

Safe and Sample-efficient Reinforcement Learning for Clustered Dynamic Environments

Hongyi Chen¹ and Changliu Liu²

Abstract—This study proposes a safe and sample-efficient reinforcement learning (RL) framework to address two major challenges in developing applicable RL algorithms: satisfying safety constraints and efficiently learning with limited samples. To guarantee safety in real-world complex environments, we use the safe set algorithm (SSA) to monitor and modify the nominal controls, and evaluate SSA+RL in a clustered dynamic environment which is challenging to be solved by existing RL algorithms. However, the SSA+RL framework is usually not sample-efficient especially in reward-sparse environments, which has not been addressed in previous safe RL works. To improve the learning efficiency, we propose three techniques: (1) avoiding behaving overly conservative by adapting the SSA; (2) encouraging safe exploration using random network distillation with safety constraints; (3) improving policy convergence by treating SSA as expert demonstrations and directly learn from that. The experimental results show that **our framework can achieve better safety performance compare to other safe RL methods during training and solve the task with substantially fewer episodes.**

I. INTRODUCTION

Recently, reinforcement learning (RL) shows promising results in a series of artificial domains; but it's still challenging to develop applicable RL algorithms for a real system due to nine challenges discussed in [1]. Our paper focuses on the two challenges: satisfying safety constraints and learning from limited samples.

Ideally, 0-safety violation should be guaranteed during both training and execution as failures are expensive (damage the real robot systems) and dangerous (hurt humans in the environment). **A comprehensive survey on optimization-based and exploration-based safe RL methods can be found in [2]. Within the optimization-based methods, they redesign the reward function, change the learning objective and add soft constraints to balance the reward and the risk [3–7]; however, no safety guarantee can be derived from these methods. Within the exploration-based methods, they modify the exploration process to avoid risky situations by incorporating external knowledge [8, 9] and using a risk-directed exploration [10]. The recent probability-based methods construct shields to avoid violating the safety constraints, but they only meets the safety requirement with some probability and are hard to be generalized to continuous systems [11, 12]. Control barrier function (CBF) method is used to provide hard constraints for RL and achieve 0-safety violation in**

training [13]. However, this method is only tested in stationary environments like inverted pendulum and car following.

People evaluate the sample efficiency by measuring the amount of data necessary to achieve a certain performance threshold [1]. But collecting sample data in real world is time-consuming and expensive, and RL agents may converge to local optima when the reward is sparse and never reach the performance threshold. In a word, the sample efficiency challenge makes it hard to deploy RL algorithms quickly in real world systems. Exploration methods, like adding action noise, adding parameter space noise (PSN) [14] and using random network distillation (RND) [15] can help to solve sparse-reward problem but it is risky to explore freely in clustered dynamic environment as the systems would fail or break before learning the optimal controller. Leveraging expert demonstration data can accelerate the agent's learning [16], however people usually get suboptimal demonstration as the expert demonstration is hard to access. Recent model-based deep RL approaches show a lot of promise for improving sample efficiency, however, an imperfect dynamics model can degrade the performance of the learning algorithm and lead to suboptimality [17].

In this work, we aim to design a reinforcement learning framework that can learn safely and efficiently even in *clustered dynamic environments*. We use the safe set algorithm (SSA) [18] to ensure safety. SSA has similar structures as barrier function methods, both of which belong to energy function-based safe control [19]. These methods can safe guard any reinforcement learning algorithm, but they can't help to find optimal policies directly, and sometimes make the system behave overly conservative. Thus, we first adapt the projection direction of SSA to generate more efficient control when possible. Also by combining SSA with normal exploration strategies, we can transform these originally unsafe explorations into safe explorations. Moreover, since getting safe expert demonstration is difficult in the real world, we decide to learn from the safe control generated from SSA online to speedup training. The key contributions of this paper are summarized below:

- We propose the SSA based safe RL training framework and prove this framework can guarantee safety with high probability even in clustered dynamic environments, except the cases that no safe control exists.
- We propose three techniques to improve the learning efficiency: adapting the SSA, exploring under safety constraints and learning from SSA demonstration. The numerical results show that we can solve the task with substantially fewer episodes and interactions.

¹Hongyi Chen is with the Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, North Ave NW, Atlanta, GA 30332, USA hchen657@gatech.edu

²Changliu Liu is with the Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, USA cliu6@andrew.cmu.edu

II. PROBLEM FORMULATION

Environment Dynamics The 2D environment contains multiple dynamic obstacles, every obstacle evolves as $\dot{x}_E = f_E(x_E, u_E)$, where the function f_E represents double integrator dynamics, state x_E involves the position and velocity of the obstacle and control u_E represents its acceleration, which is uniformly distributed on the predefined interval.

