.



(a) 3D: $k = 0.5,\ n = 1$ (b) 3D: $k = 0.5,\ n = 4$ (c) 3D: $k = 5,\ n = 1$

(d) 2D: $k = 0.5,\ n = 1$ (e) 2D: $k = 0.5,\ n = 4$ (f) 2D: $k = 5,\ n = 1$

Fig. 2. Combined visualization of the safety index ϕ with respect to d and $\dot{d}$ under varying k and n: 3D surface plots (top row; z-axis is ϕ, brighter is higher) and 2D heat maps (bottom row; darker red indicates more positive ϕ—unsafe, darker blue more negative ϕ—safe; red line shows the zero level set $\phi = 0$). In the axes, "+" denotes positive and "–" denotes negative values. The design $(k = 0.5, n = 1)$ is also discussed in Section 8.

Effect of k. With safety index parameterization form in (3), the coefficient k scales the index's sensitivity to the relative speed $\dot{d}$. Increasing k steepens the slope of ϕ along the $\dot{d}$ axis, so ϕ rises more aggressively when the robot is closing in (i.e., $\dot{d} < 0$). Visually, compare Fig. 2(a,d) (k=0.5, n=1) and Fig. 2(c,f) (k=5, n=1): the surface tilts more strongly with $\dot{d}$, indicating earlier and stronger reactions to robot-obstacle approach speed ($\dot{d}$). Equation (4) explains why k cannot be arbitrarily small: to keep the safe control set nonempty, k must grow when the maximum relative velocity $v_{\max}$ is larger or when the available maximum deceleration $|a_{\min}|$ is smaller. Intuitively, faster environments or weaker braking demand earlier reactions, i.e., the larger the k is, the more sensitive the safety mechanism is to $\dot{d}$, working like a time-to-collision style constraint.

Effect of n (distance–velocity trade-off). The distance term $d_{\min}^n - d^n$ in (3) sharpens as n increases, redistributing attention toward *short distances*. Comparing Fig. 2(a,d) (k=0.5, n=1) and Fig. 2(b,e) (k=0.5, n=4), the surface rises much more quickly as d shrinks. Thus, for the same $(d, \dot{d})$, a larger n yields a larger ϕ when the robot is close, i.e., less tolerance to running toward an obstacle at short range, yet allows more tolerance to aggressive motion when the robot is far away. In other words, increasing n changes the *preference* of the safety index: strict when close, permissive when far.

Why n and k should change together. Once n is chosen to set that preference, k must be tuned to preserve feasibility under (4). A larger n makes ϕ grow rapidly at small d; with bounded deceleration $|a_{\min}|$, there is only so much (and so fast) the controller can slow down. To ensure nonempty set of safe control that prevents ϕ from diverging upward, the design must respond more sensitively when $\dot{d}$ decreases, i.e., increase k. This coupling is explicit in (4), where n and k appear together (through a factor of the form $\frac{n}{k}\left(\sigma + d_{\min}^{n} + k\, v_{\max}\right)^{(n-1)/n}$): raising n typically necessitates raising k to keep the inequality satisfied. Practically: choose n to sculpt proximity sensitivity; then increase k as needed to guarantee a nonempty safe control set by considering speed/braking limits of the environment.

4.3 Sample-Efficient Black-Box Constrained Optimization

The nominal control u_t^r needs to be projected to $\mathcal{U}_S^D(x)$ by solving the following optimization:

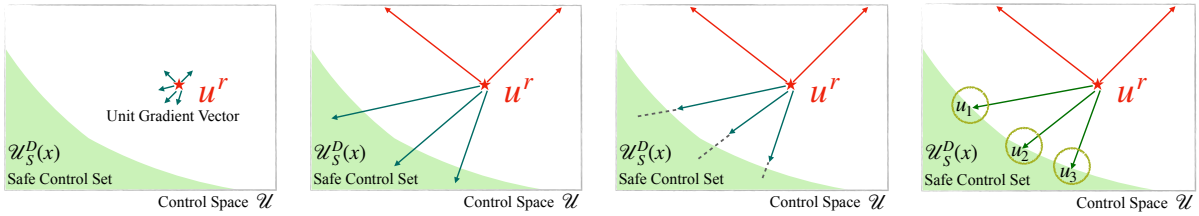
$$\begin{aligned} &\min_{u_t \in \mathcal{U}} \|u_t - u_t^r\|^2 \\ &\text{s.t. } \phi(f(x_t, u_t)) \leq \max\{\phi(x_t) - \eta, 0\} \end{aligned} \tag{5}$$

A key insight we have is that: since the objective of (5) is convex, its optimal solution will always lie on the boundary of $\mathcal{U}_S^D(x)$ if $u^r \notin \mathcal{U}_S^D(x)$. Therefore, it is desired to have an efficient algorithm to find the safe controls on the boundary of $\mathcal{U}_S^D(x)$. To efficiently perform this black-box optimization, we propose a sample-efficient boundary approximation algorithm called Adaptive Momentum Boundary Approximation Algorithm (AdamBA), which is summarized in Algorithm 1. AdamBA leverages the idea of adaptive line search with exponential outreach/decay to efficiently search for the boundary points. The inputs for AdamBA are the approximation error bound (ϵ), learning rate (β), reference control (u^r), gradient vector covariance (Σ), gradient vector number (n), reference gradient vector ($\vec{v}^r$), safety status of reference control (S), and the desired safety status of control solution (S_{goal}).

We illustrate the main boundary approximation procedures of AdamBA in Figure 3, where AdamBA is supposed to find the boundary points of $\mathcal{U}_S^D(x)$ (green area) with respect to the reference control $u^r \notin \mathcal{U}_S^D(x)$ (red star). The core idea of the AdamBA algorithm follows the adaptive line search (Armijo 1966), where three main procedures are included. I. AdamBA first initializes several unit gradient vectors (green vectors) to be the sampling directions, as shown in Figure 3a. II. AdamBA enters the *exponential outreach* phase by exponentially increasing the gradient vector length until they reach $\mathcal{U}_S^D(x)$ as shown in Figure 3b. Note that we discard those gradient vectors that go out of control space (red vectors). III. Next, AdamBA enters the *exponential decay* phase by iteratively applying Bisection search to find boundary points as shown in Figure 3c. Finally, a set of boundary points will be returned after AdamBA converges as shown in Figure 3d. Note that AdamBA and the line search methods are fundamentally similar to each other, while the purpose of AdamBA is to find the boundary of safe/unsafe action, while the line search methods are to find the minimum of a function. It can be shown that the boundary approximation error can be upper bounded within an arbitrary resolution.

Relation to momentum methods. The proposed AdamBA algorithm employs a momentum-inspired strategy in step-size control. During the exponential outreach phase, the step length β is geometrically amplified ($\beta \leftarrow 2\beta$) as long as the sampled point remains within the admissible region and retains its nominal safety label. This behavior emulates the inertial buildup of momentum and allows for rapid traversal of benign interior space. Upon detecting a change in the safety status, the algorithm transitions to a bisection-based exponential decay stage, refining the bracketing pair until a prescribed tolerance ε is satisfied. This mirrors the damping phase observed in line-search corrections.

The interplay between these phases underscores the importance of hyperparameter selection. An initial β that is too small results in wasted outreach doublings, while an overly aggressive choice may cause immediate



(a) Initial gradient vectors (b) Exponential outreach. (c) Exponential decay. (d) Boundary points found.

Fig. 3. Illustration of the procedure of the AdamBA algorithm. The algorithm is divided into four stages.

Algorithm 1 Adaptive Momentum Boundary Approximation

procedure AdamBA($\epsilon, \beta, \Sigma, n, u^r, \vec{v}^r, S, S_{goal}$)
  Initialize:
  if $\vec{v}^r$ is empty **then**
    Generate n Gaussian distributed unit gradient vectors $\vec{v}_i \sim \mathcal{N}(0, \Sigma)$, $i = 1, 2, \ldots, n$
  else
    Initialize one unit gradient vector $\vec{v}_1 = \frac{\vec{v}^r}{\|\vec{v}^r\|}$
  Approximation:
  for $i = 1, 2, \cdots, n$ **do**
    Initialize the approximated boundary point $P_i = u^r$, and *stage* = *exponential outreach.*
    while *stage* = *exponential outreach* **do**
      Set $P_S \leftarrow P_i$ and $P_{NS} \leftarrow P_i$
      $P_i = P_i + \vec{v}_i \beta$
      if P_i is out of the control set **then**
        break
      if P_i safety status $\neq S$ **then**
        Set $P_{NS} \leftarrow P_i$, *stage* $\leftarrow$ *exponential decay*
        break
      $\beta = 2\beta$
    if *stage* = *exponential decay* **then**
      Apply Bisection method to locate boundary point until $\|P_{NS} - P_S\| < \epsilon$
      Set $P_i \leftarrow P_S$ if $S_{goal} = S$, $P_i \leftarrow P_{NS}$ otherwise
  Return Approximated Boundary Set P

overshoot and necessitate excessive corrective iterations. Similarly, the tolerance ε governs the trade-off between computational cost and boundary fidelity. Empirically, setting β to 5–10% of the state-space scale, using a growth factor of 2, and selecting ε commensurate with actuator resolution have been found to minimize the total number of evaluations. Additionally, periodic resampling of probe directions helps mitigate directional bias in non-stationary environments. Collectively, these design principles enable AdamBA to harness momentum-like dynamics for fast yet precise boundary localization, making it a practical tool for safety-critical control applications.

Algorithm 2 Implicit Safe Set Algorithm (ISSA)

1: **procedure** ISSA($\epsilon, \beta, \Sigma, n, u^r$)
2: **Phase 1:** ▷ Phase 1
3: Use AdamBA($\epsilon, \beta, \Sigma, n, u^r, \emptyset$, *UNSAFE*, *SAFE*) to sample a collection $\mathbb{S}$ of safe control on the boundary of $\mathcal{U}_S^D$.
4: **if** $\mathbb{S} = \emptyset$ **then**
5: Enter **Phase 2**
6: **else**
7: For each primitive action $u_i \in \mathbb{S}$, compute the deviation $d_i = \|u_i - u^r\|^2$
8: return $\arg\min_{u_i} d_i$
9:
10: **Phase 2:** ▷ Phase 2
11: Use grid sampling by iteratively increasing sampling resolution to find an anchor safe control u^a, s.t. safety status of u^a is *SAFE*.
12: Use AdamBA($\epsilon, \frac{\|u^r-u^a\|}{4}, \Sigma, 1, u^r, \frac{u^a-u^r}{\|u^a-u^r\|}$, *UNSAFE*, *SAFE*) to search for boundary point u^*
13: **if** u^* is not found **then**
14: Use AdamBA($\epsilon, \frac{\|u^r-u^a\|}{4}, \Sigma, 1, u^a, \frac{u^r-u^a}{\|u^r-u^a\|}$, *SAFE*, *SAFE*) to search for boundary point u^{a*}
15: Return u^{a*}
16: **else**
17: Return u^*

4.4 Implicit Safe Set Algorithm for Continuous-Time System

Leveraging AdamBA and the safety index design rule, we construct the implicit safe set algorithm (ISSA) to safeguard the potentially unsafe reference control. The proposed ISSA is summarized in Algorithm 2. Note that ISSA is presented under the context of discrete-time system with negligible sampling time, and $\mathcal{U}_S^D$ (line 3 of Algorithm 2) is the same as $\mathcal{U}_S$ when the sampling time is negligible. The key idea of ISSA is to use AdamBA for efficient search and grid sampling for worst cases. The inputs for ISSA are the approximation error bound (ϵ), the learning rate (β), gradient vector covariance (Σ), gradient vector number (n) and reference unsafe control (u^r). Note that ISSA is only needed when the reference control is not safe.

ISSA contains an offline stage and an online stage. In the offline stage, we synthesize the safety index according to the design rules. There are two major phases in the online stage for solving (5). In online-phase 1, we directly use AdamBA to find the safe controls on the boundary of $\mathcal{U}_S^D(x)$, and choose the control with minimum deviation from the reference control as the final output. In the case that no safe control is returned in online-phase 1 due to sparse sampling, online-phase 2 is activated. We uniformly sample the control space and deploy AdamBA again on these samples to find the safe control on the boundary of $\mathcal{U}_S^D(x)$. It can be shown in Section 5.2 that the ISSA algorithm is guaranteed to find a feasible solution of (5) since the nonempty set of safe control is guaranteed by the safety index design rule in (4). And we show in Section 5 that the solution ensures *forward invariance* in the set $\mathcal{S} = \mathcal{X}_S \cap \mathcal{L}$. Note that Phase 2 may consume significant computational resources, potentially compromising real-time performance. To avoid jerky movements caused by delayed computations, parallel processing and other acceleration techniques can be employed to speed up grid sampling.

Although the ISSA algorithm builds upon the safe set algorithm (C. Liu and Tomizuka 2014), the proposed safety index synthesis and AdamBA algorithm can be applied to other *energy function-based methods* to generate safe controls with or without an explicit analytical dynamics model.

5 Theoretical Results for ISSA When the Sampling Time is Negligible

Theorem 1 (Forward Invariance for Continuous-Time System). *If the control system satisfies the conditions in Assumption 1 and Assumption 2, and the safety index design follows the rule described in Section 4.1, the implicit safe set algorithm in Algorithm 2 guarantees the forward invariance to the set $\mathcal{S} \subseteq \mathcal{X}_S$.*

To prove the main theorem, we introduce two important propositions to show that 1) the set of safe control is always nonempty if we choose a safety index that satisfies the design rule in section 4.1; and 2) the proposed algorithm 2 is guaranteed to find a safe control if there exists one. With these two propositions, it is then straightforward to prove the *forward invariance* to the set $S \subseteq \mathcal{X}_S$. In the following discussion, we discuss the two propositions in Section 5.1 and Section 5.2, respectively. Then, we prove Theorem 1 in Section 5.3.

5.1 Feasibility of the Safety Index for Continuous-Time System

Proposition 1 (Non-emptiness of the set of safe control). *If 1) the dynamic system satisfies the conditions in Assumption 1 and Assumption 2; and 2) the safety index is designed according to the rule in Section 4.1, then the robot system in 2D plane has nonempty set of safe control at any state, i.e., $\mathcal{U}_S^D(x) \neq \emptyset, \forall x$.*

Note that the set of safe control $\mathcal{U}_S^D(x) := \{u \in \mathcal{U} \mid \phi(f(x,u)) \leq \max\{\phi(x) - \eta, 0\}\}$ is non-empty if and only if it is non-empty in the following two cases: $\phi(x) - \eta < 0$ or $\phi(x) - \eta \geq 0$. In the following discussion, we first show that the safety index design rule guarantees a non-empty set of safe control if there's only one obstacle when $\phi(x) - \eta \geq 0$ (Lemma 1). Then we show that the set of safe control is non-empty if there's only one obstacle when $\phi(x) - \eta < 0$ (Lemma 2). Finally, we leverage Lemma 1 and Lemma 2 to show $\mathcal{U}_S^D(x)$ is non-empty if there're multiple obstacles at any state.

5.1.1 *Preliminary Results.*

In this section, we will introduce some preliminary results.

Lemma 1. *If the dynamic system satisfies the conditions in Assumption 1 and Assumption 2 and there is only one obstacle in the environment, then the safety index design rule in Section 4.1 ensures that $\mathcal{U}_S^D(x) \neq \emptyset$ for x such that $\phi(x) - \eta \geq 0$.*

Proof. For x such that $\phi(x) - \eta \geq 0$, the set of safe control becomes

$$\mathcal{U}_S^D(x) = \{u \in \mathcal{U} \mid \phi(f(x,u)) \leq \phi(x) - \eta\} \tag{6}$$

According to the first condition in Assumption 2, we have $dt \to 0$. Therefore, the discrete-time approximation error approaches zero, i.e., $\phi(f(x,u)) = \phi(x) + dt \cdot \dot{\phi}(x,u) + \Delta$, where $\Delta \to 0$. Then we can rewrite (6) as:

$$\mathcal{U}_S^D(x) = \{u \in \mathcal{U} \mid \dot{\phi} \leq -\eta/dt\} \tag{7}$$

According to (3), $\dot{\phi} = -nd^{n-1}\dot{d} - k\ddot{d}$. We ignored the subscript i since it is assumed that there is only one obstacle. Therefore, the non-emptiness of $\mathcal{U}_S^D(x)$ in (7) is equivalent to the following condition

$$\forall x \text{ s.t. } \phi(x) \geq \eta, \exists u, \text{s.t. } \ddot{d} \geq \frac{\eta/dt - nd^{n-1}\dot{d}}{k}. \tag{8}$$

Note that in the 2D problem, $\ddot{d} = -a\cos(\alpha) + v\sin(\alpha)w$ and $\dot{d} = -v\cos(\alpha)$. According to Assumption 1, there is a surjection from u to $(a,w) \in W := \{(a,w) \mid a_{min} \leq a \leq a_{max}, w_{min} \leq w \leq w_{max}\}$. Moreover, according to (3), ϕ for the 2D problem only depends on α, v, and d. Hence condition $\forall x$ s.t. $\phi(x) \geq \eta$ can be translated to $\forall(\alpha, v, d)$ s.t. $\sigma + d_{min}^n - d^n - kv\cos(\alpha) \geq \eta$. Denote the later set as

$$\Phi := \{(\alpha, v, d) \mid \sigma + d_{min}^n - d^n - kv\cos(\alpha) \geq \eta, v \in [0, v_{max}], d \geq 0, \alpha \in [0, 2\pi)\}. \tag{9}$$

Consequently, condition (8) is equivalent to the following condition

$$\forall(\alpha, v, d) \in \Phi, \exists(a, w) \in W, \text{ s.t. } -a\cos(\alpha) + v\sin(\alpha)w \geq \frac{\eta/dt + nd^{n-1}v\cos(\alpha)}{k}. \tag{10}$$

According to the safety index design rule, we have $\eta = 0$. Then we show (10) holds in different cases.

