Robotica (2025), **43**, pp. 1691–1707
doi:10.1017/S026357472500044X

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

Safety supervision framework for legged robots through safety verification and fall protection

Ming Sun[1] and Yue Gao[2,3]

[1]Department of Automation, Shanghai Jiao Tong University, Shanghai, China
[2]MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, Shanghai, China
[3]Shanghai Innovation Institute, Shanghai, China
Corresponding author: Yue Gao; Email: yuegao@sjtu.edu.cn

Received: 13 August 2024; **Revised:** 13 January 2025; **Accepted:** 1 March 2025; **First published online:** 10 April 2025

Keywords: legged robotics; robot safety; locomotion; robot learning; reinforcement learning

Abstract

Safety is an essential requirement as well as a major bottleneck for legged robots in the real world. Particularly for learning-based methods, their trial-and-error nature and unexplainable policy have raised widespread concerns. Existing methods usually treat this challenge as a trade-off between safety assurance and task performance. One reason for this drawback stems from the inaccurate inference for the robot's safety. In this paper, we re-examine the segmentation of the robot's state space in terms of safety. According to the current state and the prediction of the state transition trajectory, the states of legged robots are classified into *safe*, *recoverable*, *unsafe*, and *failure*, and a safety verification method is introduced to online infer the robot's safety. Then, task, recovery, and fall protection policies are trained to ensure the robot's safety in different states, forming a safety supervision framework independently from the learning algorithm. To validate the proposed method and framework, experiment results are conducted both in the simulation and on the real-world robot, indicating improvements in terms of safety and efficiency.

1. Introduction

Recent artificial intelligence theories and techniques have significantly advanced the performance of legged robots, enabling them to achieve locomotion capabilities nearly comparable to legged animals [1–6]. As legged robots walk from laboratory to real-world scenarios, many new challenges begin to emerge and receive more attention, and safety is one of them.

Safety is one of the most important issues for the practical application feasibility of legged robots, particularly in specialized, hazardous, and high-cost application scenarios. Many existing control-based and learning-based methods have considered safety in legged locomotion planning and control, including both in the designing and deployment stages [7–10]. However, for learning-based frameworks, due to their trial-and-error nature, considering the safety solely in the algorithm design usually remains unreliable [11]. This prompts us to propose a framework independently from the algorithm to further supervise and guarantee the safety of the legged robots, which is complementary to the interior safety of the control policy (e.g., reinforcement learning algorithm).

The motivation of the proposed framework in this work arises from the classification, criterion, and corresponding responses of the robot's running states. Typically, the states of the robots can be divided into safe and unsafe [12]. For instance, in learning-based methods, a common approach is to design an early termination criterion, such as the base of the robot contacts the ground or the body height / angular velocity/accelerated velocity exceeding a pre-determined threshold [13, 14]. When the robot reaches an unsafe state, it will be reset in the simulation. This can usually effectively improve the data efficiency during the training stage [15]. However, during the deployment stage, it will fail because there is no available safety measure when an unsafe state is recognized.

© The Author(s), 2025. Published by Cambridge University Press.

https://doi.org/10.1017/S026357472500044X Published online by Cambridge University Press

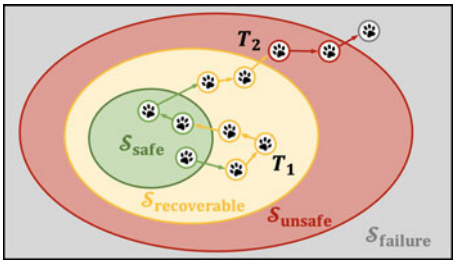


Figure 1. *Classification of state space in the proposed safe framework for legged robots. The conservative safe set $\mathcal{S}_{\text{safe}}$ and failure set $\mathcal{S}_{\text{failure}}$ are pre-defined, while the $\mathcal{S}_{\text{recoverable}}$ and $\mathcal{S}_{\text{unsafe}}$ are inferred online through safety verification.*

To enhance the safety of legged robots, we rethink the safety criteria and corresponding response measures. Motivated by prior works [10, 12, 16, 17], the proposed framework in this paper is constructed based on the policy switch structure: a task policy π_{task} to control the robot to perform desired tasks, and a recovery policy π_{recovery} to protect and recover the robot from unsafe states. However, the safety criteria in most current methods for legged robots are linear and human-designed rules, which are usually far from the realistic boundary. When the boundary is loose, the robot can easily reach an unsafe state and finally lead to a failure state; while if the boundary is too tight, the performance of the robot may be weakened, as the robot is limited to the conservative zones. Therefore, we attempt to redefine the safe and unsafe boundary for legged robots. In this paper, we propose to classify the states of legged robots into four categories: safe, recoverable, unsafe, and failure, as shown in Figure 1. First, the conservative safe set $\mathcal{S}_{\text{safe}}$ and failure set $\mathcal{S}_{\text{failure}}$ are pre-defined. Then for the left zones between $\mathcal{S}_{\text{safe}}$ and $\mathcal{S}_{\text{failure}}$, the safety verification criterion is used to infer the recoverability of the states. When the robot runs out of the conservative safe zones $\mathcal{S}_{\text{safe}}$, safety verification is used to infer whether the robot can be recovered back to $\mathcal{S}_{\text{safe}}$ through the recovery controller π_{recovery}. If the state is recoverable (i.e., in $\mathcal{S}_{\text{recoverable}}$), the task policy π_{task} can be continued; otherwise, the recovery policy π_{recovery} will take over until the robot returns to $\mathcal{S}_{\text{safe}}$. Theoretically, this structure can ensure the robot running inside the union set of $\mathcal{S}_{\text{safe}}$ and $\mathcal{S}_{\text{recoverable}}$ in every time step because once an unrecoverable state arises, the robot can return to $\mathcal{S}_{\text{safe}}$ along the recoverable trajectory of the previous time step to prevent failure. However, due to the existence of model inaccuracy and uncertain external disturbance, it is still possible to reach the unsafe (i.e., unrecoverable) set $\mathcal{S}_{\text{unsafe}}$ even with safety verification. Hence, on the basis of the two-policy structure, a fall protection policy π_{fall} is developed. When the robot reaches an unsafe state, π_{fall} allows the robot to fall in a safe manner to minimize mechanical damage while preventing strange configurations.

Briefly, the contributions of this work are as follows:

- A novel state space segmentation based on safety is proposed for legged robots. On this basis, a safety verification method is introduced to infer the safety of the robot, which can be used to determine the response mode of the robot.
- A safety supervision framework for legged robots is proposed, including safety verification, recovery policy, and fall protection. Different policies will be adopted according to the safety status of the robot.
- Experiment validations are conducted both in the simulation and on the real-world robot, which demonstrates the feasibility of the proposed framework and the performance improvement *w.r.t.* safety assurance, task performance, and data efficiency.

The remainder of this paper is organized as follows: Section 2 summarizes existing related works. Section 3 describes the problem formulation. Section 4 introduces the proposed method thoroughly. Section 5 presents the experimental setting, results, and analysis. Section 6 concludes the work and discusses future works.

https://doi.org/10.1017/S026357472500044X Published online by Cambridge University Press

2. Related works

2.1. Safe locomotion for legged robots

Many existing locomotion control methods have considered safety in the motion planning and control of legged robots. For control-based method, safety constraints can be introduced into the problem formulation and optimization. Based on the control barrier function (CBF) theory, Hsu et al. introduced physical constraints such as joint limits as barrier functions into the controller and achieved stable walking under the premise of ensuring physical feasibility [18]. Nguyen et al. combined control Lyapunov functions to achieve periodic walking and CBFs to enforce strict constraints on step length and step width [19].