Robot Dynamics Let $x_R \in X \subset R^{n_x}$ be the robot state containing positions and velocities in x, y axis; $u \in U \subset R^{n_u}$ be the control input, which is the accelerations in x, y axis. The robot dynamics are defined as:

$$\dot{x}_R = f(x_R) + g(x_R)u =: h(x_R, u) \quad (1)$$

where $f(x_R) = [0, \mathbb{I}_2; 0, 0] x_R$, $g(x_R) = [0; \mathbb{I}_2]$. We assume we have the ground truth form of f and g . **Given the dynamical system above, we can formulate an MDP (X, U, δ, r, p) . The transition function $p : X \times U \times X \rightarrow [0, 1]$ is defined as $p(x_{t+1}|x_t, u_t) = 1$ when $x_{t+1} = x_t + h(x_t, u_t)$ and 0 otherwise. The reward function r provides positive reward if reaching the goal state X^* , negative penalty if collide, and zero otherwise. The discounting factor is set to $\delta = 1$.**

Safety Specification The safety specification requires the robot to stay in a closed subset of state space, called the safe set X_S . The safe set can be presented by a zero-sublevel set of a continuous and piecewise smooth function $\phi_0 : R^{n_x} \rightarrow R$, i.e. $X_S = \{x | \phi_0(x) \leq 0\}$. In our problem, ϕ_0 is defined as $d_{min}^2 - d^2$, where d_{min} is the safety distance defined by user and d is the distance from robot to the closest obstacle. For safety metric, we evaluate the percentage of safety violations during training.

Sample Efficiency Specification To evaluate the data efficiency of a particular model, we measure the amount of data necessary to achieve a certain performance threshold:

$$J^{eff} = \min |D_i| \text{ s.t. } R(\text{Train}(D_i)) \geq R_{min}. \quad (2)$$

where D_i is the data used for training the RL policy and R_{min} is the desired performance threshold [1].

Problem The core problem of this paper is to achieve safe and sample-efficient RL learning in clustered dynamic uncertain environment. The learned RL policy will map the state (x_R, x_E^c) to control u , where x_E^c means the closest obstacle to the robot. For safety, we need to monitor and modify the nominal control u to keep the system inside the safe set X_S and achieve least safety-violations. For sample efficiency, we need to ensure RL agent wouldn't converge prematurely to a local optimum and learn the optimal controller with fewer training data.

III. REVIEW OF THE SAFE SET ALGORITHM

The SSA works as a safety monitor [18], which is suitable to safe guard the RL training. The key of SSA is to define a valid safety index ϕ such that 1) there always exists a feasible control input in U that satisfies $\dot{\phi} \leq -\eta\phi$ when $\phi \geq 0$ and 2) any control sequences that satisfy $\dot{\phi} \leq -\eta\phi$ when $\phi \geq 0$ ensures forward invariance of the safe set X_S and finite time convergence to this set. The parameter η is a

positive constant that adjusts the convergence rate. Following the safety index design rule [20] for collision avoidance with single obstacle, we define the safety index ϕ as follows:

$$\phi = d_{min}^2 - d^2 - k \cdot \dot{d}. \quad (3)$$

where \dot{d} is the relative velocity of robot to obstacle and k is a constant factor. **We add higher order term of ϕ_0 to the base ϕ_0 to ensure that relative degree one from ϕ to the control input. As proved in [18][20], this safety index ϕ will ensure forward invariance of the set $\phi \leq 0 \cap \phi_0 \leq 0$ as well as global attractiveness to that set.** With the valid safety index ϕ , we just need to project the reference control u^r to the set of safe controls that satisfy $\dot{\phi} \leq -\eta\phi$ when $\phi \geq 0$, and $\dot{\phi}$ is expressed as

$$\dot{\phi} = \frac{\partial \phi}{\partial x} f + \frac{\partial \phi}{\partial x} g u = L_f \phi + L_g \phi u. \quad (4)$$

We compute ϕ_i for every obstacle and add safety constraint whenever ϕ_i is positive. Also we have velocity and acceleration limits. With all these constraints, SSA will solve the following optimization problem through quadratic programming (QP):

$$\begin{aligned} \min_{u \in U} \|u - u^r\|^2 &= \min_{u \in U} u^T \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} u - 2u^T \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} u^r \\ \text{s.t. } L_f \phi_i + L_g \phi_i u &\leq -\eta \phi_i, i = 1, 2, \dots, m. \end{aligned} \quad (5)$$

However, in clustered dynamic environment, there are situations that don't even exist safe control to guarantee safety as we will discuss later (note ϕ in (3) only guarantees safety with single dynamic obstacle not multiple dynamic obstacles). Besides, low sample efficiency is a problem in vanilla SSA: the agent may require long training period or even fail to learn optimal controller when the task is complex or the environment is reward-sparse. To make it work, we need to improve the sample efficiency with the following three strategies.