Case 1: $v = 0$. In this case, we can simply choose $a = 0$, then the inequality in (10) holds.

Case 2: $v \neq 0$ and $\cos(\alpha) \leq 0$. Note that velocity v is always non-negative. Hence $v > 0$. In this case, we just need to choose $a = w = 0$, then the inequality in (10) holds, where the LHS becomes zero and the RHS becomes $\frac{nd^{n-1}v\cos(\alpha)}{k}$ which is non-positive.

Case 3: $v \neq 0$ and $\cos(\alpha) > 0$. Dividing $v\cos(\alpha)$ on both sides of the inequality and rearranging the inequality, (10) is equivalent to

$$\forall(\alpha, v, d) \in \Phi, \exists(a, w) \in W, \text{ s.t. } -\frac{a}{v} + \tan(\alpha)w - \frac{nd^{n-1}}{k} \geq 0, \tag{11}$$

and (11) can be verified by showing:

$$\min_{(\alpha,v,d)\in\Phi} \max_{(a,w)\in W} (-\frac{a}{v} + \tan(\alpha)w - \frac{nd^{n-1}}{k}) \geq 0. \tag{12}$$

Now let us expand the LHS of (12):

$$\min_{(\alpha,v,d)\in\Phi} \max_{(a,w)\in W} (-\frac{a}{v} + \tan(\alpha)w - \frac{nd^{n-1}}{k}) \tag{13}$$

$$= \min_{(\alpha,v,d)\in\Phi} (-\frac{a_{min}}{v} + [\tan(\alpha)]_+ w_{max} + [\tan(\alpha)]_- w_{min} - \frac{nd^{n-1}}{k}) \tag{14}$$

$$= \min_{\alpha\in[0,2\pi), v\in(0,v_{max}]} (-\frac{a_{min}}{v} + [\tan(\alpha)]_+ w_{max} + [\tan(\alpha)]_- w_{min} - \frac{n(\sigma + d_{min}^n + kv\cos(\alpha))^{\frac{n-1}{n}}}{k}) \tag{15}$$

$$= -\frac{a_{min}}{v_{max}} - \frac{n(\sigma + d_{min}^n + kv_{max})^{\frac{n-1}{n}}}{k}. \tag{16}$$

The first equality eliminates the inner maximization where $[\tan(\alpha)]_+ := \max\{\tan(\alpha), 0\}$ and $[\tan(\alpha)]_- := \min\{\tan(\alpha), 0\}$. The second equality eliminates d according to the constraint in Φ. The third equality is achieved when $\alpha = 0$ and $v = v_{max}$. According to the safety index design rule in Section 4.1, (16) is greater than or equal to zero. Hence (12) holds, which then implies that the inequality in (10) holds.

The three cases cover all possible situations. Hence (10) always hold and the claim in the lemma is verified. □

Lemma 2. *If the dynamic system satisfies the contidions in Assumption 1 and Assumption 2 and there is only one obstacle in the environment, then the safety index design rule in Section 4.1 ensures that $\mathcal{U}_S^D(x) = \mathcal{U}$ for any x that $\phi(x) - \eta < 0$.*

Proof. For x such that $\phi(x) - \eta < 0$, the set of safe control becomes

$$\mathcal{U}_S^D(x) = \{u \in \mathcal{U} \mid \phi(f(x, u)) \leq 0\} \tag{17}$$

According to the first assumption in Assumption 2, we have $dt \to 0$. Therefore, the discrete-time approximation error approaches zero, i.e., $\phi(f(x, u)) = \phi(x) + dt \cdot \dot{\phi}(x, u) + \Delta$, where $\Delta \to 0$. Then we can rewrite (17) as:

$$\mathcal{U}_S^D(x) = \{u \in \mathcal{U} \mid \dot{\phi} \leq -\phi/dt\} \tag{18}$$

Note that $\eta = 0$ according to the safety index design rule, then $\phi(x) - \eta < 0$ implies that $\phi(x) < 0$. Hence $-\phi/dt \to \infty$ since $dt \to 0$. Then as long as $\dot{\phi}$ is bounded, $\mathcal{U}_S^D(x) = \mathcal{U}$.

Now we show that $\dot{\phi}$ is bounded. According to (3), $\dot{\phi} = -nd^{n-1}\dot{d} - k\ddot{d}$. We ignored the subscript i since it is assumed that there is only one obstacle. According to Assumption 1, we have the state space is bounded, thus both d and $\dot{d}$ are bounded, which implies that $nd^{n-1}\dot{d}$ is bounded. Moreover, we have for all possible values of a and w, there always exists a control u to realize such a and w according to Assumption 1, which indicates the mapping from u to (a, w) is surjective. Since a and w are bounded and both can achieve zeros according to Assumption 1, we have $\forall u$, the corresponding (a, w) are bounded. Since $\ddot{d} = -a\cos(\alpha) + v\sin(\alpha)w$, then $\ddot{d}$ is bounded. Hence $\mathcal{U}_S^D(x) = \mathcal{U}$ any x that $\phi(x) - \eta < 0$ and the claim is true. □

5.1.2 Proof of Proposition 1. In this section, we will prove the Proposition 1.

Proof. If there is one obstacle, then lemma 1 and lemma 2 have proved that $\mathcal{U}_S^D(x) \neq \emptyset$ for all x. Now we need to consider the case where there are more than one obstacle but the environment is sparse in the sense that at any time step, there is at most one obstacle which is safety critical, i.e. $\phi_i \geq 0$. To show nonemptiness of $\mathcal{U}_S^D(x)$, we consider the following two cases. In the following discussion, we set $\eta = 0$ according to the safety index design rule.

Case 1: $\phi(x) = \max_i \phi_i(x) \geq 0$. Denote $j := \arg\max_i \phi_i(x)$. Since there is at most one obstacle that is safety critical, then $\phi_j(x) \geq 0$ and $\phi_k(x) < 0$ for all $k \neq j$. Denote $\mathcal{U}_{S\ j}^D(x) := \{u \in \mathcal{U} \mid \phi_j(f(x,u)) \leq \phi_j(x)\}$. Lemma 1 ensures that $\mathcal{U}_{S\ j}^D(x)$ is nonempty. Denote $\mathcal{U}_{S\ k}^D(x) := \{u \in \mathcal{U} \mid \phi_k(f(x,u)) \leq 0\}$ where $k \neq j$. Since $\phi_k(x) < 0$, lemma 2 ensures that $\mathcal{U}_{S\ k}^D(x) = \mathcal{U}$.

Note that the set of safe control can be written as:

$$\mathcal{U}_S^D(x) := \{u \in \mathcal{U} \mid \max_i \phi_i(f(x,u)) \leq \max_i \phi_i(x)\} \tag{19a}$$
$$= \{u \in \mathcal{U} \mid \max_i \phi_i(f(x,u)) \leq \phi_j(x)\} \tag{19b}$$
$$= \cap_i \{u \in \mathcal{U} \mid \phi_i(f(x,u)) \leq \phi_j(x)\} \tag{19c}$$
$$= \mathcal{U}_{S\ j}^D(x) \neq \emptyset \tag{19d}$$

Note that the last equality is due to the fact that $\{u \in \mathcal{U} \mid \phi_i(f(x,u)) \leq \phi_j(x)\} \supseteq \{u \in \mathcal{U} \mid \phi_i(f(x,u)) \leq 0\} = \mathcal{U} \supseteq \mathcal{U}_{S\ j}^D(x)$ for $i \neq j$.

Case 2: $\phi(x) = \max_i \phi_i(x) < 0$. Therefore, we have $\phi_i(x) < 0$ for all i. According to Lemma 2, $\{u \in \mathcal{U} \mid \phi_i(f(x,u)) \leq 0\} = \mathcal{U}$. Hence the set of safe control satisfies the following relationship

$$\mathcal{U}_S^D(x) := \{u \in \mathcal{U} \mid \max_i \phi_i(f(x,u)) \leq 0\} \tag{20a}$$
$$= \cap_i \{u \in \mathcal{U} \mid \phi_i(f(x,u)) \leq 0\} \tag{20b}$$
$$= \mathcal{U} \neq \emptyset \tag{20c}$$

In summary, $\mathcal{U}_S^D(x) \neq \emptyset, \forall x$ and the claim holds. □

5.2 Feasibility of ISSA

Proposition 2 (Feasibility of Algorithm 2). *If the set of safe control is non-empty, Algorithm 2 can always find a sub-optimal solution of* (5) *with a finite number of iterations.*

Algorithm 2 executes two phases consecutively where the second phase will be executed if no solution of (5) is returned in the first phase. Hence, Algorithm 2 can always find a sub-optimal solution of (5) (safe control on the boundary of $\mathcal{U}_S^D$) if the solution of (5) can always be found in phase 2.

Note that Phase 2 first finds an anchor safe control u^a, then use it with AdamBA (Algorithm 1) to find the solution of (5). In the following discussion, we first show that the safety index design rule guarantees u^a can

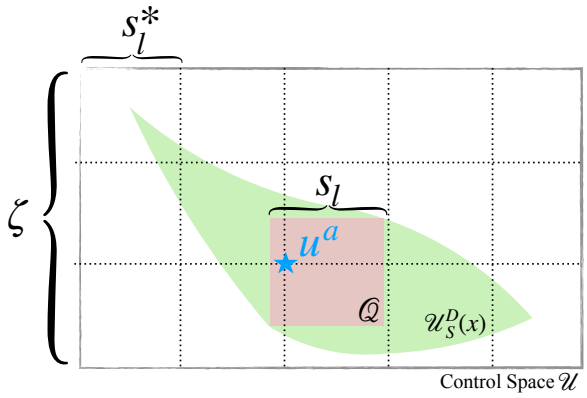


Fig. 4. Illustration of the grid sampling to find an anchor control point.

be found with finite iterations (Lemma 3). Then we show that AdamBA will return a solution if it enters the *exponential decay* phase (Lemma 4). Subsequently, we show that the evoked AdamBA procedures in phase 2 will definitely enter *exponential decay* phase (Lemma 5). Finally, we provide a upper bound of the computation iterations for Algorithm 2 at Section 5.2.2.

5.2.1 Preliminary Results. In this section, we will introduce some preliminary results.

Lemma 3 (Existence). *If the synthesized safety index can guarantee a non-empty set of safe control, then we can find an anchor point in phase 2 of Algorithm 2 with finite iterations (line 11 in algorithm 2).*

Proof. If the synthesized safety index guarantees a non-empty set of safe control $\mathcal{U}_S^D$, then there exists a hypercube inside of $\mathcal{U}_S^D$, i.e. $\exists \mathcal{Q} \subset \mathcal{U}_S^D$, where $\mathcal{Q}$ is a n_u-dimensional hypercube with the same side length of $s_l > 0$. Denote $\zeta_{[i]} = \max_{j,k} \|u^j_{[i]} - u^k_{[i]}\|$, where $u_{[i]}$ denotes the i-th dimension of control u, and $u^j \in \mathcal{U}_S^D, u^k \in \mathcal{U}_S^D$.

By directly applying grid sampling in $\mathcal{U}_s^D$ with sample interval s_l^* at each control dimension, such that $0 < s_l^* < s_l$, the maximum sampling iteration T^a for finding an anchor point in phase 2 satisfies the following condition:

$$T^a < \prod_{i=1}^{n_u} \lceil \frac{\zeta_{[i]}}{s_l^*} \rceil , \tag{21}$$

where T^a is a finite number since $s_l^* > 0$. Then we have proved that we can find an anchor point in phase 2 of Algorithm 2 with finite iteration (i.e., finite sampling time). The grid sampling to find an anchor control point is illustrated in Figure 4.

□

Lemma 4 (Convergence). *If AdamBA enters the exponential decay phase (line 16 in algorithm 1), then it can always return a boundary point approximation (with desired safety status) where the approximation error is upper bounded by ϵ.*

Proof. According to Algorithm 1, *exponential decay* phase applies Bisection method to locate the boundary point P_b until $\|P_{NS} - P_S\| < \epsilon$. Denote the returned approximated boundary point as P_{return}, according to line 21,

P_{return} is either P_{NS} or P_S, thus the approximation error satisfies:

$$\begin{aligned} \|P_{return} - P_b\| &\leq \max\{\|P_{NS} - P_b\|, \|P_S - P_b\|\} \\ &\leq \|P_{NS} - P_S\| \\ &< \epsilon. \end{aligned} \tag{22}$$

□

Lemma 5 (Feasibility). *If we enter the phase 2 of Algorithm 2 with an anchor safe control being sampled, we can always find a local optimal solution of (5).*

Proof. According to line 12-14, after an anchor safe control is being sampled, phase 2 of Algorithm 2 will evoke at most two AdamBA processes. Hence, Lemma 5 can be proved by showing one of the two AdamBA will return a local optimal solution for (5). Next we show Lemma 5 holds in two cases.

Case 1: line 12 of Algorithm 2 finds a solution.

In this case, the first AdamBA process finds a safe control u^* solution (the return of AdamBA is a set, whereas the set here has at most one element). According to Algorithm 1, a solution will be returned only if AdamBA enters *exponential decay* stage. Hence, according to Lemma 4, u^* is close to the boundary of the set of safe control with approximation error upper bounded by ϵ.

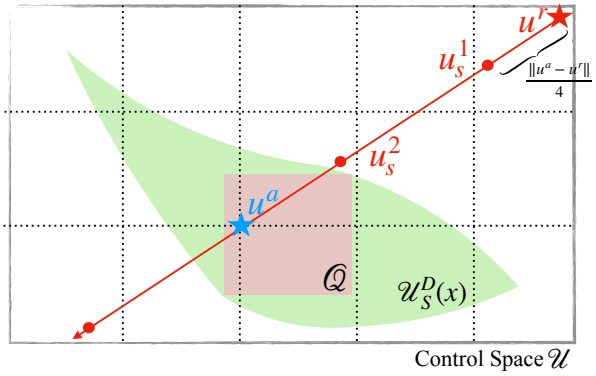


Fig. 5. Illustration of the case when it is unable to find u^*.

Case 2: line 12 of Algorithm 2 fails to find a solution.

In this case, the second AdamBA process is evoked (line 14 of Algorithm 2). Since no solution is returned from the first AdamBA process (12 of Algorithm 2), where we start from u^r and exponentially outreach along the direction $\vec{v}_a = \frac{u^a - u^r}{||u^a - u^r||}$, then all the searched control point along $\vec{v}_a$ is *UNSAFE*.

Specifically, we summarize the aforementioned scenario in Figure 5, the searched control points are represented as red dots along $\vec{v}_a$ (red arrow direction). Note that the exponential outreach starts with step size $\beta = \frac{\|u^a - u^r\|}{4}$, indicating two points $\{u_s^1, u_s^2\}$ are sampled between u^r and u^a such that

$$\begin{cases} u_s^1 = u^r + \frac{\|u^a - u^r\|}{4}\vec{v}_a = \frac{3u^r}{4} + \frac{u^a}{4} \\ u_s^2 = u^r + \frac{3\|u^a - u^r\|}{4}\vec{v}_a = \frac{u^r}{4} + \frac{3u^a}{4} \end{cases} \tag{23}$$

where the safety statuses of both u_s^1 and u_s^2 are *UNSAFE*.

During the second AdamBA process (line 14 of Algorithm 2), we start from u^a and exponentially outreach along the direction $\vec{v}_r = \frac{u^r - u^a}{||u^r - u^a||}$ with initial step size $\beta = \frac{\|u^r - u^a\|}{4}$. Hence, denote $\bar{u}_s^1$ as the first sample point along $\vec{v}_r$, and $\bar{u}_s^1$ satisfies

$$\bar{u}_s^1 = u^a + \frac{\|u^r - u^a\|}{4}\vec{v}_r = \frac{u^r}{4} + \frac{3u^a}{4}, \tag{24}$$

which indicates $\bar{u}_s^1 = u_s^2$, and the safety status of $\bar{u}_s^1$ is *UNSAFE*. Since the safe status of u^a is *SAFE* and a *UNSAFE* point can be sampled during the *exponential outreach* stage, the second AdamBA process (line 14 of Algorithm 2) will enter *exponential decay* stage. Therefore, according to Lemma 4, u^{a*} will always be returned and u^{a*} is close to the boundary of the set of safe control with approximation error upper bounded by ϵ.

The two cases cover all possible situations. Hence, after an anchor safe control u^a is sampled, phase 2 of Algorithm 2 can always find a local optima of (5).

□

5.2.2 Proof of Proposition 2. In this section, we will prove the Proposition 2.

Proof. According to Lemma 3 and Lemma 5, Algorithm 2 is able to find local optimal solution of (5). Next, we will prove Algorithm 2 can be finished within finite iterations.

According to Algorithm 2, ISSA include (i) one procedure to find anchor safe control u_a, and (ii) at most three AdamBA procedures. Firstly, based on Lemma 3, u_a can be found within finite iterations. Secondly, each AdamBA procedure can be finished within finite iterations due to:

- *exponential outreach* can be finished within finite iterations since the control space is bounded.
- *exponential decay* can be finished within finite iterations since Bisection method will exit within finite iterations.

Therefore, ISSA can be finished within finite iterations.

□

5.3 Proof of Theorem 1

Before we prove Theorem 1, we start with a preliminary result regarding $\mathcal{L}$ that is useful for proving the main theorem:

Lemma 6 (Forward Invariance of $\mathcal{L}$). *If the control system satisfies Assumption 1 and Assumption 2, and the safety index design follows the rule described Section 4.1, the implicit safe set algorithm guarantees the forward invariance to the set $\mathcal{L}$.*

Proof. If the control system satisfies the assumptions in Assumption 1 and Assumption 2, and the safety index design follows the rule described Section 4.1, then we can ensure the system has a nonempty set of safe control at any state by Proposition 1. By Proposition 2, the implicit safe set algorithm can always find a local optima solution of (5). The local optima solution always satisfies the constraint $\phi(f(x_t, u_t)) \leq \max\{\phi(x_t) - \eta, 0\}$, which indicates that 1) if $\phi(x_{t_0}) \leq 0$, then $\phi(x_t) \leq 0, \forall t \geq t_0$. Note that $\phi(x) \leq 0$ demonstrates that $x \in \mathcal{L}$. □

Now the proof the Theorem 1 is detailed below.