Learning-based safe locomotion methods can be divided into two types: safe reinforcement learning and recovery policy. Safe reinforcement learning (Safe RL) is a widely studied subfield within RL, with growing safety concerns when deploying RL algorithms in real-world tasks, such as robotics and autonomous driving [20–23]. Generally, the main idea of safe RL includes transforming the optimization criterion and modifying the exploration process [20]. For the applications in robotics, the main challenges lie in designing accurate model formulation, exploration constraints, and policy optimization for safe RL algorithms [24]. In ref. [25], a model-based RL method is proposed for learning safe policies while minimizing unrecoverable errors during the training process. The agent can avoid dead-ends through safe exploration and safe model learning. Li et al. introduce the concept of constraint-admissible set, which serves as a supervision component to prevent unsafe exploration by projecting the RL actions onto a safe set of actions [26].

These safe RL methods can effectively improve training efficiency while reducing the safe constraints violation ratio in the training process. However, it still remains challenging to strictly guarantee the safety of the algorithm due to the trial-and-error learning paradigm and the sub-optimality of the policy, especially during the initial training stage and sim-to-real deployment. Therefore, some works introduce recovery policy to prevent unsafe actions and unrecoverable failure [10, 12, 17]. Specifically, a safety criterion is defined independent of the algorithm. When the robot reaches the boundary of the safe set, the actions of the RL agent will be blocked and a recovery policy will take over to stabilize the robot back to the safe state. The main components of these frameworks are the switching criterion and the recovery policy, most of which can be integrated with various RL algorithms. In ref. [12], Recovery RL is proposed, where the agent learns the safety zones from offline data and completes the goals of task and safety through a task policy and a recovery policy. Yang et al. introduce a handing control back criterion for recovery policy, where the recovery policy hands over the control back only when the task policy does not violate safety constraints within a few steps [10].

These intuitive and effective methods can protect and recover the robot to the safe set when the safety constraints are violated. However, current methods employ simple switching criteria, typically handcrafted or estimated by a neural network, as summarized in Table I. Due to the high dimensionality and non-linearity of the legged robotic system as well as the constantly dynamic environment, accurately describing the safety boundary remains challenging. Therefore, a more practical and meaningful approach, safety verification, is proposed in this paper to infer the safety between the deterministic safe and failure set of the robot. Furthermore, because of the uncertain disturbance and diverse environment, an emergency response policy is still necessary for the legged robots in real-world applications. Therefore, a fall protection policy is employed to minimize mechanical damage while preventing strange configurations. With the integration of the above components, a safety supervision framework is constructed, which provides a solution for enhancing robot safety independently of the algorithm.

2.2. Fall protection for legged robots

To minimize the damage during the falling process, fall protection methods attempt to protect the robot through hardware and algorithm. To reduce the impact damage to the robot, impact-absorbing material and shell protector can be installed on the robot for protection [27, 28]. In ref. [29], Ma et al. proposed to use the manipulators on the legged robot to reduce fall damage and assist the recovery.

https://doi.org/10.1017/S026357472500044X Published online by Cambridge University Press

Table I. *Recovery policy-based safe locomotion methods.*

Methods	Policy	Switching criterion
ref. [12]	$\pi_{\text{task}}, \pi_{\text{recovery}}$	Safety critic network
ref. [10]	$\pi_{\text{task}}, \pi_{\text{recovery}}$	Human-designed rules
ref. [17]	$\pi_{\text{task}}, \pi_{\text{recovery}}$	Reach-avoid value network
Ours	$\pi_{\text{task}}, \pi_{\text{recovery}}, \pi_{\text{fall}}$	Safety verification

However, relying solely on hardware to ensure the safety of a robot is not reliable enough. Ding et al. designed a fall protection method for humanoid robots according to the human fall motion [30]. Wang et al. proposed an active safe falling method to tumble and recover to stable modes to reduce damage in highly dynamic scenarios [31].

In this work, we combine hardware and algorithm to protect the robot from mechanical damage and facilitate easy-to-recover configuration. Furthermore, the fall protection policy is integrated into the safety supervision framework as part of the robot system.

3. Preliminaries and definitions

3.1. Problem statement

During the operation of the robot, task completion and safety need to be considered at the same time, that is, to complete the task under the premise of ensuring safety. In the setting of classical reinforcement learning, the goal of the policy is to maximize the task reward, for example, for legged robots, the task can be to follow the velocity command. However, safety is a crucial issue for legged robots. The robot needs to avoid violating safety constraints while completing the task, such as falling or torque limit. Therefore, in this work, we consider the problem of safe learning and operation of legged robots.

Generally, the above problem considered in this paper can be formulated as a constrained Markov Decision Process [32], which can be described as a tuple $M =< \mathcal{S}, \mathcal{A}, T, R, C >$. Here $\mathcal{S}$ and $\mathcal{A}$ denote the state space and the action space respectively, while T and R represent the state transition function and the reward function. C is the constraint cost function, defining the constraints and the violation penalty. The objective is to maximize the discounted cumulative reward while avoiding constraint violation.

To achieve the goal of safe locomotion, we segment the state space of the robot into four zones. Firstly, the conservative zone $\mathcal{S}_{\text{safe}}$ is the set of the deterministic safe states. It is assumed that for the state $\mathbf{s}_t \in \mathcal{S}_{\text{safe}}$, the task policy π_{task} can guarantee the safety of the robot while completing the task, without the intervention of other modules. Accordingly, the failure set $\mathcal{S}_{\text{failure}}$ is defined as the failure states where the constraints are violated. Between $\mathcal{S}_{\text{safe}}$ and $\mathcal{S}_{\text{failure}}$, the safety criterion is difficult to precisely pre-defined. To avoid too conservative or aggressive policy, we further segment this zone into $\mathcal{S}_{\text{recoverable}}$ and $\mathcal{S}_{\text{unsafe}}$. Within $\mathcal{S}_{\text{recoverable}}$, the recovery policy π_{recovery} can save the robot back to the safe set to avoid failure. While for $\mathcal{S}_{\text{unsafe}}$, π_{recovery} will not work and eventually leads to a failure.

To complete the task while ensuring the safety of the robot, the objectives of the safety supervision framework include: (1) To maximize the task reward while keeping the robot within the safe set $\mathcal{S}_{\text{safe}}$. (2) When the robot reaches outside of $\mathcal{S}_{\text{safe}}$, to constantly supervise the state of the robot and when necessary save it back to the safe state before it reaches the unsafe set. (3) When the robot reaches the unsafe set $\mathcal{S}_{\text{unsafe}}$, to fall with minimal mechanical damage.

3.2. Reinforcement learning problem formulation

In this paper, the task of legged locomotion is formulated as follows:

State Space.

The state $\mathbf{s}_t \in \mathcal{S}$ is a vector describing the state of the robot at time t, denoted as

https://doi.org/10.1017/S026357472500044X Published online by Cambridge University Press

$$\mathbf{s}_t = [\mathbf{v}_t,\ \boldsymbol{\omega}_t,\ \mathbf{g}_t,\ \mathbf{c}_t,\ \mathbf{q}_t,\ \dot{\mathbf{q}}_t,\ \mathbf{a}_{t-1}]^T \tag{1}$$

where $\mathbf{v}_t$ and $\boldsymbol{\omega}_t$ represent the linear and angular velocity of the robot in the body frame, which are estimated through the IMU data. $\mathbf{g}_t$ is the gravity vector, $\mathbf{c}_t$ is the command of the desired body velocity, $\mathbf{q}_t$ and $\dot{\mathbf{q}}_t$ are joint angle position and joint angular velocity, which are measured and computed through the encoder. $\mathbf{a}_{t-1}$ is the previous action.

Action Space.