IV. METHODOLOGY

A. Adapting the Safe Set Algorithm

Vanilla SSA would output safe control that drives the system to the currently safest direction, which may not be an efficient direction in the long run, see fig. 1a. Besides, in multi-obstacle environment, it's not safe to directly add constraints for every dangerous obstacle whose ϕ is positive. In detail, since vanilla SSA only considers these dangerous obstacles, it may push the robot to the direction that is safe now but risky in the next step if there are unconsidered obstacles (ϕ values are negative) in that direction, see fig. 1b. Thus we decide to consider the current positions, estimate future positions of all approaching obstacles, and modify the direction of safe control by tuning parameters in the QP problem (6). After adapting the projection direction, we expect to generate control signal that is safe to all approaching obstacles, even these ϕ values are negative, and efficient for the longer time horizon.

$$\min_{u \in U} \|u - u^r\|_Q = \min_{u \in U} u^T \begin{bmatrix} \alpha & \sigma \\ \sigma & \beta \end{bmatrix} u - 2u^T \begin{bmatrix} \alpha & \sigma \\ \sigma & \beta \end{bmatrix} u^r. \quad (6)$$

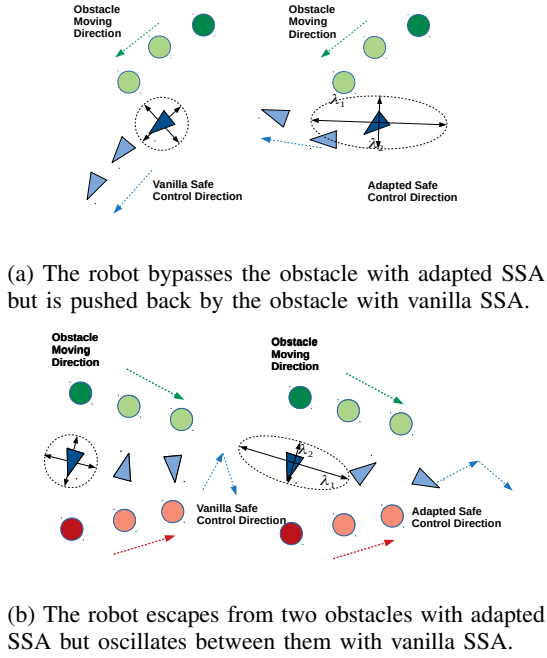


Fig. 1: Comparison between vanilla SSA and adapted SSA. The darker color presents the current positions of the robot and the obstacles. The lighter color presents the future positions of the robot and the obstacles.

In detail, we define the approaching obstacles as those whose distances to the robot are smaller than a threshold value. With the current position of the robot (x_0, y_0) and the current positions of approaching obstacles $(x_i, y_i), i = 1, 2, \dots, n$, we first predict the next k steps positions of each obstacle $(x_i^j, y_i^j), j = 1, 2, \dots, k$ using constant velocity model. Then we solve the line $l_\theta: -\sin(\theta)x + \cos(\theta)y = 0$ that maximizes the distance to all approaching obstacles

$$\max_{\theta} J(\theta), \quad J(\theta) := \sum_{j=1}^k \sum_{i=1}^n d_i^j. \quad (7)$$

where d_i^j is the distance of shifted obstacle $(x_i^j - x_0, y_i^j - y_0)$ to the line l_θ . In this setting, we regard the robot system (x_0, y_0) as the origin and calculate θ_{\max} that has largest overall distance. With θ_{\max} , we can get eigenvector $\mathbf{x}_1 = (-\sin(\theta_{\max}), \cos(\theta_{\max}))$, which is the safest direction for all approaching obstacles, and $J(\theta_{\max})$ is its corresponding eigenvalue λ_1 . The larger λ_1 is, the more we want to project the safe control to \mathbf{x}_1 direction. The second eigenvector \mathbf{x}_2 is perpendicular to \mathbf{x}_1 and has smallest overall distance λ_2 . Then, we can build the QP parameter matrix, which is represented as the ellipse in fig. 1a and fig. 1b, as follows:

$$[\mathbf{x}_1, \mathbf{x}_2] \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} [\mathbf{x}_1, \mathbf{x}_2]^{-1} = \begin{bmatrix} \alpha & \sigma \\ \sigma & \beta \end{bmatrix} =: Q. \quad (8)$$