Proof. Leveraging Lemma 6, we then proceed to prove that the *forward invariance* to the set $\mathcal{L}$ guarantees the *forward invariance* to the set $S \subseteq \mathcal{X}_S$. Recall that $\mathcal{S} = \mathcal{X}_S \cap \mathcal{L}$. Depending on the relationship between $\mathcal{L}$ and $\mathcal{X}_S$, there are two cases in the proof which we will discuss below.

Case 1: $\mathcal{L}(\phi) = \{x|\phi(x) \leq 0\}$ is a subset of $\mathcal{X}_S = \{x|\phi_0(x) \leq 0\}$.

In this case, $\mathcal{S} = \mathcal{L}$. According to Lemma 6, If the control system satisfies the assumptions in Assumption 1 and Assumption 2, and the safety index design follows the rule described Section 4.1, the implicit safe set algorithm guarantees the *forward invariance* to the set $\mathcal{L}$ and hence $\mathcal{S}$.

Case 2: $\mathcal{L}(\phi) = \{x|\phi(x) \leq 0\}$ is NOT a subset of $\mathcal{X}_S = \{x|\phi_0(x) \leq 0\}$.

In this case, if $x_t \in \mathcal{S}$, we have $\phi_0(x_t) = \max_i \phi_{0_i}(x_t) \leq 0$, which indicates $\forall i, \phi_{0_i} \leq 0$.

Firstly, we consider the case where $\phi_{0_i}(x_t) < 0$. Note that $\phi_{0_i}(x_{t+1}) = \phi_{0_i}(x_t) + \dot{\phi}_{0_i}(x_t)dt + \frac{\ddot{\phi}_{0_i}(x_t)dt^2}{2!} + \cdots$, since the state space and control space are both bounded, and $dt \to 0$ according to Assumption 1, we have $\phi_{0_i}(x_{t+1}) \to \phi_{0_i}(x_t) \leq 0$.

Secondly, we consider the case where $\phi_{0_i}(x_t) = 0$. Since $x_t \in \mathcal{S}$, we have $\max_i \phi_i(x_t) \leq 0$, which indicates $\forall i, \sigma + d_{min}^n - d_i^n - k\dot{d}_i \leq 0$. Since $\phi_{0_i}(x_t) = 0$, we also have $d_i = d_{min}$. Therefore, the following condition holds:

$$\begin{aligned} \sigma - k\dot{d}_i &\leq 0 \\ \dot{d}_i &\geq \frac{\sigma}{k} \end{aligned} \tag{25}$$

According to the safety index design rule, we have $k, \sigma \in \mathbb{R}^+$, which indicates $\dot{d}_i > 0$. Therefore, we have $\phi_{0_i}(x_{t+1}) < 0$.

Summarizing the above two cases, we have shown that if $\phi_{0_i}(x_t) \leq 0$ then $\phi_{0_i}(x_{t+1}) \leq 0$, which indicates if $\forall i, \phi_{0_i}(x_t) \leq 0$ then $\forall i, \phi_{0_i}(x_{t+1}) \leq 0$. Note that $\forall i, \phi_{0_i}(x_{t+1}) \leq 0$ indicates that $\phi_0(x_{t+1}) = \max_i \phi_{0_i}(x_{t+1}) \leq 0$. Therefore, we have that if $x_t \in \mathcal{S}$ then $x_{t+1} \in \mathcal{X}_S$. Thus, we also have $x_{t+1} \in \mathcal{S}$ by Lemma 6. By induction, we have if $x_{t_0} \in \mathcal{S}, x_t \in \mathcal{S}, \forall t > t_0$.

In summary, by discussing the two cases of whether $\mathcal{L}$ is the subset of $\mathcal{X}_S$, we have proven that if the control system satisfies the assumptions in Assumption 1 and Assumption 2, and the safety index design follows the rule described in Section 4.1, the implicit safe set algorithm guarantees the *forward invariance* to the set $\mathcal{S} \subseteq \mathcal{X}_S$.

□

6 Implicit Safe Set Algorithm for Systems with Non-Negligible Sampling Time

The implicit safe set algorithm discussed in Section 4 is for systems with negligible sampling time ($dt \to 0$). However, in reality, many systems are implemented in a discrete-time manner with non-negligible sampling time (e.g., digital twins, physical simulators), which means that the controller may not be able to respond immediately to potential violations of safety. What is even worse is that the safe control of continuous-time systems may lead to safety violations in discrete-time systems. We demonstrate a toy problem in Section 8.1 showing that the safe control of continuous-time systems (safe control from (2)) actually violates the safety constraints when it is applied in discrete-time systems (conditions in (5)). That is mainly because the higher order terms $O(dt^2)$ is not negligible in the transition from $\phi(x_t)$ to $\phi(x_{t+1})$ when dt is not sufficiently small:

$$\phi(x_{t+1}) = \phi(x_t) + \dot{\phi}(x_t)dt + O(dt^2), \dot{\phi}(x_t) = \lim_{\tau \to 0^+} \frac{\phi(x_{t+\tau/dt}) - \phi(x_t)}{\tau}. \tag{26}$$

Therefore, the assumptions concerning sparse obstacle environment in Assumption 2 should also be justified. When $dt \to 0$, $\phi_{t+1} \to \phi_t$ according to (26), thus the safety-critical condition should be $\phi \geq 0$. However, when dt is non-negligible, there will be a gap between ϕ_{t+1} and ϕ_t which can be represented as $\Delta_{\phi_{max}}$. Then the safety-critical condition should be $\phi + \Delta_{\phi_{max}} \geq 0$, where $\Delta_{\phi_{max}}$ is the maximum possible ϕ change magnitude at one time step, i.e. $\Delta_{\phi_{max}} = \max_{x,u,t} |\phi_{t+1} - \phi_t|$.

In fact, the continuous-time system can be seen as a special case of discrete-time systems where $dt \to 0$. To that end, studying the method of safe control in discrete-time systems is essential towards the real-world application of DRL.

This section introduces the implicit safe set algorithm (ISSA) for discrete-time systems, which extends the ISSA introduced in Section 4 to be applicable for systems with non-negligible sampling time. Specifically, ISSA for discrete-time systems shares the similar pipeline as introduced in Section 4:

- **Offline:** design a safety index that ensures $\mathcal{U}_S^D(x)$ is nonempty for all x for discrete-time system.
- **Online:** project the nominal control to $\mathcal{U}_S^D(x)$ via a sample-efficient black-box optimization method during online robot maneuvers.

6.1 Safety Index Design Rule for Discrete-Time System

We start our discussion by highlighting two critical restrictions of the safety index design rule for the continuous-time system from Section 4.1 as follows:

- the safety index design rule from Section 4.1 sets $\eta = 0$, hence the robot is unable to converge to the safe set (*finite time convergence*) via solving (5) if the robot starts from the unsafe state, i.e. $\phi(x_{t_0}) > 0$.
- the safety index design rule from Section 4.1 relies on the negligible sampling time assumption to guarantee the non-emptiness of the set of safe control, which doesn't hold in discrete-time system.

In the following discussions, we will introduce the safety index design rule for the general discrete-time system that addresses the aforementioned restrictions.

6.1.1 Assumption. The discrete-time system safety index for collision avoidance in 2D will be synthesized without referring to the specific dynamic model, but under the following assumptions.

Assumption 3 (2D Collision Avoidance for Discrete-Time System). *1) The discrete-time system time step satisfies the condition:* $\frac{a_{min}}{2} + \frac{v_{max}}{4dt} > (a_m + v_{max} w_m)(-\frac{a_{min}}{v_{max}} + w_m)dt$, *where* $a_m = \max\{-a_{min}, a_{max}\}$ *and* $w_m = \max\{-w_{min}, w_{max}\}$*; 2) The acceleration and angular velocity keep constant between two consecutive time steps, i.e.* $a_{[t:t+1)} = a_t$. *3) At any given time, there can at most be one obstacle becoming safety-critical, such that* $\phi \geq -\Delta_{\phi_{max}}$ *where* $\Delta_{\phi_{max}} = \max_{x,u,t} |\phi_{t+1} - \phi_t|$ *is the maximum possible* ϕ *change magnitude at one time step (Sparse Obstacle Environment).*

These assumptions are easy to meet in practice. The first assumption ensures a bounded sampling time for discrete-time systems, which will be leveraged for the theoretical proofs in Section 7. The second assumption assumes zero-order hold (ZOH) for discrete-time control signals. The third assumption enables safety index design rule applicable with multiple moving obstacles in discrete-time system. $\Delta_{\phi_{max}}$ is bounded mainly because the system dynamics are bounded and the sampling time is also bounded.

6.1.2 Safety Index Design Rule. Following the rules in (C. Liu and Tomizuka 2014), we parameterize the safety index as $\phi = \max_i \phi_i$, and $\phi_i = \sigma + d_{min}^n - d_i^n - k\dot{d}_i$, where all ϕ_i share the same set of tunable parameters $\sigma, n, k, \eta \in \mathbb{R}^+$. Our goal is to choose these parameters such that $\mathcal{U}_S^D(x)$ is always nonempty for all possible states in the discrete-time system. Although similar to the safety index design rule in continuous-time systems, the safety index design rule for discrete-time systems is updated mainly from three aspects.

- We use nontrivial positive σ as an safety margin to enable the *forward invariance* to the invariantly safe set $\mathcal{S}$ for discrete-time systems.
- We specifically pick $n = 1$ to simplify the analysis for the higher order terms of (26) (approximation error).
- We use nontrivial positive η to enable the *finite time convergence* to the invariantly safe set $\mathcal{S}$ when the discrete-time system starts from initially unsafe states or initially safe but inevitably unsafe states in $\mathcal{X}_S$.

Safety Index Design Rule for Discrete-Time System: By setting $n = 1$ and introducing an auxiliary parameter $\eta_0 \in \mathbb{R}^+$, the parameters σ, η_0, k should be chosen such that

$$\sigma > -\dot{d}^*_{min} dt \tag{27a}$$

$$\frac{\frac{\eta_0}{dt} + v_{\max}}{k} \leq \min\{-a_{min}, a_{max}\}\,, \tag{27b}$$

where $\dot{d}^*_t = \frac{d_{t+1} - d_t}{dt}$, and $\dot{d}^*_{min}$ denotes the minimum achievable $\dot{d}^*_t$ in the system. Then the parameter η is set as

$$\eta = \eta_0 |cos(\alpha)|\,. \tag{28}$$

For the discrete-time system, we will first synthesize the η_0, n, σ, k offline first. During the online execution, η is assigned on-the-fly based on α using (28). We will show that the safety index design rule is valid to guarantee non-empty $\mathcal{U}^D_S(x)$ for discrete-time systems in Proposition 3.

6.2 Implicit Safe Set Algorithm for Discrete-Time System

The ultimate goal of the implicit safe set algorithm for discrete-time systems is to ensure (i) *forward invariance* in $\mathcal{S} = \mathcal{X}_S \cap \mathcal{L}$ when systems start from safe states, and (ii) *finite time convergence* to $\mathcal{S}$ when systems start from unsafe states. The discrete-time implicit safe set algorithm starts by applying ISSA (Algorithm 2), then applies an additional convergence trigger (CTrigger) algorithm to filter ISSA solution when $|\cos(\alpha)|$ is small and $\phi(x) > 0$.

The main body of CTrigger algorithm is summarized in Algorithm 3. The inputs for CTrigger algorithm include:

(1) system state (x),
(2) the safe control from ISSA (u),
(3) the angle between the robot's heading vector and the vector from the robot to the obstacle (α),
(4) the minimum relative acceleration (a_{min}),
(5) the maximum relative acceleration (a_{max}),
(6) the triggering angular velocity ($w_{trigger}$), where $w_{trigger} = \inf_{v \geq \frac{v_{max}}{2}} \sup_u |w|$, and
(7) the triggering angle ($\Delta_{\min}$), where $\Delta_{\min} = \inf_{|\cos(\alpha)| \leq \frac{\sqrt{3}}{2}, |w| \geq \frac{|w_{trigger}|}{2}} |\Delta\cos(\alpha)|$, and $\Delta\cos(\alpha)$ denotes the change magnitude of $\cos(\alpha)$ between two consecutive time steps.

Note that $a_{min}, a_{max}, w_{trigger}$ and $\Delta_{\min}$ are fundamental system properties, which can be evaluated offline.

Algorithm 3 Convergence Trigger

procedure CTrigger($x, u, \alpha, a_{min}, a_{max}, w_{trigger}, \Delta_{\min}$)
  if $|\cos(\alpha)| < \min\{\frac{\sqrt{3}}{2}, \frac{\Delta_{\min}}{2}\}$ and $\phi(x) > 0$ **then**
    if $v < \frac{v_{max}}{2}$ **then**
      Use uniform sampling to find a safe control u^{Trig}, s.t. (i) $a \geq \frac{\min\{-a_{min}, a_{max}\}}{2}$ when $\cos(\alpha) < 0$, and (ii) $a \leq -\frac{\min\{-a_{min}, a_{max}\}}{2}$ when $\cos(\alpha) \geq 0$.
    else
      Use uniform sampling to find a safe control u^{Trig}, s.t. $|w| \geq \frac{|w_{trigger}|}{2}$
    Return u^{Trig}
  else
    Return u

The core idea of CTrigger algorithm is to enable *finite time convergence* via preventing $|\cos(\alpha)|$ from approaching zero (according to (28)). Intuitively, this is preventing the robot from moving in a circle around the

obstacle at a constant distance, and thus preventing non-convergence. Specifically, CTrigger filters the safe control solution from ISSA when $|\cos(\alpha)|$ is less than a threshold, i.e. $\min\{\frac{\sqrt{3}}{2}, \frac{\Delta_{\min}}{2}\}$, which is accomplished via two main steps. I. Generate safe control with non-trivial acceleration to enable the system to gain enough speed (Line 4 of Algorithm 3); II. Generate safe control with non-trivial angular velocity to push $|\cos(\alpha)|$ away from zero (Line 6 of Algorithm 3). And we show in Theorem 2 that ISSA with CTrigger algorithm ensures *forward invariance* and *finite time convergence* to the set $\mathcal{S} = \mathcal{X}_S \cap \mathcal{L}$ for discrete-time systems.

Remark 1. *Note that there are three heuristic based hyper-parameters in Algorithm 3, including* $\frac{\Delta_{min}}{2}$, $\frac{\min\{-a_{min}, a_{max}\}}{2}$, *and* $\frac{|w_{trigger}|}{2}$, *and the selection heuristics are summarized as following: 1)* $\frac{\Delta_{min}}{2}$ *ensures that the triggering control will only be activated when* $|\cos(\alpha)|$ *is sufficiently small ; 2)* $\frac{\min\{-a_{min}, a_{max}\}}{2}$ *ensures that the candidate safe triggering control set for acceleration (Line 4 in Algorithm 3) is non-trivial; 3)* $\frac{|w_{trigger}|}{2}$ *ensures that the candidate safe triggering control set for rotation (Line 6 in Algorithm 3) is non-trivial. Although* $a_{min}, a_{max}, w_{trigger}$ *and* $\Delta_{\min}$ *are the fundamental system properties, it is possible to change the denominators of the hyper-parameters, e.g.* $\frac{|w_{trigger}|}{3}$, *or* $\frac{\min\{-a_{min}, a_{max}\}}{5}$. *However, there are trade-offs when we change the denominators of the hyper-parameters. Specifically, system will get out of Singularity State faster if the denominator decreases, whereas it may take longer sampling time for finding a valid triggering control. In practice, we find setting the denominator as* 2 *works well for balancing the aforementioned trade-offs.*

7 Theoretical Results for ISSA When the Sampling Time is Non-Negligible

Theorem 2 (Forward Invariance and Finite Time Convergence for Discrete-Time System). *If the control system satisfies the assumptions in Assumption 1 and Assumption 3, and the safety index design follows the rule described in Section 6.1, the discrete-time implicit safe set algorithm described in Section 6.2 guarantees the forward invariance and finite time convergence to the set* $\mathcal{S} \subseteq \mathcal{X}_S$.

To prove the main theorem, we introduce two important propositions to show that 1) the set of safe control for discrete-time systems $\mathcal{U}_S^D(x)$ is always nonempty if we choose a safety index that satisfies the design rule in section 6.1; and 2) the proposed ISSA for Discrete-Time System is guaranteed to find a safe control in finite time, i.e. either u^{Trig} or u. With these two propositions, it is then straightforward to prove the *forward invariance* and *finite time convergence* to the set $\mathcal{S} \subseteq \mathcal{X}_S$. In the following discussion, we discuss the two propositions in Section 7.1 and Section 7.2, respectively. Then, we prove Theorem 2 in Section 7.3.

7.1 Feasibility of Safety Index for Discrete-Time System

Proposition 3 (Nonempty set of safe control for Discrete-Time System). *If the dynamic system satisfies the assumptions in and Assumption 1 and Assumption 3, then the discrete-time safety index design rule in Section 6.1 ensures that the robot system in 2D plane has nonempty set of safe control at any state, i.e.,* $\mathcal{U}_S^D(x) \neq \emptyset, \forall x$.

Note that the set of safe control $\mathcal{U}_S^D(x) := \{u \in \mathcal{U} \mid \phi(f(x,u)) \leq \max\{\phi(x) - \eta, 0\}\}$ is non-empty if the following condition holds:

$$\forall x, \exists u, \text{s.t. } \phi(f(x,u)) \leq \phi(x) - \eta\,, \tag{29}$$

where

$$\phi(f(x,u)) = \phi(x) + dt \cdot \dot{\phi}(x,u) + dt^2 \cdot \frac{\ddot{\phi}(x,u)}{2} + \Delta, \tag{30}$$

and Δ is the non-negligible discrete-time approximation error.