The action $\mathbf{a}_t$ consists of the desired angle position of each joint relative to the default pose of the robot $\mathbf{q}_{\text{default}}$, that is

$$\mathbf{q}_{\text{des},t} = \mathbf{q}_{\text{default}} + \mathbf{a}_t \tag{2}$$

where $\mathbf{q}_{\text{des},t}$ is the desired joint angle position tracked using a PD controller.

Reward Function.

In this paper, the task of the robot is defined as tracking the desired velocity command. Since the main contribution of this paper does not lie in reward shaping or tuning, the form of reward function closely follows existing state-of-the-art works [1–3] to highlight the effect of the safety supervision framework. The total reward of the task policy is presented as

$$r_t(\mathbf{s}_t, \mathbf{a}_t) = \sum_{i=1}^{N} w_i r_{t,i}(\mathbf{s}_t, \mathbf{a}_t) \tag{3}$$

where $r_{t,i}$ is the reward term component, and w_i is the corresponding weight.

The objective is to train a task policy for legged locomotion control. The parameters of the policy are optimized using the proximal policy optimization (PPO) algorithm [33] to maximize the cumulative discounted reward

$$J(\theta) = \sum_{t=0}^{T-1} \mathbb{E}_{\mathbf{a}_t \sim \pi_{\text{task}}}[\gamma^t r(\mathbf{s}_t, \mathbf{a}_t)] \tag{4}$$

where γ is the discount factor, θ is the parameter of the policy.

4. Methods

In this section, we present the details of each component of the proposed method and then outline the safety supervision framework for legged robot locomotion.

4.1. State space segmentation

In prior works, safety is usually defined as a clear boundary of the state space. However, due to the high dimensionality, non-linearity, and dynamics of the legged robot, precisely defining the safety boundary is challenging. Therefore, in this work, the state space of the robot is segmented into four zones according to the safety, and safety verification is utilized to infer the safety of the robot and activate the corresponding policies.

First, we define the safe set

$$\mathcal{S}_{\text{safe}} = \{\mathbf{s}_t \mid h_z > h_{\text{T}},\ \boldsymbol{\omega}_{x,y} < \omega_{\text{T}}\} \tag{5}$$

where h_z is the height of robot base, $\boldsymbol{\omega}_{x,y}$ is the angular velocity along x and y axis, h_{T} and ω_{T} are pre-defined threshold. During the training process, the state is obtained from the simulator; while on the real robot, it is estimated through the state estimator. For the safe set, it is assumed that the thresholds are defined conservatively so that the task policy can guarantee the safety of the robot when completing the task.

Then, the failure set of the robot is presented as

$$\mathcal{S}_{\text{failure}} = \{\mathbf{s}_t \mid f_z > 0\} \tag{6}$$

https://doi.org/10.1017/S026357472500044X Published online by Cambridge University Press

where f_z is the contact force of the robot's base. To infer the safety between $\mathbf{s}_t \in \mathcal{S}_{\text{safe}}$ and $\mathbf{s}_t \in \mathcal{S}_{\text{failure}}$, we introduce the recoverability of the robot's state. The motivation is to verify that whether a state $\mathbf{s}_t \in \{\mathcal{S} \backslash \mathcal{S}_{\text{safe}} \backslash \mathcal{S}_{\text{failure}}\}$ can be recovered back to $\mathcal{S}_{\text{safe}}$ using the recovery policy π_{recovery} in a certain time steps. If yes, we call that the state $\mathbf{s}_t$ is recoverable; otherwise, $\mathbf{s}_t$ is unrecoverable. Hence, the recoverable criterion can be presented as

$$\exists\ \{\mathbf{a}_{t+i}\}_{i=0}^{n-1} \in \pi_{\text{recovery}} \tag{7a}$$

$$\text{s.t.} \quad \mathbf{s}_{t+n} \in \mathcal{S}_{\text{safe}} \tag{7b}$$

$$\mathbf{s}_{t+i+1} = T'(\mathbf{s}_{t+i}, \mathbf{a}_{t+i}) \tag{7c}$$

where T' is a prediction model of the transition function trained using offline data and n is the step length of prediction. Then the recoverable set is defined as

$$\mathcal{S}_{\text{recoverable}} = \{\mathbf{s}_t \,|\mathbf{s}_t \vDash (\text{Eq. } 7)\} \tag{8}$$

From the perspective of safety, the recoverable set is allowed to be reached because the recovery policy can stabilize the robot back to the safe state. That is, in this paper, the boundary of the recovery set, which is verified through Eq. (7), determines the robot safety, rather than the boundary of human-defined conservative safe set. Finally, the area other than $\mathcal{S}_{\text{safe}}$, $\mathcal{S}_{\text{failure}}$ and $\mathcal{S}_{\text{recoverable}}$ is defined as the unsafe set

$$\begin{aligned}\mathcal{S}_{\text{unsafe}} &= \mathcal{S} \backslash \mathcal{S}_{\text{safe}} \backslash \mathcal{S}_{\text{recoverable}} \backslash \mathcal{S}_{\text{failure}} \\ &= \{\mathbf{s}_t \,|(\mathbf{s}_t \nvDash (\text{Eq. } 7)) \cap (\mathbf{s}_t \notin \mathcal{S}_{\text{failure}})\}\end{aligned} \tag{9}$$

In the unsafe set, the robot is unrecoverable but has not yet reached the failure set. This set is usually reached during the robot's falling before reaching a failure state.

4.2. Safety verification

From the definition of the robot's state in Section 4.1, the robot is expected to work within

$$\mathcal{S}_{\text{reachable}} = \mathcal{S}_{\text{safe}} \cup \mathcal{S}_{\text{recoverable}} \tag{10}$$

where an example of the state trajectory is shown as T_1 in Figure 1. For $\mathbf{s}_t \in \mathcal{S}_{\text{safe}}$, the task policy π_{task} is applied with the assumption that the safety can be guaranteed. For $\mathbf{s}_t \in \mathcal{S}_{\text{recoverable}}$, task policy can still work, and the recovery policy π_{recovery} only takes over the control only when the robot is to reach an unsafe state from a recoverable state. This approach can balance the performance and safety of the robot.

However, if the policy activation is only determined through current state $\mathbf{s}_t$, it is unable to infer the above condition, and the safety of the robot is still not guaranteed. Although the robot can be recovered within $\mathcal{S}_{\text{reachable}}$, the transition from $\mathcal{S}_{\text{reachable}}$ to $\mathcal{S}_{\text{unsafe}}$ may occur under the control of task policy, and the recovery policy will no longer work. To this end, a one-step prediction is required through the prediction model to infer the recoverability of $\mathbf{s}_{t+1}$, as well as determine the corresponding policy. For the recoverable state $\mathbf{s}_t$, the task policy can be applied if the next state under task policy $\mathbf{s}_{t+1}$ is still recoverable for the consideration of task performance. Otherwise, the recovery policy should take over the control to prevent the robot from reaching an unsafe state in advance.

Nevertheless, in real-world scenarios, even with safety verification, the robot may fail to recover to a safe state due to external disturbance or model inaccuracy. In this case, the state trajectory of the robot may eventually lead to a failure state, as T_2 in Figure 1. Therefore, constant safety verification is necessary along the recovery trajectory. If the robot reaches the unsafe set, the robot should fall in a safe manner under the fall protection policy π_{fall} to prevent mechanical faults and unrecoverable configurations, such as spreading the legs to increase the contact area with the ground and avoid rollover or tip-over.