B. Exploration under Safety Constraints

In real world reward-sparse and clustered dynamic environments, it's challenging to find a sequence of actions that can lead to positive reward and generalize to related situations, thus RL agents need long training time. Traditional

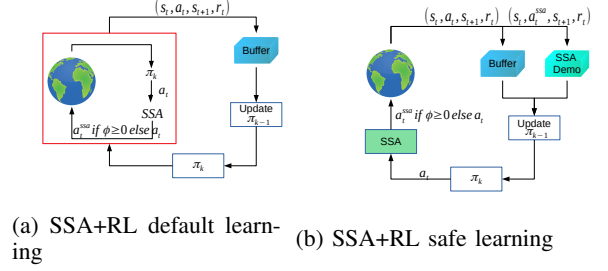


Fig. 2: Block diagrams of the default learning and proposed safe learning. In default learning, the environment (the red box) contains the world and SSA module, while in safe learning, SSA is separated from the environment.

exploration techniques used to address this problem are not suitable for safety-critical tasks, as they may explore unsafe controls. In our framework, with the help of SSA, we add safety constraints to the following two exploration strategies to improve the suboptimal policy safely.

Parameter Space Noise (PSN) [14] At the start of each episode, we create a copy of RL policy and add noise directly to the policy's parameters, which can lead to consistent exploration and a richer set of behaviors. Suppose we parameterize the policy π_θ as a neural network with weights θ . Then the exploration policy is $\pi_{\tilde{\theta}}$, where

$$\tilde{\theta} = \theta + N(0, \sigma^2 I) \quad (9)$$

Random Network Distillation (RND) [15] This exploration strategy will modify the reward function to encourage the agent to visit novel states. In detail, we create two neural networks that take the state $s = (x_R, x_E^c)$ as input and train one of the networks to predict the output of the other. The prediction error of two neural networks is defined as novelty and will be added to reward:

$$\tilde{r}(s, a) = r(s, a) + \|f_{\theta_1}(s) - f_{\theta_2}(s)\|_2^2 \quad (10)$$

C. Learning from SSA Safe Demonstration

Another technique people use to improve sample efficiency is learning from demonstration (LfD) instead of learning from scratch. Different from the traditional LfD or safe controller guided training in [13], we don't need to prepare demonstration data and pre-train the RL agent or approximate all prior safe controllers. Instead, SSA would generate safe controls during the interactions with the world and these safe controls are regarded as expert demonstration. To be more specific, in default SSA+RL framework fig. 2a, SSA is part of the environment which means the RL agent's control signal will be modified when $\phi \geq 0$ and the agent wouldn't realize that. While in safe learning, we separate the SSA from the environment and make it an independent module, see fig. 2b. In this way, the agent could know the world is taking a_t or a_t^{ssa} , then store the self-generated data and demonstration data into two buffers. When updating the policy, we simply use a fixed ratio between self-generated data and SSA demonstrations to mix the training samples. We will explore the optimal ratio in future work.

V. EXPERIMENTS

Environment and Evaluation Method We evaluate our proposed framework in a clustered dynamic environment with sparse reward. Our goal is to move the vehicle, starting from the bottom, to the green area on the top while avoiding 50 moving obstacles in between, which is challenging to be solved by the state-of-the-art RL algorithms alone. A success case is shown in fig. 3a. We assume the vehicle can sense the correct positions and velocities of obstacles, but doesn't know their accelerations. The obstacles will be randomly initialized at each episode. This environment tries to simulate the real world scenarios like parking lots and busy streets that have multiple dynamic objects moving around.

We adopt the Twin Delayed Deep Deterministic Policy Gradients (TD3) [21] as our baseline RL model. For experiments that evaluate safety, we train the models for 50 episodes as we notice it's long enough for the SSA-based models to converge. For experiments that evaluate sample efficiency, we train the models until their performances reach a threshold reward R_{min} . The required R_{min} is to achieve at least 1900 on average for the past 20 episodes. For models that fail to meet R_{min} within 1000 episodes, we set its result as 1000 episodes. We repeat each training with different seeds for 10 times and calculate the average performance. The code is open sourced [here](#).