In the following discussion, we first analysis the upper bound of Δ if there's only one obstacle (Lemma 7). Then we leverage Lemma 7 show that the safety index design rule in Section 6.1 guarantees (29) hold if there's only one obstacle (Lemma 8). Finally, we leverage Lemma 8 to show $\mathcal{U}_S^D(x)$ is non-empty if there're multiple obstacles.

7.1.1 Preliminary Results. In this section, we will introduce some preliminary results.

Lemma 7. *If the dynamic system satisfies the assumptions in Assumption 1 and Assumption 3, and there's only one obstacle in the environment, then the safety index design rule for discrete-time systems in Section 6.1 ensures the approximation error Δ for discrete-time systems is upper bounded by $\Delta_{max} dt^2 (e^{|w|dt} - 1)$, where Δ_{max} is a positive constant, dt is the discrete time step, and w is the applied angular velocity.*

Proof. First, we will discuss the derivatives of d, which is the relative distance between the robot and the obstacle. According to the zero-order hold assumption for acceleration and angular velocity from Assumption 3, we have $\dot{a} = 0$, and $\dot{w} = 0$. And we already have $\dot{d} = -v\cos(\alpha)$, then following condition holds:

$$d^{(i)} = (i-1)a(\pm\cos(\alpha)|\pm\sin(\alpha))w^{i-2} + v(\pm\sin(\alpha)|\pm\cos(\alpha))w^{i-1} \tag{31}$$

where $\pm\cos(\alpha)|\pm\sin(\alpha)$ denotes one of the terms from set
$\{\cos(\alpha), -\cos(\alpha), \sin(\alpha), -\sin(\alpha)\}$. $d^{(i)}$ denotes the i-th order derivatives of d.

According to the discrete-time system safety index design rule in Section 6.1, we have $n = 1$, thus the derivatives of safety index satisfy the following conditions:

$$\phi^{(i)} = -d^{(i)} - kd^{(i+1)} \tag{32}$$

Without loss of generality, suppose $d^{(i)} = -(i-1)a\cos(\alpha)w^{i-2} + v\sin(\alpha)w^{i-1}$, and $d^{(i+1)} = ia\sin(\alpha)w^{i-1} + v\cos(\alpha)w^i$, we have:

$$\begin{aligned}\phi^{(i)} &= -d^{(i)} - kd^{(i+1)} \\ &= -(-(i-1)a\cos(\alpha)w^{i-2} + v\sin(\alpha)w^{i-1}) - k(ia\sin(\alpha)w^{i-1} + v\cos(\alpha)w^i) \\ &= w^{i-2}\Big((i-1)a\cos(\alpha) - (v + kia)\sin(\alpha)w - kv\cos(\alpha)w^2\Big)\end{aligned} \tag{33}$$

Next, we use the Taylor expansion to represent the safety index at next time step as following:

$$\phi_{t+1} = \phi_t + \frac{\dot{\phi}_t}{1!}dt + \frac{\ddot{\phi}_t}{2!}dt^2 + \frac{\dddot{\phi}_t}{3!}dt^3 + \cdots \tag{34}$$

where each term from (34) except ϕ_t can be represented in the form of $\frac{\phi_t^{(i)}}{i!}dt^i$, $i = 1, 2, \cdots$. When $i \geq 2$, we have an important property for $\frac{\phi_t^{(i)}}{i!}dt^i$ as following:

$$\frac{\phi_t^{(i)}}{i!}dt^i = dt^2 \frac{(i-1)a\cos(\alpha) - (v + kia)\sin(\alpha)w - kv\cos(\alpha)w^2}{i!}(wdt)^{i-2} \tag{35}$$

Next, we can find the upper bound of one part of the RHS as following:

$$\begin{aligned}
&\frac{(i-1)a\cos(\alpha) - (v+kia)\sin(\alpha)w - kv\cos(\alpha)w^2}{i!} \\
&\leq \left|\frac{a\cos(\alpha)}{(i-1)!}\right| + \left|\frac{(v+k|a|)\sin(\alpha)w}{(i-1)!}\right| + \left|\frac{kv\cos(\alpha)w^2}{(i-1)!}\right| \\
&\leq \frac{a_m + v_{max}w_m + ka_mw_m + kv_{max}w_m^2}{(i-1)!} \\
&= \frac{(a_m + v_{max}w_m)(1+kw_m)}{(i-1)!}
\end{aligned} \tag{36}$$

where $a_m = \max\{-a_{min}, a_{max}\}$ and $w_m = \max\{-w_{min}, w_{max}\}$. Let's define $\Delta_{max} = (a_m + v_{max}w_m)(1+kw_m)$, and Δ_{max} is a constant given a specific k value. Based on (36) and (34), the upper bound of $\frac{\phi_t^{(i)}}{i!}dt^i$ is defined as following:

$$\begin{aligned}
&\frac{\phi_t^{(i)}}{i!}dt^i \\
&\leq dt^2\frac{\Delta_{max}}{(i-1)!}(wdt)^{i-2} \\
&\leq dt^2\frac{\Delta_{max}}{(i-1)!}(|w|dt)^{i-2} \\
&\leq \Delta_{max}dt^2\frac{(|w|dt)^{i-2}}{(i-2)!}
\end{aligned} \tag{37}$$

Recall that $\phi(x_{t+1}) = \phi(x_t) + dt\cdot\dot{\phi}(x_t) + dt^2\cdot\frac{\ddot{\phi}(x_t)}{2} + \Delta$, where Δ is the approximation error due to discrete-time systems, and $\Delta = \frac{\dddot{\phi}_t}{3!}dt^3 + \frac{\ddddot{\phi}_t}{4!}dt^4 + \cdots$. According to the Taylor series of exponential function $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$, and leverage the result from (37), the upper bound of Δ can be derived as following:

$$\begin{aligned}
\Delta &= \frac{\dddot{\phi}_t}{3!}dt^3 + \frac{\ddddot{\phi}_t}{4!}dt^4 + \cdots \\
&\leq \sum_{i=1}^{\infty}\Delta_{max}dt^2\frac{(|w|dt)^i}{i!} \\
&= \Delta_{max}dt^2(e^{|w|dt} - 1)
\end{aligned} \tag{38}$$

□

Lemma 8. *If the dynamic system satisfies the assumptions in Assumption 1 and Assumption 3, and there's only one obstacle in the environment, then the safety index design rule for discrete-time systems in Section 6.1 ensures that the robot system in 2D plane has nonempty set of safe control at any state.*

Proof. According to (3), $\dot{\phi} = -nd^{n-1}\dot{d} - k\ddot{d}$. We ignored the subscript i since it is assumed that there is only one obstacle. Therefore, according to (30), condition (29) is equivalent to the following condition

$$\forall x, \exists u, \text{s.t. } \ddot{d} \geq \frac{\frac{\eta + dt^2\cdot\frac{\ddot{\phi}(x,u)}{2} + \Delta}{dt} - nd^{n-1}\dot{d}}{k}. \tag{39}$$

To prove the non-empty set of safe control for all system state, we will prove that there always exists a safe control such that (39) holds. According to the safety index design rule for discrete-time systems in Section 6.1, we have the parameter n is chosen to be 1. Then combined with (31) and (32), the condition for (39) can be rewrote as:

$$\forall(\alpha, v), \exists(a, w), \text{ s.t.} \tag{40}$$
$$-a\cos(\alpha) + v\sin(\alpha)w \geq \frac{\frac{\eta}{dt} + dt \cdot \frac{a\cos(\alpha) - (v+2ka)\sin(\alpha)w - kv\cos(\alpha)w^2}{2} + \frac{\Delta}{dt} + v\cos(\alpha)}{k}$$

According to Lemma 7, we have $\Delta \leq \Delta_{max} dt^2(e^{|w|dt} - 1)$ and (40) can be verified by showing:

$$\forall(\alpha, v), \exists(a, w), \text{ s.t.} \tag{41}$$
$$-a\cos(\alpha) + v\sin(\alpha)w \geq \frac{\frac{\eta}{dt} + dt \cdot \frac{a\cos(\alpha) - (v+2ka)\sin(\alpha)w - kv\cos(\alpha)w^2}{2}}{k} + \frac{\Delta_{max} dt(e^{|w|dt} - 1) + v\cos(\alpha)}{k}$$

By selecting $w \to 0$, the condition for (41) becomes:

$$\forall(\alpha, v), \exists(a), \text{ s.t. } -a\cos(\alpha) \geq \frac{\frac{\eta}{dt} + dt \cdot \frac{a\cos(\alpha)}{2} + v\cos(\alpha)}{k} \tag{42}$$

There are only two scenarios $\cos(\alpha) < 0$ and $\cos(\alpha) \geq 0$.

Case 1: $\cos(\alpha) < 0$. According to safety index design rule described in Section 6.1, we have $\eta = \eta_0|\cos(\alpha)|$ and $a_{max} \geq \min\{-a_{min}, a_{max}\} \geq \frac{\frac{\eta_0}{dt} + v_{max}}{k} \geq \frac{\eta_0}{kdt}$. Thus, the following condition holds:

$$\begin{aligned} \forall(\alpha, v), \exists(a > 0), \text{ s.t. } -a\cos(\alpha) &= a|\cos(\alpha)| \\ &\geq \frac{\eta_0|\cos(\alpha)|}{kdt} \\ &\geq \frac{\frac{\eta_0|\cos(\alpha)|}{dt} + dt \cdot \frac{a\cos(\alpha)}{2} + v\cos(\alpha)}{k} \end{aligned} \tag{43}$$

which indicates (42) holds when $\cos(\alpha) < 0$.

Case 2: $\cos(\alpha) \geq 0$. According to the safety index design rule, we have $\frac{\frac{\eta_0}{dt} + v_{\max}}{k} \leq \min\{-a_{min}, a_{max}\}$. Thus, by selecting $a < 0$, the following inequality holds:

$$\max_a -a = -a_{min} \geq \frac{\frac{\eta_0}{dt} + v_{\max}}{k} \geq \frac{\frac{\eta_0}{dt} + dt \cdot \frac{a}{2} + v_{\max}}{k} \geq \frac{\frac{\eta_0}{dt} + dt \cdot \frac{a}{2} + v}{k} \tag{44}$$

Note that $\cos(\alpha) = |\cos(\alpha)|$, when $\cos(\alpha) \geq 0$. Therefore, (44) indicates the following condition holds:

$$\begin{aligned} \forall(\alpha, v), \exists(a < 0), \text{ s.t. } -a\cos(\alpha) &\geq \frac{\frac{\eta_0}{dt} + dt \cdot \frac{a}{2} + v}{k}\cos(\alpha) \\ &= \frac{\frac{\eta_0|\cos(\alpha)|}{dt} + dt \cdot \frac{a\cos(\alpha)}{2} + v\cos(\alpha)}{k} \end{aligned} \tag{45}$$

which further indicates (42) holds when $\cos(\alpha) \geq 0$.

Combine these two cases, we have proved that if the control system satisfies the assumptions in Assumption 1 and Assumption 3, then the safety index design rule in Section 6.1 ensures that the robot system in 2D plane has nonempty set of safe control at any state.

□

7.1.2 Proof of Proposition 3. In this section, we will prove the Proposition 3.

Proof. Note that the set of safe control $\mathcal{U}_S^D(x_t) := \{u \in \mathcal{U} \mid \phi(f(x,u)) \leq \max\{\phi(x) - \eta, 0\}\}$ is non-empty if it is non-empty in the following two cases: $\phi(x) \geq -\Delta_{\phi_{max}}$ and $\phi(x) < -\Delta_{\phi_{max}}$.

Case 1: Firstly, we consider the case where $\phi(x) \geq -\Delta_{\phi_{max}}$. We have $\phi(x) = \max_i \phi_i(x) \geq -\Delta_{\phi_{max}}$, where ϕ_i is the safety index with respect to the i-th obstacle. According to assumption 3, we have that at any given time, there can at most be one obstacle becoming safety critical, such that $\phi \geq -\Delta_{\phi_{max}}$ (Sparse Obstacle Environment). Therefore, $\max_i \phi_i(x) > 0$ indicates there's only one obstacle (j-th obstacle) in the environment that $\phi_j(x) \geq -\Delta_{\phi_{max}}$. Whereas for the rest of the obstacles, we have $\phi_k(x) < -\Delta_{\phi_{max}}, k \neq j$.

Denote $\mathcal{U}_{S_j}^D(x) := \{u \in \mathcal{U} \mid \phi_j(f(x,u)) \leq \max\{\phi_j(x) - \eta, 0\}\}$. According to Lemma 8, we have if the dynamic system satisfies the assumptions in Assumption 3, then the safety index design rule in Section 6.1 ensures $\mathcal{U}_{S_j}^D(x)$ is nonempty. Since $\Delta_{\phi_{max}}$ is the maximum possible ϕ change magnitude at one time step, we also have that $\phi_k(f(x,u)) \leq 0 \leq \max\{\phi_j(x) - \eta, 0\}$, for $u \in \mathcal{U}_{S_j}^D(x)$.

Therefore, we further have that if $\phi(x) > 0$, by applying $u \in \mathcal{U}_{S_j}^D(x)$, the following condition holds:

$$\phi(f(x,u)) = \max_i \phi_i(f(x,u)) \leq \max\{\phi_j(x) - \eta, 0\} = \max\{\phi(x) - \eta, 0\} \tag{46}$$

Case 2: Secondly, we consider the case where $\phi(x) < -\Delta_{\phi_{max}}$. We have $\phi(x) = \max_i \phi_i(x) < -\Delta_{\phi_{max}}$, where ϕ_i is the safety index with respect to the i-th obstacle. Therefore, we have $\forall i, \phi_i(x) < -\Delta_{\phi_{max}}$. Since $\Delta_{\phi_{max}}$ is the maximum possible ϕ change magnitude at one time step, we have that by applying $u \in \mathcal{U}$, the following condition holds:

$$\phi(f(x,u)) = \max_i \phi_i(f(x,u)) \leq 0 = \max\{\phi(x) - \eta, 0\} \tag{47}$$

In summary, if the dynamic system satisfies the assumptions in Assumption 1 and Assumption 3, then the discrete-time safety index design rule in Section 6.1 ensures that the robot system in 2D plane has nonempty set of safe control at any state, i.e., $\mathcal{U}_S^D(x) \neq \emptyset, \forall x$.

□

7.2 Feasibility of Implicit Safe Set Algorithm for Discrete-Time System

Here we define set $\mathcal{L}(\phi) := \{x | \phi(x) \leq 0\}$.

Proposition 4 (Finite Time Convergence to $\mathcal{L}$ with One Obstacle). *If the control system satisfies the assumptions in Assumption 1 and Assumption 3, and there's only one obstacle in the environment, the safety index design follows the rule described in Section 6.1, the implicit safe set algorithm with convergence trigger algorithm together guarantee the finite time convergence to the set $\mathcal{L}$.*

In the following discussion, we first proof that one important branch in Algorithm 3 (Line 6) can always find solution u^{Trig}. And then we can prove the finite time convergence described in Proposition 4.

7.2.1 Preliminary Results. In this section, we will introduce some preliminary results.

Lemma 9 (Persistent Existence of Safe Angular Velocity). *If the dynamic system satisfies the assumptions in Assumption 1 and Assumption 3, and there's only one obstacle in the environment, then the safety index design rule in Section 6.1 ensures that the robot system in 2D plane can always generate safe control such that angular velocity is non-zero for states where $|\sin(\alpha)| \geq \frac{1}{2}, v \geq \frac{v_{max}}{2}$.*

Proof. According to (41), the non-empty set of safe control for discrete-time systems can be verified by showing the following condition:

$$\forall(\alpha, v), \exists(a, w), \text{ s.t.}$$
$$-a\cos(\alpha) + v\sin(\alpha)w \geq \frac{\frac{\eta}{dt} + dt \cdot \frac{a\cos(\alpha) - (v+2ka)\sin(\alpha)w - kv\cos(\alpha)w^2}{2}}{k} + \frac{\Delta_{max}dt(e^{|w|dt} - 1) + v\cos(\alpha)}{k}$$

The proof of Lemma 8 has shown that by setting $w \to 0$, the condition (42) holds:

$$\forall(\alpha, v), \exists(a), \text{ s.t. } -a\cos(\alpha) \geq \frac{\frac{\eta}{dt} + dt \cdot \frac{a\cos(\alpha)}{2} + v\cos(\alpha)}{k}$$

Thus, by subtracting the components of (42) from (41), we can verify there exists non-empty set of safe control where angular velocity is non-zero at states where $|\sin(\alpha)| \geq \frac{1}{2}, v \geq \frac{v_{max}}{2}$, by showing the following condition holds:

$$\forall(|\sin(\alpha)| \geq \frac{1}{2}, v \geq \frac{v_{max}}{2}, a), \exists(|w| > 0), \text{ s.t.}, \tag{48}$$
$$v\sin(\alpha)w \geq \frac{dt \cdot \frac{-(v+2ka)\sin(\alpha)w - kv\cos(\alpha)w^2}{2} + \Delta_{max}dt(e^{|w|dt} - 1)}{k}$$

Note that w can either be positive or negative, by selecting w such that $\sin(\alpha)w > 0$, showing (48) is equivalent to show the following condition holds:

$$\forall(|\sin(\alpha)| \geq \frac{1}{2}, v \geq \frac{v_{max}}{2}, a), \exists(|w| > 0), \text{ s.t.}, \tag{49}$$
$$(\frac{kv}{dt} + \frac{v + 2ka}{2})|\sin(\alpha)||w| + \frac{kv\cos(\alpha)|w|^2}{2} - \Delta_{max}(e^{|w|dt} - 1) \geq 0$$

Denote $F(|w|) = (\frac{kv}{dt} + \frac{v+2ka}{2})|\sin(\alpha)||w| + \frac{kv\cos(\alpha)|w|^2}{2} - \Delta_{max}(e^{|w|dt} - 1)$, we have $\nabla_{|w|}F(|w|) = (\frac{kv}{dt} + \frac{v+2ka}{2})|\sin(\alpha)| + kv\cos(\alpha)|w| - dt\Delta_{max}e^{|w|dt}$. Since $\Delta_{max} = (a_m + v_{max}w_m)(1 + kw_m)$, We have the following condition hold:

$$\nabla_{|w|}F(0) = (\frac{kv}{dt} + \frac{v + 2ka}{2})|\sin(\alpha)| - dt\Delta_{max} \tag{50}$$
$$> k(\frac{\frac{v}{dt} + a}{2} - dt(a_m + v_{max}w_m)(\frac{1}{k} + w_m))$$

According to the safety index design rule in Section 6.1, we have $\frac{v_{\max}}{k} \leq \frac{\frac{\eta_0}{dt} + v_{\max}}{k} \leq \min\{-a_{min}, a_{max}\} \leq -a_{min}$, which indicates $0 < \frac{1}{k} \leq \frac{-a_{min}}{v_{max}}$. We also have $\frac{a_{min}}{2} + \frac{v_{max}}{4dt} > (a_m + v_{max}w_m)(-\frac{a_{min}}{v_{max}} + w_m)dt$ according to Assumption 3. Therefore, when $v > \frac{v_{max}}{2}$, the lower bound for (50) is summarized as following:

$$\begin{aligned}\nabla_{|w|}F(0) &> k(\frac{\frac{v}{dt}+a}{2} - dt(a_m + v_{max}w_m)(\frac{1}{k}+w_m)) \\ &> k(\frac{\frac{v}{dt}+a}{2} - dt(a_m + v_{max}w_m)(\frac{-a_{min}}{v_{max}}+w_m)) \\ &> k(\frac{\frac{v_{max}}{2dt}+a_{min}}{2} - dt(a_m + v_{max}w_m)(\frac{-a_{min}}{v_{max}}+w_m)) \\ &> 0\end{aligned} \tag{51}$$

Since $F(0) = 0$, $\nabla_{|w|}F(0) > 0$, and $\nabla_{|w|}F(|w|)$ is differentiable everywhere, there exists $|w_{trigger}| > 0$, such that $\forall |w| \in [0, |w_{trigger}|]$, (49) holds.