According to the proposed state space segmentation in Section 4.1 and the safety verification method, the state segmentation criteria and policy activation of the robot can be presented in Figure 2. The above process is to infer the safety of the robot and determine the control authority of the policies, which is called safety verification.

https://doi.org/10.1017/S026357472500044X Published online by Cambridge University Press

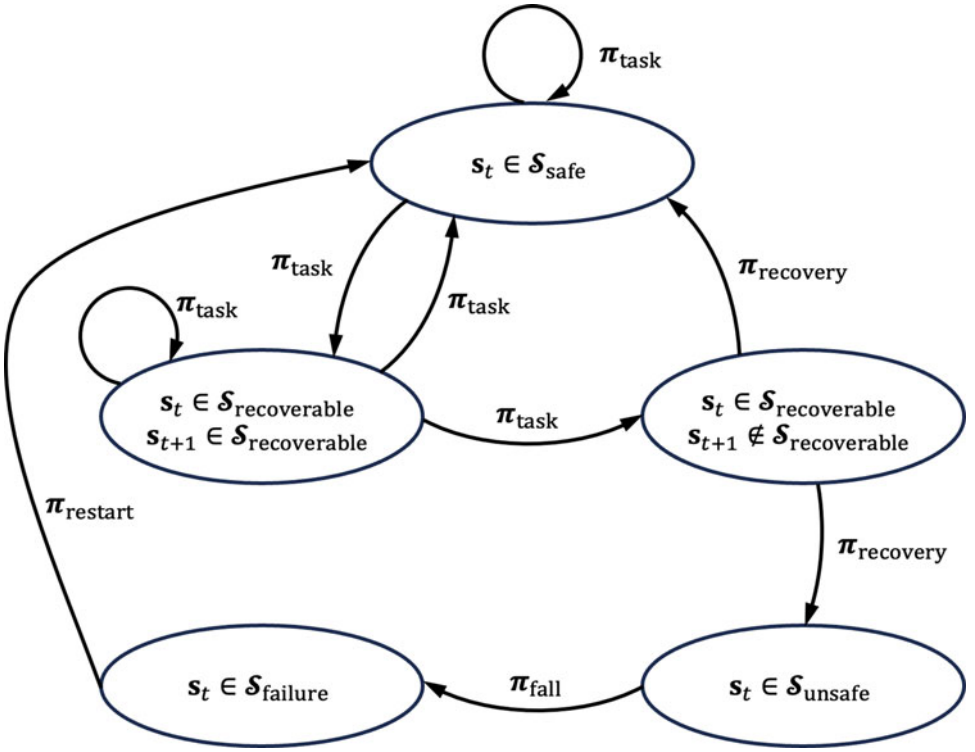


Figure 2. *State segmentation criteria and policy activation of the robot through safety verification.*

4.3. Safety supervision framework

To construct the safety supervision framework, three policies are trained and activated in different safety states.

The task policy is optimized to maximize the task reward and auxiliary reward independently with other components of the framework, where the reward function is defined in Section 3.2 and Table II. Since the proposed framework is independent of the specific implementation of the task policy, various RL algorithms or control-based methods can be employed. In this paper, we utilize the widely used PPO algorithm [33] to optimize the policy parameters.

The objective of the recovery policy is to recover the robot to the default stable posture. Its reward function consists of the recovery reward for encouraging being close to the default posture and the auxiliary reward, as shown in Table II. The recovery policy is pre-trained with random initialization of robot configurations between $\mathcal{S}_{\text{safe}}$ and $\mathcal{S}_{\text{failure}}$.

The fall protection policy is to minimize mechanical damage and facilitate easy-to-recover configuration during the robot's falling process. For most current frameworks, the falling process is uncontrolled and thus prone to causing damage. A reasonable alternative is to design a buffer device, such as a sponge or bumper, to absorb the impact force when the robot falls. In this paper, we install a sponge cushion underneath the base of the robot, as shown in Figure 3(a). The objective of the fall protection policy is to make the robot contact the ground with the bottom of the robot base, in order to absorb the impact using the sponge cushion, as illustrated in Figure 3(b). On the other hand, this posture also facilitates the robot's reboot from a failure state. The reward function of the fall protection policy is shown in Table II, encouraging the desired falling posture and light impact on the base of the robot.

The safety supervision framework for legged robots is illustrated in Figure 4 and Algorithm 1. Through safety verification, the state of the robot can be inferred online, activating the corresponding policy to complete tasks while guaranteeing safety. In most cases, safety verification can ensure that

https://doi.org/10.1017/S026357472500044X Published online by Cambridge University Press

Table II. *Reward terms and weights for the task policy.*

Term	**Equation** ($r_i(\mathbf{s}_t, \mathbf{a}_t)$)	**Weight** (w_i)
Task reward		
Lin. velocity tracking	$\exp\{-(\mathbf{v}_{xy}^{\text{cmd}} - \mathbf{v}_{xy}^{\text{base}})^2/\sigma_{vxy}\}$	1.0
Ang. velocity tracking	$\exp\{-(\boldsymbol{\omega}_z^{\text{cmd}} - \boldsymbol{\omega}_z^{\text{base}})^2/\sigma_{\omega z}\}$	0.5
Recovery reward		
Default posture	$\sum_{\text{joint}} (\mathbf{q}_{i,\text{default}} - \mathbf{q}_i)^2$	1.0×10^{-5}
Foot contact ground	$\sum_{\text{foot}} \mathbb{1}_{i,\text{contact}}$	0.02
Fall protection reward		
Desired base height	$(h_z^{\text{des}} - h_z^{\text{base}})^2$	−5.0
Base impact	$(\mathbf{f}_{\text{base, ground}})^2$	−0.001
Tip-over	$\mathbb{1}_{\alpha\|\beta>\pi/2}$	−100.0
Auxiliary reward		
Linear velocity (z)	$(v_z^{\text{base}})^2$	−2.0
Angular velocity (x, y)	$(\boldsymbol{\omega}_{xy}^{\text{base}})^2$	−0.05
Desired base height	$(h_z^{\text{des}} - h_z^{\text{base}})^2$	−0.4
Joint torques	$\boldsymbol{\tau}^2$	−0.0002
Joint acceleration	$\ddot{\mathbf{q}}^2$	-2.5×10^{-7}
Swing time of legs	$\sum_{\text{foot}} T_{\text{swing}}$	1.0
Collision	$\mathbb{1}_{\text{collision}}$	−1.0
Action rate	$(\mathbf{a}_{t-1} - \mathbf{a}_t)^2$	−0.01
Action smoothness	$(\mathbf{a}_{t-2} - 2\mathbf{a}_{t-1} + \mathbf{a}_t)^2$	−0.01



Figure 3. *To absorb the impact force during the falling process, a sponge cushion is installed underneath the base of the robot. The fall protection policy is to minimize mechanical damage using the cushion.*

the recovery policy prevents the robot from reaching unsafe states. Even in the case that unexpected disturbances lead to an inevitable failure, the fall protection policy can still minimize the damage by mitigating the impact through the sponge cushion, as well as allowing the robot to reboot and resume the tasks. Moreover, in order to avoid action mutation, a low-pass filter is applied to the action signal during policy switching.

https://doi.org/10.1017/S026357472500044X Published online by Cambridge University Press

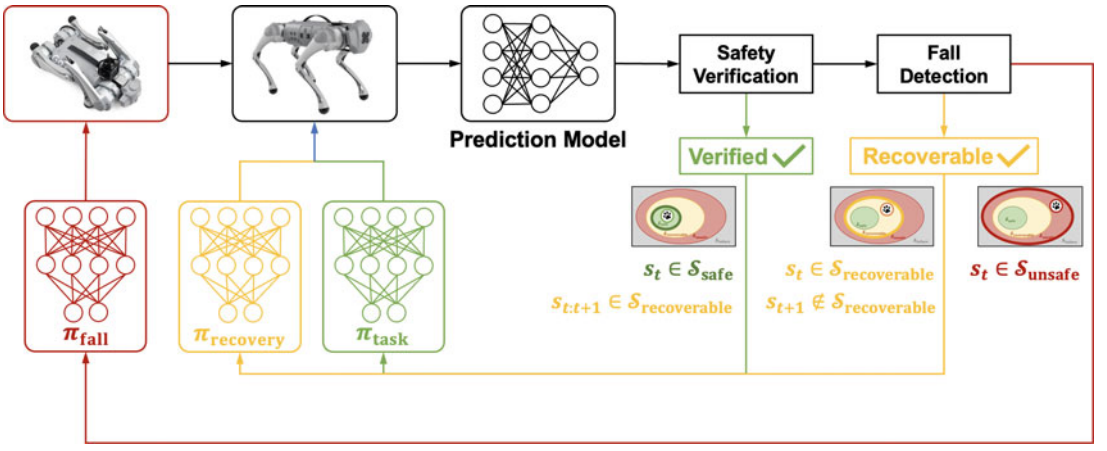


Figure 4. *The framework of the proposed safety supervision for legged robots. In each time step, the safety of the robot is inferred according to the current state and the prediction of state transition trajectory, and the corresponding policy is activated to complete the task while ensuring safety.*

Algorithm 1: Safety Supervision Framework

Input: State space $\mathcal{S}$, safe/failure set $\mathcal{S}_{\text{safe}}, \mathcal{S}_{\text{failure}}$;
Input: task policy π_{task}, recovery policy π_{recovery}, fall policy π_{fall};
Input: Prediction model T', prediction step length n;
for $t = 0, 1, \ldots, N$ **do**
  Get current state of the robot $\mathbf{s}_t$;
  switch $\mathbf{s}_t$ **do**
    case $\mathbf{s}_t \in \mathcal{S}_{\text{safe}}$ **do**
      $\mathbf{a}_t = \pi_{\text{task}}(\mathbf{s}_t)$;
    end
    case $\mathbf{s}_t \in \mathcal{S}_{\text{recoverable}}$ **do**
      Predict $s'_{t+1} = T'(\mathbf{s}_t, \pi_{\text{task}}(\mathbf{s}_t))$;
      if $s'_{t+1} \in \mathcal{S}_{\text{recoverable}}$ **then**
        $\mathbf{a}_t = \pi_{\text{task}}(\mathbf{s}_t)$;
      else
        $\mathbf{a}_t = \pi_{\text{recovery}}(\mathbf{s}_t)$;
      end
    end
    case $\mathbf{s}_t \in \mathcal{S}_{\text{unsafe}}$ **do**
      $\mathbf{a}_t = \pi_{\text{fall}}(\mathbf{s}_t)$;
    end
    case $\mathbf{s}_t \in \mathcal{S}_{\text{failure}}$ **do**
      Reboot the robot;
    end
  end
  Send action $\mathbf{a}_t$ to the PD joint controller;
end

https://doi.org/10.1017/S026357472500044X Published online by Cambridge University Press

Table III. *Domain randomization terms and ranges during policy training for sim-to-real transfer.*

Term	Min	Max	Unit
Ground friction	0.3	1.25	–
Ground restitution	0.0	1.0	–
Gravity offset	−1.0	1.0	m/s^2
Motor strength	0.8	1.2	×torque
Push base velocity (x,y)	−1.0	1.0	m/s
Base mass offset	−1.0	2.0	kg
Base CoM offset	−0.02	0.02	m
Initial joint position	0.5	1.5	×default value

5. Experiments

In this section, we report the experimental results of the proposed safety supervision framework *w.r.t.* data efficiency, safety, and task performance, with the comparison of baseline methods. The experiments are conducted both in the simulation and on a quadrupedal robot Unitree Go1 in the real world.

5.1. Experiment setup

Robot and Platform.

The experiments are conducted on a quadrupedal robot Unitree Go1 [34], which weighs 13 kg, stands at 0.28 m, and has 12 actuator joints. For the simulation experiments, the Isaac Gym simulator platform [35] is used, which allows us to train 4,096 domain-randomized agents in parallel. The training is performed on a PC with an Intel Core i7-11700 CPU and an NVIDIA RTX 3090 GPU. For real-world deployment, a robot Go1 equipped with an onboard computing platform Jetson Xavier NX is used. Policy inference, state estimation, and communication with the controller board through SDK run on the onboard computer with 50 Hz.

Network Architecture.

Both the actor and the critic network are the MLP with hidden layer sizes [512, 256, 128] and exponential linear unit activations.

Pre-Training.

Before training the task policy, the prediction model, recovery policy, and fall protection policy are pre-trained separately. During the task policy training process, the recovery policy and fall protection policy are fixed, and the prediction model is updated using the online collected data to adapt to new state transition pairs.

Sim-to-Real Transfer.

For better sim-to-real transfer, the policy is trained with domain randomization [36] *w.r.t.* robot's base mass, the center of mass, motor strength, friction coefficient, gravity, observation noise, and random external force. The specific settings for domain randomization are listed in Table III.

Hyperparameter.

For the safety verification, the main hyperparameter is the step length of prediction n in Eq. (7). Since safety verification requires inferring safety of the robot through the rollout of the predicted robot state trajectory, the increased computational cost is non-negligible compared to the baseline method without safety verification. Nevertheless, the inference of policy network and prediction network is usually fast. In the experiment, the average computational time for a single inference of policy network and prediction network s 0.00019 and 0.00016 s, respectively, which is acceptable for onboard computation. Considering the effectiveness and feasibility of method, in the experiments, we set $n = 3$ as a reasonable trade-off between performance and computational complexity.

https://doi.org/10.1017/S026357472500044X Published online by Cambridge University Press

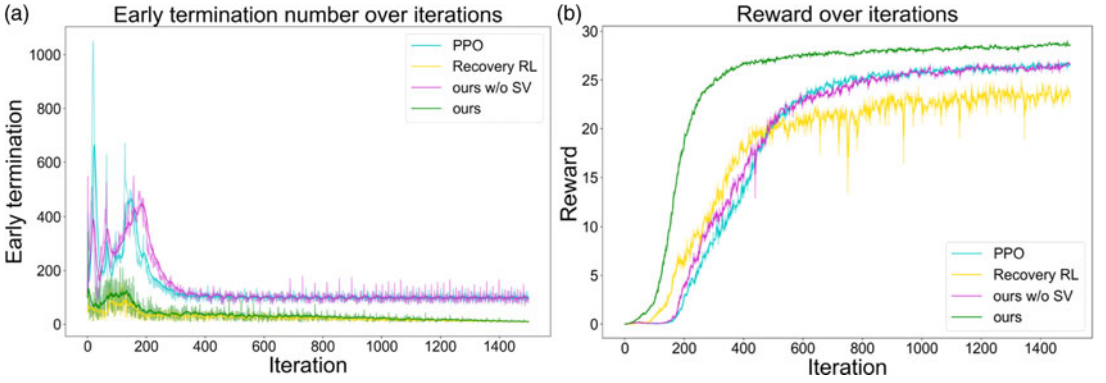


Figure 5. *Experiment results of baselines and the proposed method in simulation training. (a) Early termination number and (b) Reward over training iterations.*

Baselines.

We consider the following settings for comparative evaluations:

1. PPO [1]: The state-of-the-art RL baseline for legged robots using PPO without the consideration of safety.
2. Recovery RL [12]: The policy switch method with a recovery policy.
3. Ours w/o SV: Our proposed method without safety verification.
4. Ours: Our proposed method.

5.2. Simulation experiments

Early Termination and Data Efficiency.