□

7.2.2 Proof of Proposition 4. In this section, we will prove the Proposition 4.

Proof. According to the safety index design rule, we have that $\eta = \eta_0 |cos(\alpha)|$. Next, we will discuss the only two situations for $|\cos(\alpha)|$:

Case 1: $|\cos(\alpha)| \geq \min\{\frac{\sqrt{3}}{2}, \frac{\Delta_{\min}}{2}\}$. In this case, Algorithm 3 won't be triggered, thus the properties of finite iterations convergence follows Algorithm 2.

Case 2.1: $|\cos(\alpha)| < \min\{\frac{\sqrt{3}}{2}, \frac{\Delta_{\min}}{2}\}$ and $v < \frac{v_{max}}{2}$. In this case, CTrigger algorithm is activated to filter the control solution from ISSA, which seeks to find a safe control u^{Trig} such that $a_t \geq \frac{\min\{-a_{min}, a_{max}\}}{2}$ when $\cos(\alpha) < 0$ and $a_t \leq -\frac{\min\{-a_{min}, a_{max}\}}{2}$ when $\cos(\alpha) \geq 0$ and $w \to 0$. The existence of u^{Trig} is guaranteed by the (43), (45) from the proof of Lemma 8, and can be found through uniform sampling within finite iterations by Lemma 3.

Case 2.2: $|\cos(\alpha)| < \min\{\frac{\sqrt{3}}{2}, \frac{\Delta_{\min}}{2}\}$ and $v \geq \frac{v_{max}}{2}$. According to Assumption 3, we have that $|a_t|$ is constant between two consecutive time steps, which indicates the following property holds:

(P1) When $|\cos(\alpha)| < \min\{\frac{\sqrt{3}}{2}, \frac{\Delta_{\min}}{2}\}$, CTrigger algorithm ensures $v \geq \frac{v_{max}}{2}$ within at most $\frac{v_{max}}{\min\{-a_{min}, a_{max}\}}$ iterations

In this case, we have CTrigger algorithm seeks to find a safe control u^{Trig} such that $|w_t| \geq \frac{|w_{trigger}|}{2}$. Note that $|\cos(\alpha)| < \frac{\sqrt{3}}{2}$ indicates $|\sin(\alpha)| > \frac{1}{2}$, thus the existence of u^{Trig} is guaranteed by the lemma 9, and can be found through uniform sampling within finite iterations by Lemma 3.

Suppose at time step t, $|\cos(\alpha_t)| < \min\{\frac{\sqrt{3}}{2}, \frac{\Delta_{\min}}{2}\}$ and $v_t \geq \frac{v_{max}}{2}$. Denote $\Delta_t \cos(\alpha)$ as the change of $\cos(\alpha)$ at time step t after applying u^{Trig}, we have $|\Delta_t \cos(\alpha)| \geq \Delta_{\min} > \frac{\Delta_{\min}}{2} > |\cos(\alpha_t)|$, which indicates:

$$\begin{aligned}|\cos(\alpha_{t+1})| &= |\cos(\alpha_t) + \Delta_t \cos(\alpha)| \\ &\geq \Big||\Delta_t \cos(\alpha)| - |\cos(\alpha_t)|\Big| \\ &\geq \Big|\Delta_{\min} - |\cos(\alpha_t)|\Big| \\ &> \min\{\frac{\sqrt{3}}{2}, \frac{\Delta_{\min}}{2}\}\end{aligned} \tag{52}$$

which further shows that $\eta_{t+1} > \eta_0 \min\{\frac{\sqrt{3}}{2}, \frac{\Delta_{\min}}{2}\}$.

Therefore, according to property item (P1), ISSA with CTrigger ensures $\eta \geq \eta_0 \min\{\frac{\sqrt{3}}{2}, \frac{\Delta_{\min}}{2}\}$ in at most every $\frac{v_{max}}{\min\{-a_{min}, a_{max}\}} + 1$ time steps. Thus, if $\phi(x_{t_0}) > 0$, then $\phi(x_{t_1}) \leq 0$ for finite time $t_1 > t_0$, where $t_1 - t_0 < \frac{\phi(x_{t_0})}{\eta_0 \min\{\frac{\sqrt{3}}{2}, \frac{\Delta_{\min}}{2}\}}(\frac{v_{max}}{\min\{-a_{min}, a_{max}\}} + 1)$.

□

7.3 Proof of Theorem 2

7.3.1 Preliminary Results. In this section, we will introduce some preliminary results before proofing the Theorem 2.

Lemma 10 (Forward Invariance and Finite Time Convergence of $\mathcal{L}$). *If the control system satisfies the assumptions in Assumption 1 and Assumption 3, and the safety index design follows the rule described in Section 6.1, the implicit safe set algorithm with convergence trigger algorithm together guarantee the forward invariance and finite time convergence to the set $\mathcal{L}$.*

Proof. Firstly, we will prove the *forward invariance* of set $\mathcal{L}$. If the control system satisfies the assumptions in Assumption 1 and Assumption 3, and the safety index design follows the rule described Section 6.1, then we can ensure the system has nonempty set of safe control at any state by Proposition 3. By Proposition 2, implicit safe set algorithm can always find local optima solution of (5). The local optima solution always satisfies the constraint $\phi(f(x_t, u_t)) \leq \max\{\phi(x_t) - \eta, 0\}$, which indicates that if $\phi(x_{t_0}) \leq 0$, then $\phi(x_t) \leq 0, \forall t \geq t_0$ since $\eta \geq 0$. Note that $\phi(x) \leq 0$ demonstrates that $x \in \mathcal{L}$. Thus the forward invariance of $\mathcal{L}$ is proved.

Secondly, we will prove the *finite time convergence* to $\mathcal{L}$. Suppose at time step t_0, we have $\phi(x_{t_0}) = \max_i \phi_i(x_{t_0}) > 0$. According to Sparse Obstacle Environment assumption from Assumption 3, there's only one obstacle (j-th obstacle) in the environment that $\phi_j(x_{t_0}) > 0$. Whereas for the rest of the obstacles, we have $\phi_k(x_{t_0}) < -\Delta_{\phi_{max}}, k \neq j$.

According to Proposition 4, we have implicit safe set algorithm together with convergence trigger algorithm guarantee that $\phi_j(x_{t_1}) \leq 0$ for a finite time $t_1 > t_0$ while $\phi_j(x_{t_1-1}) > 0$. By Sparse Obstacle Environment assumption, we also have $\forall k \neq j, \phi_k(x_{t_1-1}) < -\Delta_{\phi_{max}}$, hence $\forall k \neq j, \phi_k(x_{t_1}) \leq 0$. Therefore, $\phi(x_{t_1}) = \max_i \phi_i(x_{t_1}) \leq 0$ and it demonstrates that $x_{t_1} \in \mathcal{L}$.

□

7.3.2 Proof the Theorem 2. In this section, we will prove the Theorem 2.

Proof. Leveraging Lemma 10, we then proceed to prove that the *forward invariance* and *finite time convergence* to the set $\mathcal{L}$ guarantees the *forward invariance* and *finite time convergence* to the set $S \subseteq \mathcal{X}_S$. Recall that $\mathcal{S} = \mathcal{X}_S \cap \mathcal{L}$. Depending on the relationship between $\mathcal{L}$ and $\mathcal{X}_S$, there are two cases in the proof which we will discuss below.

Case 1: $\mathcal{L}(\phi) = \{x|\phi(x) \leq 0\}$ is a subset of $\mathcal{X}_S = \{x|\phi_0(x) \leq 0\}$.

In this case, $\mathcal{S} = \mathcal{L}$. According to Proposition 1 and Lemma 10, if the safety index design follows the rule described in Section 4.1, the implicit safe set algorithm guarantees the *forward invariance* and *finite time convergence* to the set $\mathcal{L}$ and hence $\mathcal{S}$.

Case 2: $\mathcal{L}(\phi) = \{x|\phi(x) \leq 0\}$ is NOT a subset of $\mathcal{X}_S = \{x|\phi_0(x) \leq 0\}$.

1) To prove *finite time convergence* in this case, recall the forms of safety index are $\phi_i = \sigma + d_{min}^n - d_i^n - k\dot{d}_i$ and $\phi_{0_i} = d_{min} - d_i$. Since the finite time convergence to $\mathcal{L}$ is proved in Lemma 10, we then discuss the finite time convergence of $\mathcal{X}_S$ based on the fact that $\phi(x) \leq 0$. Note that the set $\mathcal{L}$ can be divided into two subsets $\mathcal{L} \setminus \mathcal{S} = \{x \mid \phi(x) \leq 0, \phi_0(x) > 0\}$ and $\mathcal{S} = \{x \mid \phi(x) \leq 0, \phi_0(x) \leq 0\}$. If $x_{t_0} \in \mathcal{L} \setminus \mathcal{S}$, meaning the following two

conditions hold:

$$\max_i d_{min} - d_i > 0 \tag{53}$$

$$\forall i, \sigma + d_{min}^n - d_i^n - k\dot{d}_i \leq 0 \tag{54}$$

Thus, the following condition hold:

$$\forall i_{\phi_{0_i}>0}, \sigma - k\dot{d}_i \leq d_i^n - d_{min}^n < 0 \tag{55}$$

$$\dot{d}_i > \frac{\sigma}{k}$$

(55) indicates that $\forall i, \phi_{0_i}(x_t) \leq 0$ for some finite time $t > t_0$, which shows $\phi_0(x_t) \leq 0$ for some finite time $t > t_0$. Therefore, we have proved *finite time convergence* to the set $\mathcal{S} \subseteq \mathcal{X}_S$.

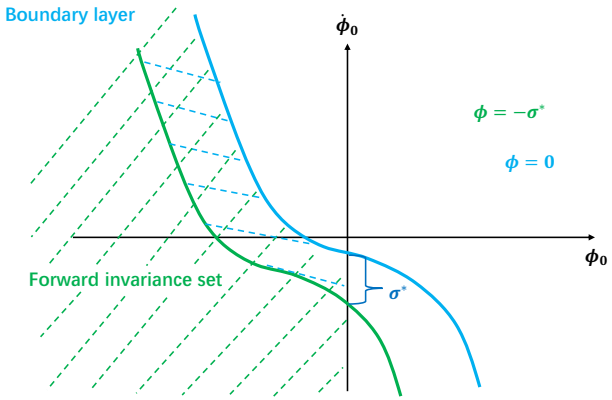


Fig. 6. The illustration of *forward invariance* under the constraint of $\phi(x_{t+1}) \leq \max\{\phi(x_t) - \eta, 0\}) - \sigma^*$.

2) To prove the *forward invariance* in this case, we first need to prove that if $x_t \in \mathcal{S}$, then $x_{t+1} \in \mathcal{X}_S$.

we first define σ^* such that $(\sigma + d_{min}^n)^{\frac{1}{n}} > d_{min} + \sigma^*$. For $x_{t_0} \in \mathcal{S}$ and for each $d_{t_{0_i}}$, there are only two situations for $d_{t_{0_i}}$:

2.1) $d_{min} + \sigma^* \geq d_{t_{0_i}} \geq d_{min}$. And we have $x_t \in \mathcal{S}$, thus the following condition holds:

$$\phi(x_t) \leq 0 \Rightarrow \sigma + d_{min}^n - d_{t_i}^n - k\dot{d}_{t_i} \leq 0 \tag{56}$$

$$d_{t_i} \leq d_{min} + \sigma^* \Rightarrow -d_{t_i}^n \geq -(d_{min} + \sigma^*)^n = -(\sigma + d_{min}^n) + \iota$$

where $\iota = \sigma + d_{min}^n - (d_{min} + \sigma^*)^n$ is a positive constant. Thus, the following condition holds:

$$\begin{aligned} \dot{d}_{t_i} &\geq \frac{\sigma + d_{min}^n - d_{t_i}^n}{k} \\ &\geq \frac{\sigma + d_{min}^n - (\sigma + d_{min}^n) + \iota}{k} \\ &\geq \frac{\iota}{k} > 0 \end{aligned} \tag{57}$$

which indicates the d_i will increase at next time step, then $\phi_{0_i}(x_{t+1}) \leq \phi_{0_i}(x_t) \leq 0$. Thus in this case, if $x_t \in \mathcal{S}$, then $x_{t+1} \in \mathcal{X}_S$.

2.2) $d_{t_{0_i}} > d_{min} + \sigma^*$. Note that $x_t \in \mathcal{S}$ and $d_{t_i} > d_{min} + \sigma^*$ indicate $\sigma + d_{min}^n - d_{t_i}^n - k\dot{d}_{t_i} \leq 0$ and $d_{min} - d_{t_i} < 0$, whereas $\dot{d}$ can either < 0 or ≥ 0.

If $\dot{d}_{t_i} \geq 0$, ϕ_{0_i} keeps decreasing, then $\phi_{0_i}(x_{t+1}) \leq \phi_{0_i}(x_t) \leq 0$.

If $\dot{d}_{t_i} < 0$, $\phi_{0_i}(x_{t+1}) \leq 0$ can be guaranteed if $d_{t+1_i} = d_{t_i} + \dot{d}_i^* dt \geq d_{min}$, where $\dot{d}_i^* \neq \dot{d}_i$ due to discrete-time systems. Denote $\dot{d}_{min}^*$ as the minimum $\dot{d}^*$ can be achieved in the system, the following condition holds:

$$\begin{aligned} d_{t+1_i} &= d_{t_i} + \dot{d}_i^* dt \\ &\geq d_{t_i} + \dot{d}_{min}^* dt \end{aligned} \tag{58}$$

According to (27a) from the safety index design rule for discrete-time system in Section 6.1, we have $\dot{d}_{min}^* dt > -\sigma^*$, then:

$$\begin{aligned} d_{t+1_i} &\geq d_{t_i} + \dot{d}_{min}^* dt \\ &> d_{t_i} - \sigma^* \\ &> d_{min} + \sigma^* - \sigma^* \\ &> d_{min} \end{aligned} \tag{59}$$

which also indicates $\phi_{0_i}(x_{t+1}) \leq 0$. Thus in this case, if $x_t \in \mathcal{S}$, then $x_{t+1} \in \mathcal{X}_S$.

Summarizing the above content, we have shown that if $x_t \in \mathcal{S}$, then $\forall i, \phi_{0_i}(x_{t+1}) \leq 0$, which indicates $\phi_0(x_{t+1}) = \max_i \phi_{0_i}(x_{t+1}) \leq 0$, hence $x_{t+1} \in \mathcal{X}_S$. We also have if $x_t \in \mathcal{S}$, $x_{t+1} \in \mathcal{L}$ by Lemma 10. Therefore, if $x_t \in \mathcal{S}$, $x_{t+1} \in \mathcal{S}$. By induction, we have shown if $x_{t_0} \in \mathcal{S}$, $x_t \in \mathcal{S}, \forall t > t_0$. Thus the forward invariance of $\mathcal{S}$ is proved.

Leveraging the results introduced above, in practice, we can ensure the *forward invariance* of $\mathcal{S} = \{x \mid \phi(x) \leq 0, \phi_0(x) \leq 0\}$ by using the safe control generation constraint $\phi(x_{t+1}) \leq \max\{\phi(x_t) - \eta, 0\}) - \sigma^*$ through adding an extra safety boundary σ^*. The resulting *forward invariance* set and newly added boundary layer are illustrated in Figure 6, where the *forward invariance* set is the green area and the boundary layer is the blue area.