For all experiment settings, the contact of the robot's base with the ground is considered a failure and leads to an early termination of the current episode. The number of early termination and the reward function *w.r.t.* the training iteration are shown in Figure 5(a) and (b). The experiment results demonstrate the trade-off between robot safety and task performance during training. *PPO* does not require any model, safety definition, or recovery policy. However, due to the lack of consideration for safety and protection, it has more early termination, especially in the early stage of training. Therefore, this could slow down the training process on account of low data efficiency. *Recovery RL* is the most conservative agent among the experiment settings, where its safety boundary for recovery policy is set to $\mathcal{S}_{\text{safe}}$ in Eq. (5). Therefore, the number of early termination is significantly reduced through the frequent intervention of recovery policy, thereby improving the training efficiency. However, this conservative safety protection also prevents the agent from fully exploring and utilizing the robot's capability, leading to the conservative task policy. Consequently, the reward of Recovery RL has finally converged to a relatively lower value. An intuitive approach for improvement is to adjust the safety boundary to facilitate the exploration of the state space, but the safety of the robot will be accordingly weakened. Since hand-crafted rules and parameters cannot accurately describe the robot's safe zone, this approach always faces the trade-off between safety and performance. The ablation setting of our framework without safety verification is approximately equivalent to PPO with fall protection, exhibiting similar performance as PPO. Our proposed method can infer robot safety online through safety verification, enabling the robot to leave the conservative safety zone while ensuring safety. Hence the number of early termination during training is only slightly higher than the conservative Recovery RL, without weakening the converged task reward. These experiment results demonstrate that our proposed framework can reach a better trade-off between safety and task performance while avoiding the challenges of defining and adjusting the safety boundary and rules.

https://doi.org/10.1017/S026357472500044X Published online by Cambridge University Press

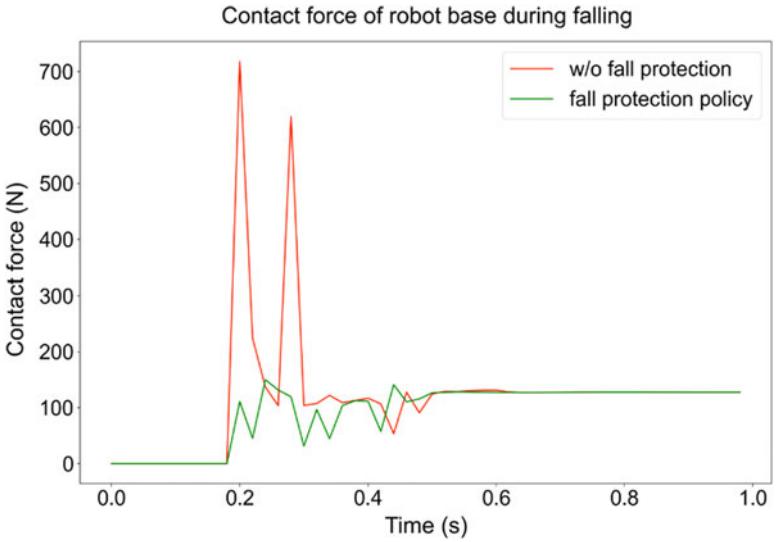


Figure 6. *Impact force during the robot falling process with and without the fall protection policy.*

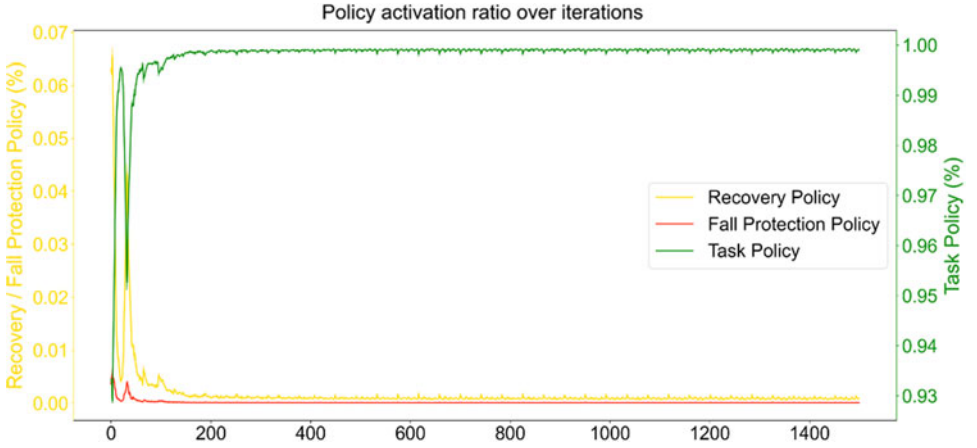


Figure 7. *Activation ratio of task, recovery, and fall protection policies during training.*

Impact Force of Falling.

Figure 6 shows the impact force during the falling process with and without fall protection policy when the same external force is applied to the robot. The results indicate that the fall protection policy can significantly reduce the impact of the robot during the falling process. At the same time, it promotes the robot to contact the ground with the bottom of the robot base to avoid tip-over. On the real-world robot, this falling posture can further reduce the impact through the sponge cushion underneath the robot and also makes it easier to recover to the default posture when rebooting.

Policy Activation.

The trend of the activation ratio of the three policies over iterations during training is shown in Figure 7. In the early stage of training, frequent activation of the recovery policy is observed, while with the training process of the task policy, the intervention of the recovery policy gradually decreases, which is intuitively reasonable. In the later stage of training, the recovery policy is usually activated only due to external disturbances. In comparison, the activation ratio of the fall protection policy remains low during training. On the one hand, the safety verification and the recovery policy can handle most of the unstable states in simulation, where the robot is constrained within the safe and recoverable zones. Only

https://doi.org/10.1017/S026357472500044X Published online by Cambridge University Press

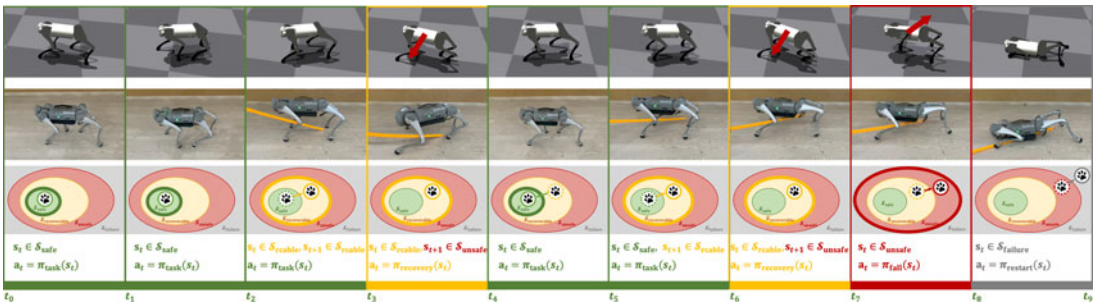


Figure 8. *Snapshots of the experiment result in the simulator and on the real robot. The transition of the robot's safety state and activated policy are shown, where green lines and words indicate the task policy, yellow indicates the recovery policy and red indicates the fall protection policy.*

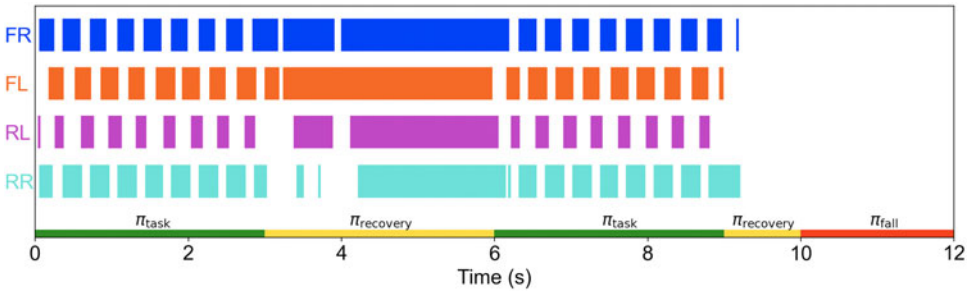


Figure 9. *Gait sequence of the robot in the experiment on flat terrain. The slight and strong external disturbances are applied to the robot at $t = 3$ and $t = 6$ respectively. The x-axis indicates the activated policies.*

when there is a large prediction model error or random push will the fall protection policy be activated. On the other hand, once the fall protection policy is activated, the current episode will be terminated due to the robot's base contacting the ground. Therefore, the fall protection policy is less activated during the training process in simulation.