In summary, by discussing the two cases of whether $\mathcal{L}$ is the subset of $\mathcal{X}_S$, we have proved that if the safety index design follows the rule described in Section 4.1, the implicit safe set algorithm guarantees the *forward invariance* and *finite time convergence* to the set $S \subseteq \mathcal{X}_S$.

□

8 Experimental Results

In our experiments, we aim to answer the following questions:

- **Q1**: How does the discrepancy between continuous-time safe control and discrete-time safe control affect the evolution of the safety index in discrete-time systems? (Answered in Section 8.1)
- **Q2**: How does ISSA compare with other state-of-the-art methods for safe RL? Can ISSA achieve zero-violation of the safety constraint? (Answered in Section 8.3)
- **Q3**: How does the design of the safety index affect the set of safe control? (Answered in Section 8.4)
- **Q4**: How do the hyper-parameters of ISSA and the dimensionality of the system impact its performance? (Answered in Section 8.5)

8.1 Toy Problem Experiment Details

To demonstrate the discrepancy between continuous-time systems and discrete-time systems, we build a toy problem environment to show that directly applying the result from (2) in discrete-time systems will lead to safety violations. The toy robot is a 3-state unicycle model with state $x = [p_x, p_y, \theta]$, where p_x, p_y denote the coordinates on the x-axis and y-axis, respectively, and θ is the heading direction. The control inputs of the toy environment are $[v, w]$, where v denotes the velocity of the robot, and w denotes the angular velocity of the robot.

Next, we define the underlying dynamics of the robot for the toy environment in control affine form as follows:

$$x_{t+1} = x_t + \dot{x}_t dt = x_t + \begin{bmatrix} \cos(\theta)dt & 0 \\ \sin(\theta)dt & 0 \\ 0 & dt \end{bmatrix} u \tag{60}$$

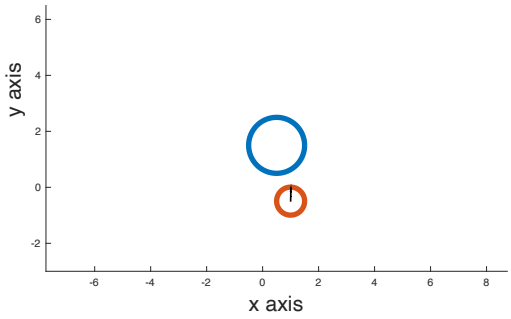


Fig. 7. The toy experimental platform. The blue circle is a static obstacle, and red circle is the robot, whose heading direction is illustrated as the black arrow.

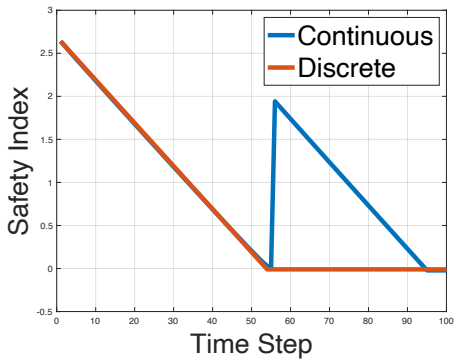


Fig. 8. Safety Index evolution curves comparison.

8.1.1 Experimental Setup: Toy Problem. The toy environment is illustrated in **??**. Note that the environment may contain multiple obstacles, and the robot is required to be collision-free with all the obstacles. However, we can always consider the safety constraint between the robot and its closest obstacle at each time step. The justification is summarized in the following remark.

Remark 2. *If the robot is collision-free with its closest obstacle at each time step, then the robot is collision-free with environmental obstacles at each time step.*

Therefore, in **??** we only consider one obstacle. For simplicity, we currently assume the obstacle is static, which can be easily extended to moving obstacles.

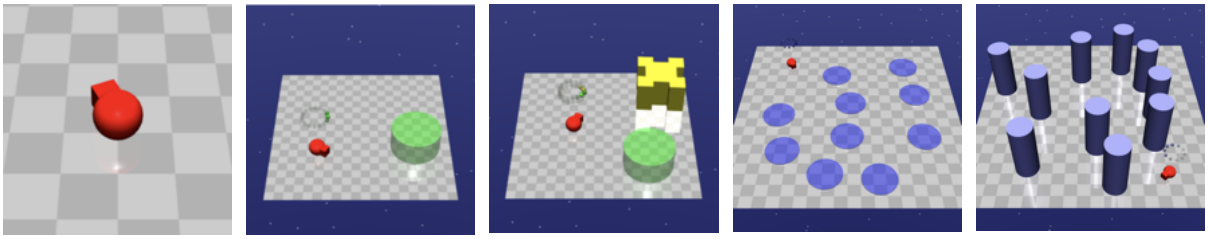
Next, we define the safety index $\phi(x)$ by constraining the perpendicular line from the center of the obstacle to the robot heading direction to be larger than $(R + r)$, which is shown as follows:

$$\phi(x) = (r + R)^2 - ((p_{x_0} - p_x)\sin(\theta) - (p_{y_0} - p_y)\cos(\theta))^2 \tag{61}$$

where r, R are the radii of the robot and obstacle, respectively. $[p_{x_0}, p_{y_0}]$ is the location of the obstacle. It can be easily shown that $\phi(x) < 0$ will ensure the robot is collision-free with environmental obstacles. Finally, we design a dummy nominal policy to control the point robot to move forward with constant velocity, which has no safety guarantee. Therefore, if the initial system state is unsafe, our proposed methods are expected to generate the safe control to gradually drive the system state to the safe set where $\phi(x) < 0$, and remain in the safe set.

8.1.2 Results: Toy Problem. For toy problem, we simulate the system for 100 time steps, and system $dt_{system} = 0.01$. We compare the safety index evolution for the safe control generated by: 1) solving (5) with ISSA (i.e., $dt_{ISSA} = 0.01 = dt_{system}$ for ISSA simulation) and 2) sloving (2) with ISSA (i.e., $dt_{ISSA} = 0.00001 \ll dt_{system}$ for ISSA simulation). The comparison result is demonstrated in **??**.

We can see that the safe control generated by solving continuous-time system problem (2) fails to ensure the safety index is monotonically decreasing due to (26), where the safety index suddenly increased to a large positive value at the time step between 50 and 60. However, the safe control generated by solving discrete-time



(a) Point robot: a simple 2D robot that can turn and move.

(b) Goal: navigating the robot inside the green goal area.

(c) Push: pushing the yellow box inside the green goal area.

(d) Hazards: non-physical dangerous areas.

(e) Pillars: fixed dangerous obstacles

Fig. 9. The environmental settings for benchmark problems in Safety Gym.

system problem (5) ensures the safety index is monotonically decreasing until below zero. Therefore, **??** strongly supports the discrepancy between the discrete-time system and the continuous-time system.

8.2 Safety Gym Experiment Details

In recent years, numerous benchmark environments for safe control have emerged (Ji et al. 2024; Ray et al. 2019; Sun et al. 2025; Zhao, Chen, Sun, R. Liu, et al. 2024). We adopt Safety Gym (Ray et al. 2019) as our testing platform to evaluate the effectiveness of the proposed implicit safe set algorithms. Our experiments adopt the Point robot ($\mathcal{U} \subseteq \mathbb{R}^2$) as shown in Figure 9a and the Doggo robot ($\mathcal{U} \subseteq \mathbb{R}^{12}$) as shown in Figure 1. We design 8 experimental environments with different task types, constraint types, constraint numbers and constraint sizes. We name these environments as `{Task}-{Constraint Type}{Constraint Number}-{Constraint Size}`. Note that `Constraint Size` equals d_{min} in the safety index design. Two tasks are considered:

- `Goal`: The robot must navigate to a goal as shown in Figure 9b.
- `Push`: The robot must push a box to a goal as shown in Figure 9c.

And two different types of constraints are considered:

- `Hazard`: Dangerous (but admissible) areas as shown in Figure 9d. Hazards are circles on the ground. The agent is penalized for entering them.
- `Pillar`: Fixed obstacles as shown in Figure 9e. The agent is penalized for hitting them.

The methods in the comparison group include: unconstrained RL algorithm PPO (Schulman et al. 2017) and constrained safe RL algorithms PPO-Lagrangian, CPO (Achiam et al. 2017) and PPO-SL (PPO-Safety Layer) (Dalal et al. 2018). We select PPO as our baseline method since it is state-of-the-art and already has safety-constrained derivatives that can be tested off-the-shelf. We set the limit of cost to 0 for both PPO-Lagrangian and CPO since we aim to avoid any violation of the constraints. To make sure ISSA can complete tasks while guaranteeing safety, we use a PPO agent as the nominal policy and ISSA as a safety layer to solve (5), we call this structure as PPO-ISSA, and it is illustrated in Figure 10. Such safety layer structure has also been used in PPO-SL (Dalal et al. 2018) which leverages offline dataset to learn a linear safety-signal model and then construct a safety layer via analytical optimization.

For all experiments, we use neural network policies with separate feedforward MLP policy and value networks of size (256, 256) with tanh activations. More details are as follows.

8.2.1 Environment Settings. In this section, we will introduce our environment settings.

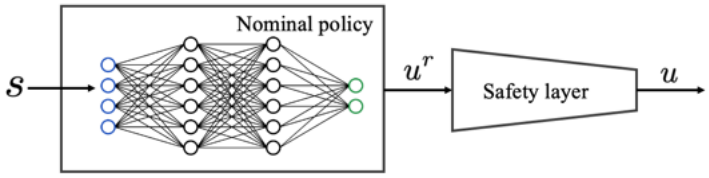


Fig. 10. The structural safe agent architecture.

Goal Task. In the Goal task environments, the reward function is:

$$r(x_t) = d^g_{t-1} - d^g_t + \mathbb{1}[d^g_t < R^g] \,,$$

where d^g_t is the distance from the robot to its closest goal and R^g is the size (radius) of the goal. When a goal is achieved, the goal location is randomly reset to someplace new while keeping the rest of the layout the same.

Push Task. In the Push task environments, the reward function is:

$$r(x_t) = d^r_{t-1} - d^r_t + d^b_{t-1} - d^b_t + \mathbb{1}[d^b_t < R^g] \,,$$

where d^r and d^b are the distance from the robot to its closest goal and the distance from the box to its closest goal, and R^g is the size (radius) of the goal. The box size is 0.2 for all the Push task environments. Like the goal task, a new goal location is drawn each time a goal is achieved.

Hazard Constraint. In the Hazard constraint environments, the cost function is:

$$c(x_t) = \max(0, R^h - d^h_t) \,,$$

where d^h_t is the distance to the closest hazard and R^h is the size (radius) of the hazard.

Pillar Constraint. In the Pillar constraint environments, the cost $c_t = 1$ if the robot contacts with the pillar otherwise $c_t = 0$.

Black-box Dynamics. The underlying dynamics of Safety Gym is directly handled by MuJoCo physics simulator (Todorov et al. 2012). This indicates the dynamics is not explicitly accessible but rather can be implicitly evaluated, which is suitable for our proposed implicit safe set algorithm. The implementation of block-box dynamics for ISSA is through simulation in the MuJoCo physics simulator and recovering to the pre-simulated state.

State Space. The state space is composed of various physical quantities from standard robot sensors (accelerometer, gyroscope, magnetometer, and velocimeter) and lidar (where each lidar sensor perceives objects of a single kind). The state spaces of all the test suites are summarized in Table 1. Note that Vase is another type of constraint in Safety Gym (Ray et al. 2019) and all the returns of vase lidar are zero vectors (i.e., $[0, 0, \cdots, 0] \in \mathbb{R}^{16}$) in our experiments since none of our eight test suites environments have vases.

Control Space. For all the experiments, the control space $\mathcal{U} \subset \mathbb{R}^2$. The first dimension $u_1 \in [-10, 10]$ is the control space of moving actuator, and second dimension $u_2 \in [-10, 10]$ is the control space of turning actuator. For each actuator the maximum torque corresponds to $a_{\max} = 2.5\,\mathrm{m/s^2}$ and $\omega_{\max} = 4.0\,\mathrm{rad/s}$; within these limits the inner-loop PID reaches the desired (a, w) in $\leq 6\,\mathrm{ms}$, satisfying Assumption 1.

Table 1. The state space components of different test suites environments.

State Space Option	Goal-Hazard	Goal-Pillar	Push-Hazard
Accelerometer ($\mathbb{R}^3$)	✓	✓	✓
Gyroscope ($\mathbb{R}^3$)	✓	✓	✓
Magnetometer ($\mathbb{R}^3$)	✓	✓	✓
Velocimeter ($\mathbb{R}^3$)	✓	✓	✓
Goal Lidar ($\mathbb{R}^{16}$)	✓	✓	✓
Hazard Lidar ($\mathbb{R}^{16}$)	✓	✗	✓
Pillar Lidar ($\mathbb{R}^{16}$)	✗	✓	✗
Vase Lidar ($\mathbb{R}^{16}$)	✓	✓	✓
Box Lidar ($\mathbb{R}^{16}$)	✗	✗	✓

8.2.2 Policy Settings. Detailed parameter settings are shown in Table 2. All the policies in our experiments use the default hyper-parameter settings hand-tuned by Safety Gym (Ray et al. 2019) except the cost limit = 0 for PPO-Lagrangian and CPO.

Table 2. Important hyper-parameters of PPO, PPO-Lagrangian, CPO, PPO-SL and PPO-ISSA

Policy Parameter	PPO	PPO-Lagrangian	CPO	PPO-SL & PPO-ISSA
Timesteps per iteration	30000	30000	30000	30000
Policy network hidden layers	(256, 256)	(256, 256)	(256, 256)	(256, 256)
Value network hidden layers	(256, 256)	(256, 256)	(256, 256)	(256, 256)
Policy learning rate	0.0004	0.0004	(N/A)	0.0004
Value learning rate	0.001	0.001	0.001	0.001
Target KL	0.01	0.01	0.01	0.01
Discounted factor γ	0.99	0.99	0.99	0.99
Advantage discounted factor λ	0.97	0.97	0.97	0.97
PPO Clipping ϵ	0.2	0.2	(N/A)	0.2
TRPO Conjugate gradient damping	(N/A)	(N/A)	0.1	(N/A)
TRPO Backtracking steps	(N/A)	(N/A)	10	(N/A)
Cost limit	(N/A)	0	0	(N/A)

8.2.3 Safety Index Settings. The parameters of safety index design are summarized in Table 3, where we adopt $k = 0.375$ for test suites with constraint size of 0.05 and $k = 0.5$ for test suites with constraint size of 0.15.

8.2.4 Algorithm 3 Parameter Settings. In the simulation environment, the parameters $a_{min}, a_{max}, w_{trigger}$, and $\Delta_{\min}$ are computed offline based on the system dynamics and sampled trajectories. Specifically, a_{min} and a_{max} are obtained by applying extreme control inputs and recording the resulting relative accelerations. The triggering angular velocity $w_{trigger}$ is evaluated by sampling feasible angular velocities under different control inputs at high-speed regimes ($v \geq v_{max}/2$) and taking the infimum of the corresponding suprema. Finally, $\Delta_{\min}$ is estimated

Table 3. Experiment-specific parameters of safety index design for PPO-ISSA.

Safety Index Parameter	Constraint size = 0.05	Constraint size = 0.15
n	1	1
k	0.375	0.5
η	0	0

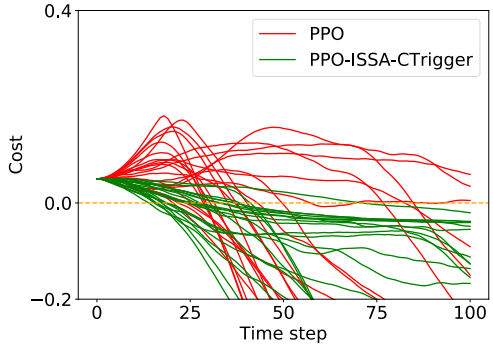


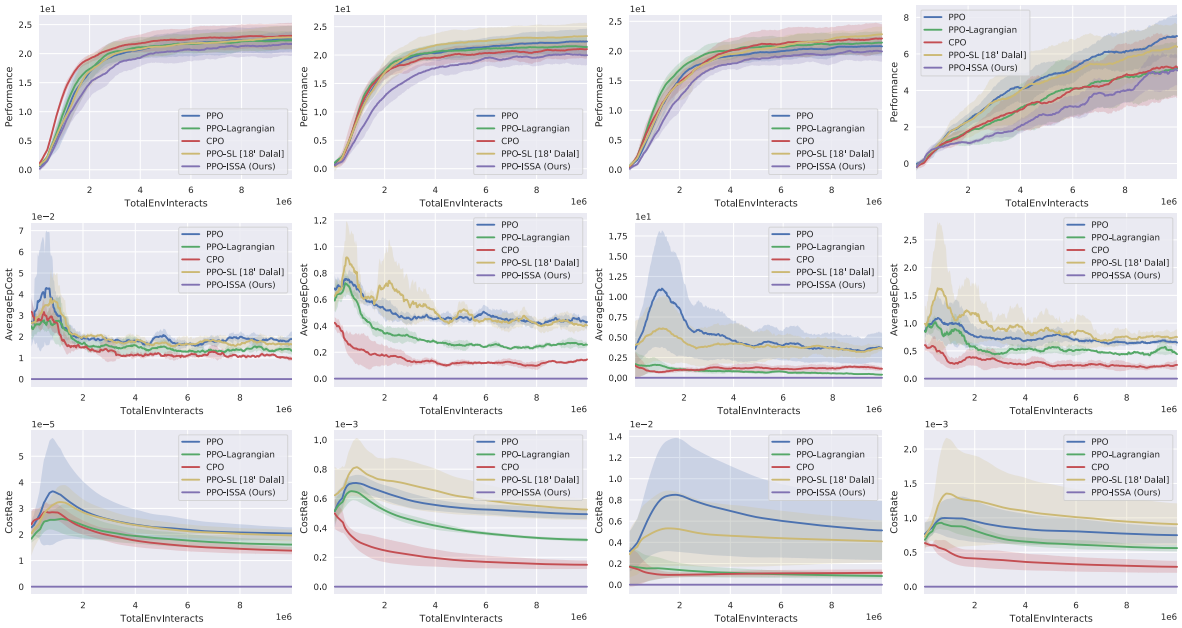
Fig. 11. Cost changes over 100 time steps of PPO and PPO-ISSA-CTrigger starting from the same unsafe state over 20 trails.

by simulating trajectories with various approach angles that satisfy $|\cos(\alpha)| \leq \sqrt{3}/2$ and $|w| \geq |w_{trigger}|/2$, and then measuring the change in $\cos(\alpha)$ between consecutive time steps; the minimum recorded value is selected as $\Delta_{\min}$.