5.3. Real-world experiments

To verify the effectiveness of the proposed method on the real robot, external disturbances are imposed on the robot to demonstrate the response to different states and the activation of the policies. The experimental results are shown in Figure 8, where the first and second columns show the snapshots in sequence, respectively, in the simulation and the real world. The third column illustrates the state inference and the activated policy at the current time step, where the green lines and words indicate the task policy, yellow indicates the recovery policy and red indicates the fall protection policy. In this experiment, the task policy is first activated to follow the desired velocity command. During the period $t_2 \sim t_4$ in Figure 8, the external disturbance was imposed on the robot, where the left front leg is obstructed by a stick. As a result, the robot cannot continue to perform the task, and the state of the robot reaches the recoverable zone, activating the recovery policy to maintain stability. Once the external disturbance was removed, the robot returned to a safe state and resumed the task. Then at time step $t_5 \sim t_9$, the robot was hindered by a stronger external disturbance, whose leg and base were disturbed by the stick. In response, the recovery policy is first activated to attempt to recover the robot to a safe state. However, the continuous external force eventually led the robot to fail to recover and reach the unsafe zone. Subsequently, the fall protection policy is activated to control the robot to fall vertically in the desired posture and utilize the

https://doi.org/10.1017/S026357472500044X Published online by Cambridge University Press

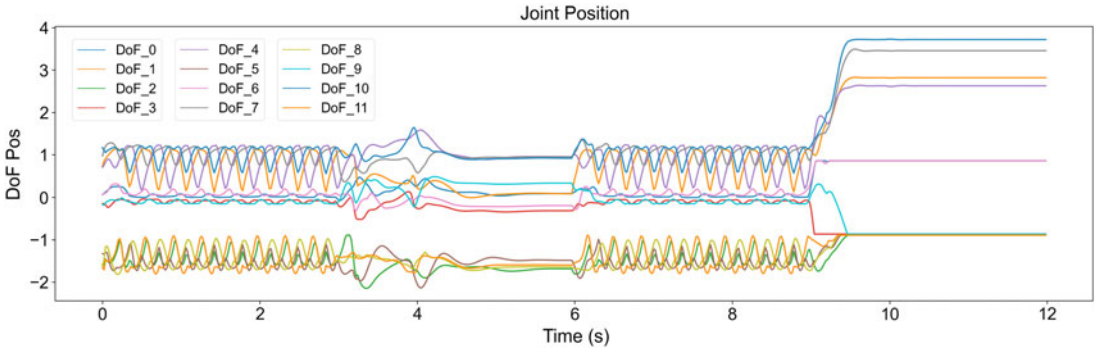


Figure 10. *Joint position measurements of the robot in simulation for the experiment on flat terrain.*

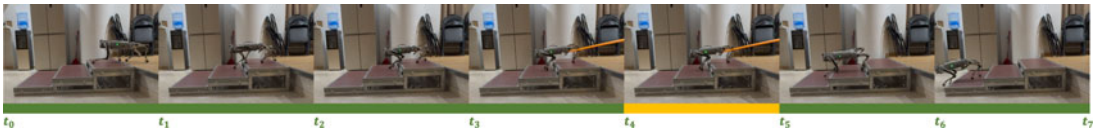


Figure 11. *Snapshots of the real-world experiment on stairs. The transition of the robot's activated policy is shown, where green lines and words indicate the task policy, yellow indicates the recovery policy, and red indicates the fall protection policy.*

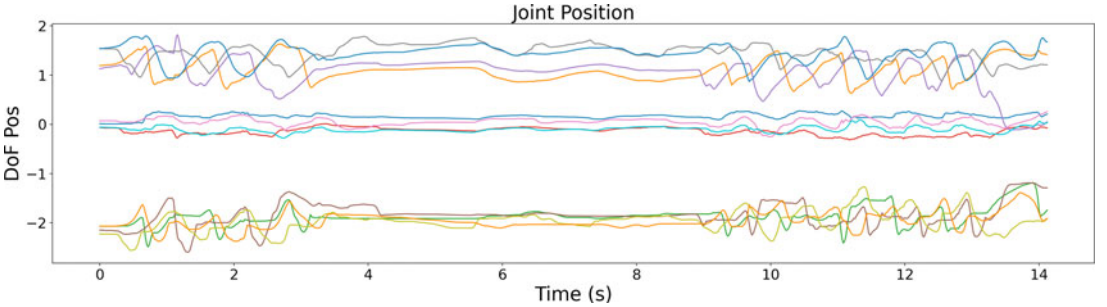


Figure 12. *Joint position measurements of the robot in the real world for the experiment on stairs. The legend is the same as Figure 10.*

sponge cushion to mitigate the impact. After falling with the bottom of the robot base contacting the ground, the robot can easily and safely restart and recover to a safe state.

This experiment demonstrates that the proposed method can effectively infer the safety of the robot and activate the corresponding policies. The gait sequence and joint position of the robot during the experiment are shown in Figures 9 and 10, where the activation of different policies is indicated on the x-axis.

To demonstrate that the proposed framework can generalize to complex terrains, the experiment is conducted on stairs and the results are shown in Figures 11 and 12. In this experiment, the robot followed the forward command to walk down the stairs. During $t_3 \sim t_4$, slight external disturbance was applied to the robot leg but did not affect task execution. So the task policy is still performed. During $t_4 \sim t_5$, the shaking of the robot base caused by the external disturbance intensified, and continuing to walk forward may cause the robot's fall. Therefore, the recovery policy took over the control and the robot suspended the task to maintain stability. Then after t_5, the disturbance was removed and the task policy regained the control and finally completed the task.

https://doi.org/10.1017/S026357472500044X Published online by Cambridge University Press



Figure 13. *Snapshots of the real-world experiment on flat, wooden, grass, and gravel terrains.*

Furthermore, the experiments are conducted on different types of indoor and outdoor terrains, including wooden, grass, and gravel, as shown in Figure 13. During the experiments, the disturbance was applied to the robot in different ways, and the robot can recover to the safe state and resume the task.

6. Conclusions and limitations

This work addresses the problem of safety supervision and guarantee of legged robots. In order to more clearly define the safety of the legged robot, its state space is segmented into four zones. The proposed framework can infer the state of the robot online and select the corresponding policy. This framework can be effectively integrated with various learning-based and control-based methods to facilitate the robot's safety from external supervision. Simulated and real-world experiments demonstrate the effectiveness and improvements of the proposed method.

However, although the experiment validations have demonstrated the effectiveness and improvements of the proposed method, the formal stability proof is not provided in this paper. Furthermore, addressing more complex terrains is a part of future work, which requires a more thorough model to enable accurate safety verification. In addition, the current method requires online safety verification, which brings additional onboard computation. In future works, we can distill the three policies and the safety verification criteria into a single policy before the deployment on the real robot, which can simplify the framework and reduce the online computational complexity.

Author contributions. Ming Sun and Yue Gao designed the study. Ming Sun conducted the experiments and data gathering. Ming Sun and Yue Gao performed statistical analyses. Ming Sun and Yue Gao wrote the article.

Financial support. This work was supported by the National Natural Science Foundation of China (Grant No. 62373242 and No.92248303), Shanghai Municipal Science and Technology Major Project (Grant No. 2021SHZDZX0102).

Competing interests. The authors declare no conflicts of interest exist.

Ethical approval. Not applicable.

References

[1] N. Rudin, D. Hoeller, P. Reist and M. Hutter, "Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning," **In:** *Conference on Robot Learning (PMRL, 2022)*, pp. 91–100.

[2] G. B. Margolis and P. Agrawal, "Walk These Ways: Tuning Robot Control for Generalization with Multiplicity of Behavior," **In:** *Conference on Robot Learning (PMLR, 2023)*, pp. 22–31.

https://doi.org/10.1017/S026357472500044X Published online by Cambridge University Press

[3] I. M. A. Nahrendra, B. Yu and H. Myung, "Dreamwaq: Learning Robust Quadrupedal Locomotion with Implicit Terrain Imagination via Deep Reinforcement Learning," **In:** *IEEE International Conference on Robotics and Automation* (IEEE, 2023) pp. 5078–5084.

[4] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Sci. Rob.* **547**(47), eabc5986 (2020).

[5] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Sci. Rob.* **762**(62), eabk2822 (2022).

[6] S. E. Ada, E. Ugur and H. L. Akin, "Generalization in transfer learning: Robust control of robot locomotion," *Robotica* **40**(11), 3811–3836 (2022).

[7] R. Grandia, A. J. Taylor, A. D. Ames and M. Hutter, "Multi-layered safety for legged robots via control barrier functions and model predictive control," **In:** *IEEE International Conference on Robotics and Automation* (IEEE, 2021) pp. 8352–8358.

[8] Q. Liao, Z. Li, A. Thirugnanam, J. Zeng and K. Sreenath, "Walking in Narrow Spaces: Safety-Critical Locomotion Control for Quadrupedal Robots with Duality-based Optimization," **In:** *IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, 2023) pp. 2723–2730.

[9] J. García and D. Shafie, "Teaching a humanoid robot to walk faster through safe reinforcement learning," *Eng. Appl. Artif. Intell.* **88**, 103360 (2020).

[10] T.-Y. Yang, T. Zhang, L. Luu, S. Ha, J. Tan and W. Yu, "Safe Reinforcement Learning for Legged Locomotion," **In:** *IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, 2022) pp. 2454–2461.

[11] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor and S. Levine, "How to train your robot with deep reinforcement learning: Lessons we have learned," *Int. J. Rob. Res.* **40**(4-5), 698–721 (2021).

[12] B. Thananjeyan, A. Balakrishna, S. Nair, M. Luo, K. Srinivasan, M. Hwang, J. E. Gonzalez, J. Ibarz, C. Finn and K. Goldberg, "Recovery rl: Safe reinforcement learning with learned recovery zones," *IEEE Rob. Autom. Lett.* **6**(3), 4915–4922 (2021).

[13] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Sci. Rob.* **4**(26), eaau5872 (2019).

[14] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez and V. Vanhoucke, "Sim-to-Real: Learning Agile Locomotion for Quadruped Robots," **In:** *Robotics: Science and Systems (RSS, 2018)*, pp. 1–11.

[15] X. Huang, Z. Li, Y. Xiang, Y. Ni, Y. Chi, Y. Li, L. Yang, X. B. Peng and K. Sreenath, "Creating a Dynamic Quadrupedal Robotic Goalkeeper with Reinforcement Learning," **In:** *IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, 2023) pp. 2715–2722.

[16] W. Yu, C. K. Liu and G. Turk, "Protective Policy Transfer," **In:** *IEEE International Conference on Robotics and Automation* (IEEE, 2021) pp. 10595–10602.

[17] T. He, C. Zhang, W. Xiao, G. He, C. Liu and G. Shi, "Agile but safe: Learning collision-free high-speed legged locomotion," arXiv preprint arXiv: 2401.17583 (2024).

[18] S. C. Hsu, X. Xu and A. D. Ames, "Control Barrier Function Based Quadratic Programs with Application to Bipedal Robotic Walking," **In:** *American Control Conference* (IEEE, 2015) pp. 4542–4548.

[19] Q. Nguyen, A. Hereid, J. W. Grizzle, A. D. Ames and K. Sreenath, "3D Dynamic Walking on Stepping Stones with Control Barrier Functions," **In:** *55th Conference on Decision and Control* (IEEE, 2016) pp. 827–834.

[20] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *J. Mach. Learn. Res.* **16**(1), 1437–1480 (2015).

[21] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annu. Rev. Control Rob. Auton. Syst.* **5**(1), 411–444 (2022).

[22] Y. Chow, O. Nachum, E. Duenez-Guzman and M. Ghavamzadeh, "A Lyapunov-Based Approach to Safe Reinforcement Learning," In: *Advances in Neural Information Processing Systems* (2018).

[23] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum and U. Topcu, "Safe Reinforcement Learning via Shielding," **In:** *AAAI Conference on Artificial Intelligence (AAAI, 2018)*, pp. 2669–2678.

[24] S. Gu, L. Yang, Y. Du, G. Chen, F. Walter, J. Wang, Y. Yang and A. Knoll, "A review of safe reinforcement learning: Methods, theory and applications," *IEEE Trans. Pattern Anal. Mach. Intell.* **46**(12), 11216–11235 (2022) arXiv preprint arXiv: 2205.10330.

[25] D. Martínez, G. Alenya and C. Torras, "Safe Robot Execution in Model-based Reinforcement Learning," **In:** *IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, 2015) pp. 6422–6427.

[26] Z. Li, U. Kalabić and T. Chu, "Safe Reinforcement Learning: Learning with Supervision Using a Constraint-Admissible Set," **In:** *Annual American Control Conference* (IEEE, 2018) pp. 6390–6395.

[27] S. H. Lee and A. Goswami, "Fall on backpack: Damage minimization of humanoid robots by falling on targeted body segments," *J. Comput. Nonlinear Dyn.* **8**(2), 021005 (2013).

[28] Y. Kojio, S. Noda, F. Sugai, K. Kojima, Y. Kakiuchi, K. Okada and M. Inaba, "Dynamic fall recovery motion generation on biped robot with shell protector," *IEEE Rob. Autom. Lett.* **6**(4), 6741–6748 (2021).

[29] Y. Ma, F. Farshidian and M. Hutter, "Learning Arm-Assisted Fall Damage Reduction and Recovery for Legged Mobile Manipulators," **In:** *International Conference on Robotics and Automation* (IEEE, 2023) pp. 12149–12155.

[30] W. Ding, X. Chen, Z. Yu, L. Meng, M. Ceccarelli and Q. Huang, "Fall Protection of Humanoids Inspired by Human Fall Motion," **In:** *International Conference on Humanoid Robots* (IEEE, 2018) pp. 827–833.

[31] Y. Wang, M. Xu, G. Shi and D. Zhao, "Guardians as You Fall: Active Mode Transition for Safe Falling," **In:** *International Automated Vehicle Validation Conference* (IEEE, 2024) pp. 1–8.

[32] E. Altman, *Constrained Markov Decision Processes* (Routledge, 2021).

https://doi.org/10.1017/S026357472500044X Published online by Cambridge University Press

[33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and O. Klimov, "Proximal policy optimization algorithms," arXiv preprint arXiv: 1707.06347 (2017).
[34] Unitree. Unitree go1 (2024) [Online]. Available: https://www.unitree.com/cn/go1/.
[35] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa and G. State, "Isaac gym: High performance gpu-based physics simulation for robot learning," arXiv preprint arXiv: 2108.10470 (2021).
[36] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba and P. Abbeel, "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World," **In:** *IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, 2017) pp. 23–30.

Cite this article: M. Sun and Y. Gao (2025). "Safety supervision framework for legged robots through safety verification and fall protection", Robotica **43**, 1691–1707. https://doi.org/10.1017/S026357472500044X

https://doi.org/10.1017/S026357472500044X Published online by Cambridge University Press