8.3 Evaluating PPO-ISSA and Comparison Analysis

To compare the reward and safety performance of PPO-ISSA to the baseline methods in different tasks, constraint types, and constraint sizes, we design 4 test suites with 4 constraints which are summarized in Figure 12. The comparison results reported in Figure 12 demonstrate that PPO-ISSA is able to achieve zero average episode cost and zero cost rate across all experiments while slightly sacrificing the reward performance. The baseline soft safe RL methods (PPO-Lagrangian and CPO) fail to achieve zero-violation safety even when the cost limit is set to be 0. Moreover, the safety advantage of safe RL baseline methods over unconstrained RL method (PPO) becomes trivial as the constraint number and constraint size decrease as shown in Figure 13a, where the cost rate and average episode cost of PPO-Lagrangian, CPO and PPO are nearly the same when there is only one constraint with size 0.05.

PPO-Lagrangian and CPO fail since both methods rely on trial-and-error to enforce constraints while ISSA is able to guarantee *forward invariance* by Theorem 1. We also observe that PPO-SL fails to lower the violation during training, due to the fact that the linear approximation of cost function $c(x_{t+1}) \approx c(x_t) + g(x_t, w)^T u$ (Dalal et al. 2018) becomes inaccurate when the dynamics are highly nonlinear like the ones we used in MuJoCo (Todorov et al. 2012). More importantly, PPO-SL cannot guarantee that these always exist a feasible safe control to lower the cost, since they directly use the user defined cost function which cannot always ensure feasibility. More detailed metrics for comparison and experimental results on test suites with 1 constraint are summarized in Section 8.3.1.



(a) Goal-Hazard4-0.05 (b) Goal-Hazard4-0.15 (c) Goal-Pillar4-0.15 (d) Push-Hazard4-0.15

Fig. 12. Average episodic return, episodic cost and overall cost rate of constraints of PPO-ISSA and baseline methods on 4-constraint environments over five seeds.

8.3.1 Metrics Comparison. In this section, we report all the results of eight test suites by three metrics defined in Safety Gym (Ray et al. 2019):

- The average episode return J_r.
- The average episodic sum of costs M_c.
- The average cost over the entirety of training ρ_c.

The average episode return J_r and the average episodic sum of costs M_c were obtained by averaging over the last five epochs of training to reduce noise. Cost rate ρ_c was just taken from the final epoch. We report the results of these three metrics in Table 4 normalized by PPO results. We calculate the converged reward $\bar{J}_r$ percentage of PPO-ISSA compared to other three safe RL baseline methods (PPO-Lagrangian, CPO and PPO-SL) over eight control suites. The computed mean reward percentage is 95%, and the standard deviation is 9%. Therefore we conclude that PPO-ISSA is able to gain 95% ± 9% cumulative reward compared to state-of-the-art safe DRL methods.

Note that in safety-critical environments, there is always a tradeoff between reward performance and safety, where safety guarantees prevent aggressive strategies for seeking high reward. On the other hand, the provable safety is prominently weighted in the tradeoff, since any safety violation may lead to property loss, life danger in the real robotics applications. Therefore, PPO-ISSA does a better job in terms of balancing the tradeoff between provable safety and task performance.

To validate the *finite time convergence* of Theorem 2 that ISSA combined with CTrigger algorithm (PPO-ISSA-CTrigger) can ensure the *finite time convergence* to $\mathcal{X}_S$ for discrete-time system, we further compare the cost evolution of PPO and PPO-ISSA-CTrigger agents when starting from the same unsafe state (i.e., cost > 0). The

Table 4. Normalized metrics obtained from the policies at the end of the training process, which is averaged over eight test suits environments and five random seeds.

(a) Goal-Hazard1-0.05

Algorithm	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$
PPO	1.00	1.00	1.00
PPO-Lagrangian	1.003	1.587	0.859
CPO	1.012	1.052	0.944
PPO-SL [18' Dalal]	1.038	1.031	1.110
PPO-ISSA (Ours)	**1.077**	**0.000**	**0.000**

(b) Goal-Hazard4-0.05

Algorithm	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$
PPO	1.000	1.000	1.000
PPO-Lagrangian	0.983	0.702	0.797
CPO	**1.022**	0.549	0.676
PPO-SL [18' Dalal]	1.014	0.923	0.963
PPO-ISSA (Ours)	0.961	**0.000**	**0.000**

(c) Goal-Hazard1-0.15

Algorithm	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$
PPO	1.000	1.000	1.000
PPO-Lagrangian	**1.086**	0.338	0.760
CPO	1.011	0.553	0.398
PPO-SL [18' Dalal]	1.018	0.898	1.048
PPO-ISSA (Ours)	1.008	**0.000**	**0.000**

(d) Goal-Hazard4-0.15

Algorithm	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$
PPO	1.000	1.000	1.000
PPO-Lagrangian	0.948	0.581	0.645
CPO	0.932	0.328	0.303
PPO-SL [18' Dalal]	**1.038**	0.948	1.063
PPO-ISSA (Ours)	0.895	**0.000**	**0.000**

(e) Goal-Pillar1-0.15

Algorithm	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$
PPO	1.000	1.000	1.000
PPO-Lagrangian	0.968	0.196	0.239
CPO	0.976	0.328	0.494
PPO-SL [18' Dalal]	1.017	0.948	1.063
PPO-ISSA (Ours)	**1.056**	**0.000**	**0.000**

(f) Goal-Pillar4-0.15

Algorithm	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$
PPO	1.000	1.000	1.000
PPO-Lagrangian	1.035	0.105	0.159
CPO	1.060	0.304	0.221
PPO-SL [18' Dalal]	**1.094**	1.055	0.780
PPO-ISSA (Ours)	0.965	**0.000**	**0.000**

(g) Push-Hazard1-0.15

Algorithm	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$
PPO	1.000	1.000	1.000
PPO-Lagrangian	**1.124**	0.356	0.384
CPO	0.872	0.231	0.228
PPO-SL [18' Dalal]	1.107	0.685	0.610
PPO-ISSA (Ours)	0.841	**0.000**	**0.000**

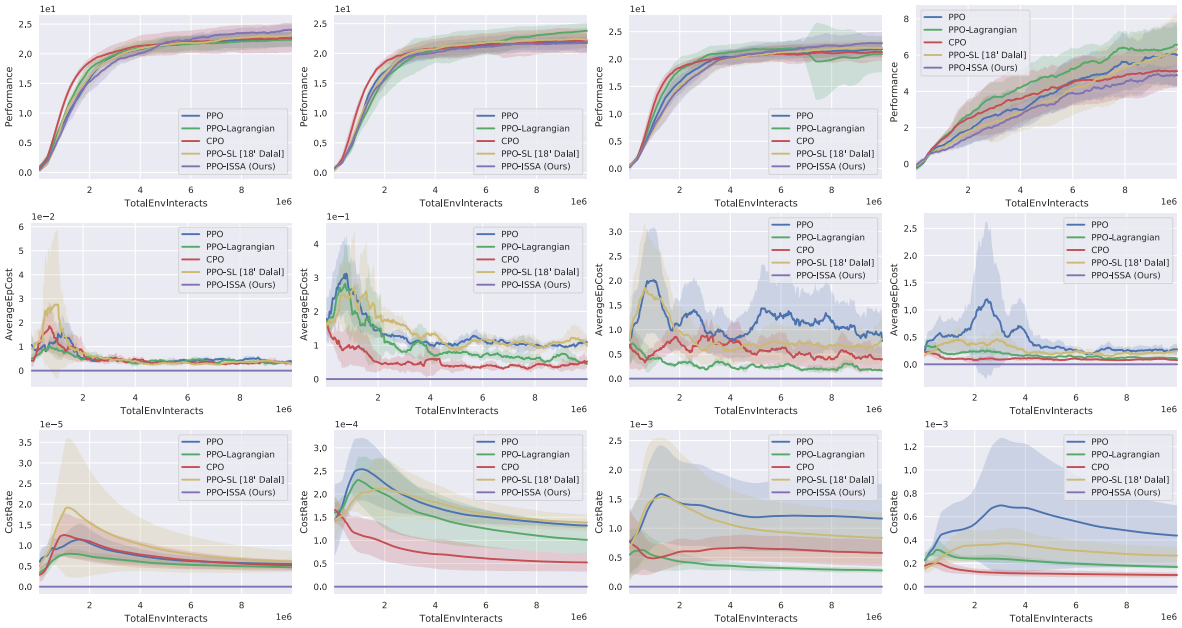
(h) Push-Hazard4-0.15

Algorithm	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$
PPO	**1.000**	1.000	1.000
PPO-Lagrangian	0.72	0.631	0.748
CPO	0.758	0.328	0.385
PPO-SL [18' Dalal]	0.914	1.084	1.212
PPO-ISSA (Ours)	0.727	**0.000**	**0.000**

comparison results are shown in Figure 11, which shows that PPO-ISSA-CTrigger can converge to $\mathcal{X}_S$ within 100 time steps across all experiments while cost evolution of PPO agents fluctuates wildly without preference to converge to safe set. The cost changes of PPO-ISSA-CTrigger aligns with ours theory of *finite time convergence*.

8.4 Feasibility of Safety Index Synthesis

To demonstrate how the set of safe control is impacted by different safety index definition, we randomly pick an unsafe state x^* such that $\phi(x^*) > 0$, and visualize the corresponding set of safe control $\mathcal{U}_S^D$ under different safety index definitions, which are shown in Figure 15. Red area means $\Delta\phi > 0$ (i.e. unsafe control) and blue area means $\Delta\phi < 0$ (i.e. safe control). Figure 15a shows the set of safe control of distance safety index $\phi_d = \sigma + d_{min} - d$, which is the default cost definition of Safety Gym. The heatmap is all red in Figure 15a, which means that the set of safe control under the default ϕ_d is empty. Figure 15b shows the set of safe control of the synthesized safety index $\phi = \sigma + d_{min}^2 - d^2 - k\dot{d}$ with different value of k. With the synthesized safety index, Figure 15b demonstrates that the size of the set of safe control grows as k increases, which aligns with the safety index synthesis rule discussed in Section 4.1 as larger k is easier to satisfy (4). To demonstrate the reward performance of PPO-ISSA under



(a) Goal-Hazard1-0.05 (b) Goal-Hazard1-0.15 (c) Goal-Pillar1-0.15 (d) Push-Hazard1-0.15

Fig. 13. Average performance of PPO-ISSA and baseline methods on 1-constraint environments over five random seeds. The three rows represent average episodic return, average episodic cost and overall cost rate of constraints. The safety advantage of safe RL baseline methods over unconstrained RL method (PPO) becomes trivial as the constraint number and constraint size decrease, where the cost rate and average episode cost of PPO-Lagrangian, CPO and PPO are nearly the same when there is only one constraint with size 0.05.

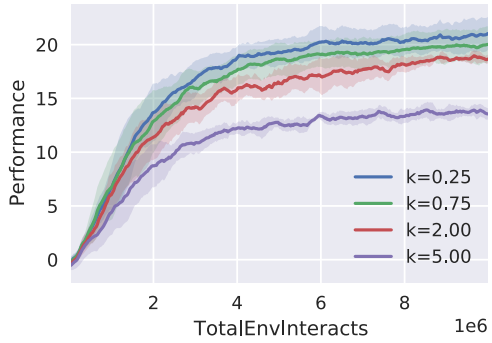
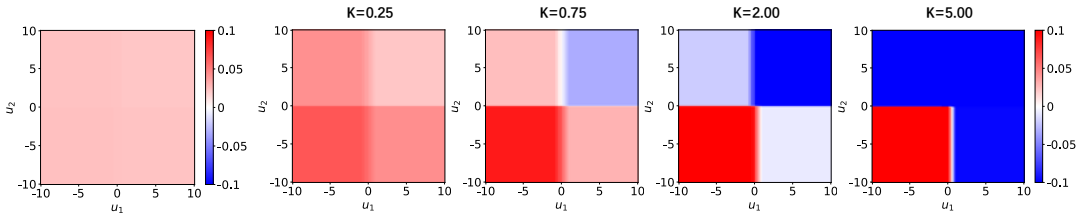


Fig. 14. Average return of PPO-ISSA with different safety index design on Goal-Hazard4-0.15.

different safety index designs, we select Goal-Hazard4-0.15 test suite. Figure 14 demonstrates the average return of PPO-ISSA under different value of k, which shows that the reward performance of PPO-ISSA deteriorates as k value increases (since larger k makes the control more conservative). Note that the set of safe control increases as the k value increases, thus the optimal k should be the smallest k that makes the set of safe control nonempty for all states. Our safety index synthesis rule in (4) provides the condition to pick the optimal k.



(a) Distance Safety Index (b) Synthesized Safety Index

Fig. 15. Heat maps of the difference of safety index $\Delta\phi = \phi(f(x,u)) - \phi(x)$. The x-axis u_1 represents the control space of moving actuator, and the y-axis u_2 represents the control space of turning actuator.

Table 5. Normalized computation time and return under different number of vectors in ISSA. These results are average on 100 ISSA runs over five random seeds on Goal-Hazard4-0.15.

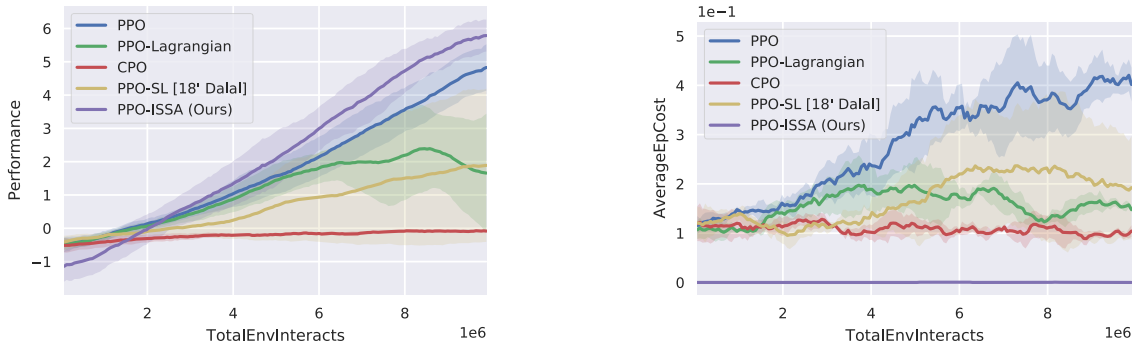
Number of vectors	Simulation Time T_{sim}	Overall ISSA Time T_{all}	Return $\bar{J}_r$
$n = 3$	0.297	0.301	0.738
$n = 5$	0.504	0.511	0.826
$n = 10$	0.987	1.000	1.000

8.5 Ablation Study

8.5.1 Sensitivity Analysis. To demonstrate the scalability and the performance of PPO-ISSA when ISSA chooses different parameters, we conduct additional tests using the test suite Goal-Hazard4-0.15. Among all input parameters of ISSA, the gradient vector number n is critical to impact the quality of the solution of (5). Note in the limit when $n \to \infty$, ISSA is able to traverse all boundary points of the set of safe control, hence able to find the global optima of (5). We pick three different n values: 3, 5, 10; and report the average episode reward of PPO-ISSA, and the computation time of ISSA when solving (5), which includes the normalized average ISSA computation time and the normalized average simulation time for each run. The results are summarized in Table 5, which demonstrates that the reward performance of PPO-ISSA would improve as n gets bigger since we get better optima of Equation (5). In practice, we find that the reward performance will stop improving when n is big enough ($n > 10$). The computation time scales linearly with respect to n while the majority (98%) of computation cost is used for environment simulation, which can be improved in the future by replacing the simulator with a more computationally efficient surrogate model.

8.5.2 Scalability Analysis. The safe control algorithm ISSA is based on the sampling method algorithm 1 AdamBA. In this section, we demonstrate the scalability of AdamBA and ISSA in systems with higher dimensions of state and control.

8.5.3 ISSA in Higher Dimensional Control Systems. We test ISSA with a doggo robot as shown in Figure 17 with 12 dimensional control space and 80 dimensional state space. We evaluate ISSA with the doggo robot in the Goal-Hazard1-0.15 suite. As shown in Figure 18, ISSA is able to guarantee zero-violation in higher dimensional control systems. We notice that in system with higher control dimensions, safe RL methods like PPO-Lagrangian and CPO perform poorly compared to PPO, which demonstrates the constrained RL algorithms struggle to learn good reward performance for complex locomotion behavior. Similar comparison results are also reported in Safety Gym benchmarks (Ray et al. 2019). In contrast, PPO-ISSA is able to achieve the best reward performance while



(a) Average reward (b) Average cost

Fig. 16. Average episodic return and episodic cost of PPO-ISSA and baseline methods on Goal-hazard1-0.15 environment of a doggo robot over five seeds.

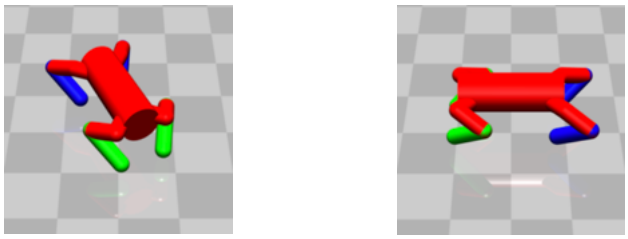
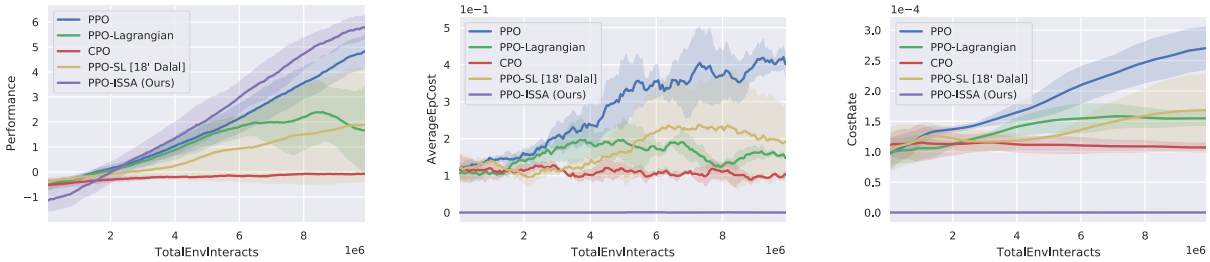


Fig. 17. doggo robot: a quadrupedal robot with bilateral symmetry with 12-dimensional control space.



(a) Average episode reward (b) Average episode cost (c) Average episode costrate

Fig. 18. Average performance of PPO-ISSA and baseline methods on Goal-hazard1-0.15 environment of doggo robot over five seeds. The three columns represent average episode return, average episode cost and overallcost rate of constraints.

guaranteeing zero safety violation, showing the scalability of ISSA to achieve satisfying reward performance in systems with higher control dimensions.

To illustrate the computation cost of searching for a safe control using ISSA in systems with higher control dimensions, we apply ISSA to find safe controls for both Point robot and doggo robot in the Goal-Hazard1-0.15

Table 6. Average Computation time, succuss ratio of ISSA phase 1 and number of safe control candidates of 200 ISSA runs on Goal-Hazard1-0.15.

	Number of vectors	Overall (non-parallel) ISSA Time T_{all} (s)	ISSA phase 1 success rate	Number of safe control
Point robot Control dimension = 2	$n = 3$	0.023	0.962	1.193
	$n = 5$	0.038	1.000	2.258
	$n = 10$ **(used)**	0.076	1.000	4.576
	$n = 20$	0.160	1.000	9.838
	$n = 40$	0.302	1.000	18.055
	$n = 100$	0.737	1.000	25.740
Doggo robot Control dimension = 12	$n = 3$	0.068	0.802	2.041
	$n = 5$	0.116	0.903	3.051
	$n = 10$	0.228	0.925	5.127
	$n = 20$ **(used)**	0.461	0.981	10.673
	$n = 40$	0.910	0.990	19.373
	$n = 100$	2.190	1.000	33.910

suite with different number of unit gradient vectors generated by guassian distribution. Note that we desire to encourage ISSA phase 1 to find safe control, while the functionality of ISSA phase 2 is the fail-safe strategy for ISSA phase 1 to ensure feasibility of ISSA since 1) ISSA phase 2 is relatively more computational expensive than ISSA phase 1 due to random sampling, and 2) ISSA phase 2 can only return one safe control candidate. Thus, we are especially interested in analyzing the performance of ISSA phase 1.

We report the average computation time, ISSA phase 1 success rate and number of safe control candidates found by ISSA phase 1 under different robot types and number of unit gradient vectorss in Table 6, where the success of ISSA phase 1 is defined as it returns at least 1 safe control which may lead to a large deviation from the original nominal control. As demonstrated in Table 6, we only need 20 unit gradient vectors for 12 dimensional control space doggo robot to achieve satisfying success rate in ISSA phase 1 and number of safe control candidates. On the other hand, we need 10 unit gradient vectors for 2 dimensional control space Point robot. Therefore, higher control dimensionality will not necessarily increase the computation cost exponentially for ISSA to find safe control. We also notice that, even with the same number of vectors, the computaion time of doggo robot is 3 times longer than that of Point robot due to the doggo robot simulation takes longer than point robot simulaton per step in MuJoCo simulator. Here we also highlight a fact that the process of AdamBA outreach/decay for each unit gradient vector is independent from each other, thus we can always accelerate ISSA by parallel computation, which is then discussed in Section 8.5.4.

8.5.4 AdamBA in higher dimensional space. As reported in Section 8.5.3, in MuJoCo environment, the sampling cost of AdamBA is not exponentially increasing as the control dimensions increase, where we only need 10 vectors for point robot (2-dimensional control space) and only need 20 vectors for doggo robot (12-dimensional control space). However, let's consider a worse case for AdamBA, where the relative size of safe control space in the entire control space is exponentially decreasing with dimension linearly increases, which could result in the computation cost of AdamBA to find the boundary exponentially increasing. We synthesis a simple control problem, where only half of the control space is safe for each dimension, which means the portion of safe control space to the entire control space is $\frac{1}{2^n}$ where n is the number of control space dimensions. Note that for every unit gradient vector, the process of AdamBA outraech/decay is independent of the other unit gradient vectors, which means we could utilize parallel computation in practice when applying real-time robots. We report the average computation time of AdamBA on the prescribed toy control problem in Figure 19, where we can see that the

computation cost of parallel AdamBA remains nearly the same as the number of vectors exponentially increases, which shows the potential capability of AdamBA scaling to higher dimensional real-time control systems.

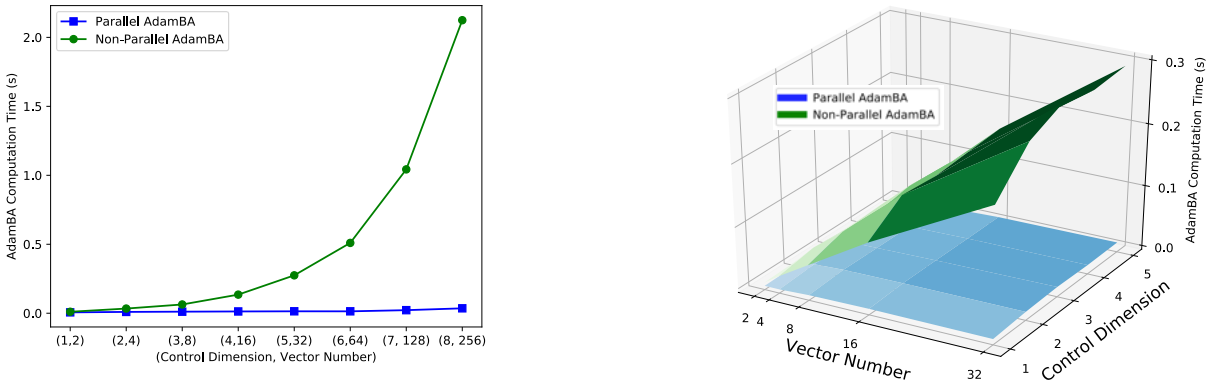


Fig. 19. The average computation time of non-parallel AdamBA and parallel AdamBA on toy problem with different control dimensions and number of vectors.

9 Conclusion and Future Work

Safety guarantee is critical for robotic applications in real world, such that robots can persistently satisfy safety constraints. This paper presents a model-free safe control strategy to synthesize safeguards for DRL agents, which will ensure zero safety violation during training. In particular, we present an implicit safe set algorithm as a safeguard, which synthesizes the safety index (also called the barrier certificate) and the subsequent safe control law only by querying a black-box dynamics function (e.g., a digital twin simulator). The theoretical results indicate that the synthesized safety index guarantees nonempty set of safe control for all system states, and ISSA guarantees *finite time convergence* and *forward invariance* to the safe set for both continuous-time and discrete-time system. We further validate the proposed safeguard with DRL on state-of-the-art safety benchmark Safety Gym. Our proposed method achieves zero safety violation and 95% ± 9% reward performance compared to state-of-the-art safe DRL methods.

There are three major directions for future work. Firstly, we will further generalize the safety index synthesis rule to cover a wider range of applications other than collision avoidance in 2D. Secondly, we will further speed up the implicit model evaluation step by replacing the physical engine based simulator with a learned surrogate model while taking the learned dynamics error into account. Third, obstacle avoidance tasks in densely cluttered environments — where multiple safety-critical constraints may be active simultaneously — still require further theoretical advancements and algorithmic exploration.

Acknowledgments

This work is in part supported by Amazon Research Award and National Science Foundation through Grant #2144489.

References

J. Abou-Chakra, L. Sun, K. Rana, B. May, K. Schmeckpeper, N. Suenderhauf, M. V. Minniti, and L. Herlant. 2025. *Real-is-Sim: Bridging the Sim-to-Real Gap with a Dynamic Digital Twin.* (2025). arXiv: 2504.03597.

J. Achiam, D. Held, A. Tamar, and P. Abbeel. 2017. "Constrained policy optimization." In: *International Conference on Machine Learning*. PMLR, 22–31.

A. D. Ames, J. W. Grizzle, and P. Tabuada. 2014. "Control barrier function based quadratic programs with application to adaptive cruise control." In: *53rd IEEE Conference on Decision and Control*. IEEE, 6271–6278.

L. Armijo. 1966. "Minimization of functions having Lipschitz continuous first partial derivatives." *Pacific Journal of mathematics*, 16, 1, 1–3.

F. P. Bejarano, L. Brunke, and A. P. Schoellig. Jan. 2025. "Safety Filtering While Training: Improving the Performance and Sample Efficiency of Reinforcement Learning Agents." *IEEE Robotics and Automation Letters*, 10, 1, (Jan. 2025), 788–795. doi:10.1109/lra.2024.3512374.

F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause. 2017. "Safe Model-based Reinforcement Learning with Stability Guarantees." *Advances in Neural Information Processing Systems*, 30.

R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick. 2019. "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33, 3387–3395.

Y. Chow, O. Nachum, A. Faust, E. Duenez-Guzman, and M. Ghavamzadeh. 2019. "Lyapunov-based safe policy optimization for continuous control." *ICML 2019 Workshop RL4RealLife*. arXiv: 1901.10031.

R. K. Cosner, P. Culbertson, A. J. Taylor, and A. D. Ames. 2023. "Robust Safety under Stochastic Uncertainty with Discrete-Time Control Barrier Functions." In: *Robotics: Science and Systems (RSS)*. arXiv: 2302.07469.

G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa. 2018. "Safe exploration in continuous action spaces." *CoRR*, abs/1801.08757.

S. K. Das, M. H. Uddin, and S. Baidya. 2022. "Edge-assisted Collaborative Digital Twin for Safety-Critical Robotics in Industrial IoT." In: *IEEE International Conference on Sensing, Communication, and Networking (SECON Demo)*. arXiv: 2209.12854.

D. Du, S. Han, N. Qi, H. B. Ammar, J. Wang, and W. Pan. 2023. "Reinforcement Learning for Safe Robot Control using Control Lyapunov Barrier Functions." In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 9442–9448. doi:10.1109/ICRA48891.2023.10160991.

Y. Emam, G. Notomista, P. Glotfelter, Z. Kira, and M. Egerstedt. 2025. "Safe Reinforcement Learning Using Robust Control Barrier Functions." *IEEE Robotics and Automation Letters*, 10, 3, 2886–2893. doi:10.1109/LRA.2022.3216996.

J. Ferlez, M. Elnaggar, Y. Shoukry, and C. Fleming. 2020. "Shieldnn: A provably safe nn filter for unsafe nn controllers." *CoRR*, abs/2006.09564.

J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin. 2018. "A general safety framework for learning-based control in uncertain robotic systems." *IEEE Transactions on Automatic Control*, 64, 7, 2737–2752.

S. Fujimoto, P. D'Oro, A. Zhang, Y. Tian, and M. Rabbat. 2025. "Towards General-Purpose Model-Free Reinforcement Learning." In: *International Conference on Learning Representations (ICLR)*. arXiv: 2501.16142.

J. Garcıa and F. Fernández. 2015. "A comprehensive survey on safe reinforcement learning." *Journal of Machine Learning Research*, 16, 1, 1437–1480.

S. Govinda, B. Brik, and S. Harous. 2025. "A Survey on Deep Reinforcement Learning Applications in Autonomous Systems: Applications, Open Challenges, and Future Directions." *IEEE Transactions on Intelligent Transportation Systems*, 26, 7, 11088–11113. doi:10.1109/TITS.2025.3560379.

L. Gracia, F. Garelli, and A. Sala. 2013. "Reactive sliding-mode algorithm for collision avoidance in robotic systems." *IEEE Transactions on Control Systems Technology*, 21, 6, 2391–2399.

J. Ji et al.. 2024. "OmniSafe: An Infrastructure for Accelerating Safe Reinforcement Learning Research." *Journal of Machine Learning Research*, 25, 285, 1–6. http://jmlr.org/papers/v25/23-0681.html.

O. Khatib. 1986. "Real-time obstacle avoidance for manipulators and mobile robots." In: *Autonomous robot vehicles*. Springer, 396–404.

J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli. 2015. "Kinematic and dynamic vehicle models for autonomous driving control design." In: *2015 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 1094–1099.

F. Li, A. Abuduweili, Y. Sun, R. Chen, W. Zhao, and C. Liu. 2025. "Continual Learning and Lifting of Koopman Dynamics for Linear Control of Legged Robots." In: *Learning for Dynamics and Control Conference (L4DC)*. arXiv: 2411.14321.

Z. Li, C. Hu, W. Zhao, and C. Liu. 2023. "Learning predictive safety filter via decomposition of robust invariant set." arXiv: 2311.06769.

C. Liu and M. Tomizuka. 2014. "Control in a safe set: Addressing safety in human-robot interactions." In: *ASME 2014 Dynamic Systems and Control Conference*. American Society of Mechanical Engineers Digital Collection.

M. Liu, S. Fang, H. Dong, and C. Xu. 2021. "Review of digital twin about concepts, technologies, and industrial applications." *Journal of Manufacturing Systems*, 58, 346–361.

E. Noorani, C. N. Mavridis, and J. S. Baras. 2025. "Risk-Sensitive Reinforcement Learning With Exponential Criteria." *IEEE Transactions on Cybernetics*, 55, 8, 3774–3787. doi:10.1109/TCYB.2025.3575240.

R. Pandya, T. Wei, and C. Liu. 2024. "Multimodal Safe Control for Human-Robot Interaction." In: *2024 American Control Conference (ACC)*, 2672–2678. doi:10.23919/ACC60939.2024.10644925.

M. Papini, M. Pirotta, and M. Restelli. 2022. "Smoothing policies and safe policy gradients." *Machine Learning*, 111, 11, 4081–4137. ISBN: 1573-0565. doi:10.1007/s10994-022-06232-6.

T.-H. Pham, G. De Magistris, and R. Tachibana. 2018. "Optlayer-practical constrained optimization for deep reinforcement learning in the real world." In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 6236–6243.

M. Pirotta, M. Restelli, A. Pecorino, and D. Calandriello. 2013. "Safe policy iteration." In: *International Conference on Machine Learning*. PMLR, 307–315.

A. Ray, J. Achiam, and D. Amodei. 2019. *Benchmarking safe exploration in deep reinforcement learning*. Tech. rep. arXiv: 1910.01708.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. 2017. *Proximal Policy Optimization Algorithms*. (2017). arXiv: 1707.06347.

Y. Sun, R. Chen, K. S. Yun, Y. Fang, S. Jung, F. Li, B. Li, W. Zhao, and C. Liu. 2025. "SPARK: Safe Protective and Assistive Robot Kit." In: *IFAC Symposium on Robotics*. arXiv: 2502.03132.

A. J. Taylor and A. D. Ames. 2020. "Adaptive safety with control barrier functions." In: *2020 American Control Conference (ACC)*. IEEE, 1399–1405.

E. Todorov, T. Erez, and Y. Tassa. 2012. "Mujoco: A physics engine for model-based control." In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 5026–5033.

H. Zhang, G. Solak, S. Hjorth, and A. Ajoudani. 2025. *Passivity-Centric Safe Reinforcement Learning for Contact-Rich Robotic Tasks*. (2025). arXiv: 2503.00287.

J. Zhang, M. Kang, X. Li, and G.-y. Liu. 2017. "Bio-inspired genetic algorithms with formalized crossover operators for robotic applications." *Frontiers in neurorobotics*, 11, 56.

W. Zhang, Z. Yang, J. Shen, M. Liu, Y. Huang, X. Zhang, R. Tang, and Z. Li. 2021. "Learning to Build High-Fidelity and Robust Environment Models." In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 104–121.

W. Zhao, R. Chen, Y. Sun, F. Li, T. Wei, and C. Liu. 2024. "State-wise Constrained Policy Optimization." *Transactions on Machine Learning Research*.

W. Zhao, R. Chen, Y. Sun, R. Liu, T. Wei, and C. Liu. 2024. "Guard: A safe reinforcement learning benchmark." *Transactions on Machine Learning Research*. arXiv: 2305.13681.

W. Zhao, T. He, R. Chen, T. Wei, and C. Liu. 2023. "State-wise safe reinforcement learning: a survey." In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, 6814–6822.

W. Zhao, T. He, and C. Liu. 2021. "Model-free safe control for zero-violation reinforcement learning." In: *5th Annual Conference on Robot Learning*.

W. Zhao, F. Li, Y. Sun, R. Chen, T. Wei, and C. Liu. 2024. "Absolute Policy Optimization: Enhancing Lower Probability Bound of Performance with High Confidence." In: *Forty-first International Conference on Machine Learning*. https://openreview.net/forum?id=Ss3h1ixJAU.

W. Zhao, F. Li, Y. Sun, Y. Wang, R. Chen, T. Wei, and C. Liu. 2024. *Absolute State-wise Constrained Policy Optimization: High-Probability State-wise Constraints Satisfaction*. (2024). arXiv: 2410.01212.

Received 25 September 2024; revised 15 August 2025; accepted 27 October 2025