Hypothesis we evaluate the proposed framework by verifying the following four hypotheses:

- H1: The SSA+RL framework can greatly reduce safety violation comparing to the TD3 baseline model in clustered dynamic environment.
- H2: The adapted SSA can achieve better efficiency and higher task success rate than vanilla SSA.
- H3: Traditional exploration strategies are not safe and hence not data efficient, but exploration under safety constraints could improve the sample efficiency.
- H4: Direct learning from the safe controls demonstrated by SSA can best speedup training and maintain safety compared to pure reward-driven approaches.
- H5: **Compared to other safe RL methods, our SSA+RL framework can achieve best safety performance.**

A. Results

H1: Through our experiments, TD3 model gets 31.7% collision rate and 66.3% failure rate on average, see table I. During training, the policy gradually converges to a local optimum after repeated collisions, which is staying at the bottom where obstacles can't get to. On the other hand, SSA helps to reduce the collision rate from 31.7% to 0.8%. We find SSA can't achieve 0-safety violation discussed in other papers using static testing cases like avoiding fixed obstacles or static hazard areas; in which cases, these always exist a safe control to meet safety constraints [13][20]. But in our environment, the obstacles are dynamic and the vehicle can be surrounded by multiple obstacles. In fig. 3c, there were four obstacles driving from three different directions toward the vehicle, and SSA cannot find a safe control meet

all safety constraints and collision happens (as shown in the control space plot in fig. 5b). Besides, the success rate increased from 2% to 50.2% as SSA prevents the RL agent from collisions and increases the possibility of reaching the goal. But SSA+RL may still stuck in local optimums as its failure rate is 49%. In fig. 3b, even though the vehicle tries to navigate towards the goal area, it is pushed back by obstacles and stays at the bottom to avoid collision.

TABLE I: Safety comparison between proposed models.

	Model	Success	Failure	Collision	Reward
Baseline Models	RL	2%	66.3%	31.7%	-118.6
	PSN+RL	3.6%	76.8%	19.6%	-26
	RND+RL	5.2%	49.5%	47.8%	-133.8
Proposed Models	SSA+RL	50.2%	49%	0.8%	1000
	Adapted SSA+RL	68.6%	30%	1.4%	1365
	PSN+SSA+RL	40.4%	58.3%	1.3%	800
	RND+SSA+RL	71.4%	27.2%	1.4%	1421
	LfD+SSA+RL	89.8%	9.4%	0.8%	1792
	Penalty+SSA+RL	43.8%	55.2%	1.0%	871

H2: The improvement of SSA adaptation is less significant compared to other techniques we used due to two main reasons. Firstly, SSA adaption only works when SSA is triggered, which wouldn't happen at every step. Secondly, in many cases, vanilla SSA and adapted SSA will generate very close safe controls as both of them need to satisfy the same hard safety constraint. As shown in table II, the averaged interaction number of adapted SSA+RL model is 2.3×10^5 , which is very close to the interaction number 2.4×10^5 of SSA+RL model, because the adapted SSA+RL model may also get stuck in the local optimum. Nevertheless, we find the failure rate drops from 49% to 30% and the success rate increases from 50.2% to 68.6% in table I. This improvement is due to that adapted SSA generates fewer detours when meeting dangerous obstacles, and the projection guidance leads the vehicle to a more efficient direction. The effects of adapted safe controls will accumulate and improve the overall success rate and reward. In fig. 3d, adapted SSA and vanilla SSA are tested in the same environment. At the beginning, their paths are identical, but diverge gradually. When meeting dangerous obstacle at the black star point, which is the most critical divergence point, the adapted SSA outputs control in upwards direction while the vanilla SSA outputs control in downwards direction. At last, the adapted SSA helps the vehicle reach the goal while the vanilla SSA pushes the vehicle back to the bottom.

TABLE II: Efficiency comparison between proposed models.

	Model	Episodes	Interaction
Baseline Models	RL	1000	10^6
	PSN+RL	1000	10^6
	RND+RL	1000	10^6
Proposed Models	SSA+RL	254	2.4×10^5
	Adapted SSA+RL	245	2.3×10^5
	PSN+SSA+RL	251	2.4×10^5
	RND+SSA+RL	23	6183
	LfD+SSA+RL	26	8464
	Penalty+SSA+RL	331	3.2×10^5

H3: This hypothesis can be proved via results in table I.

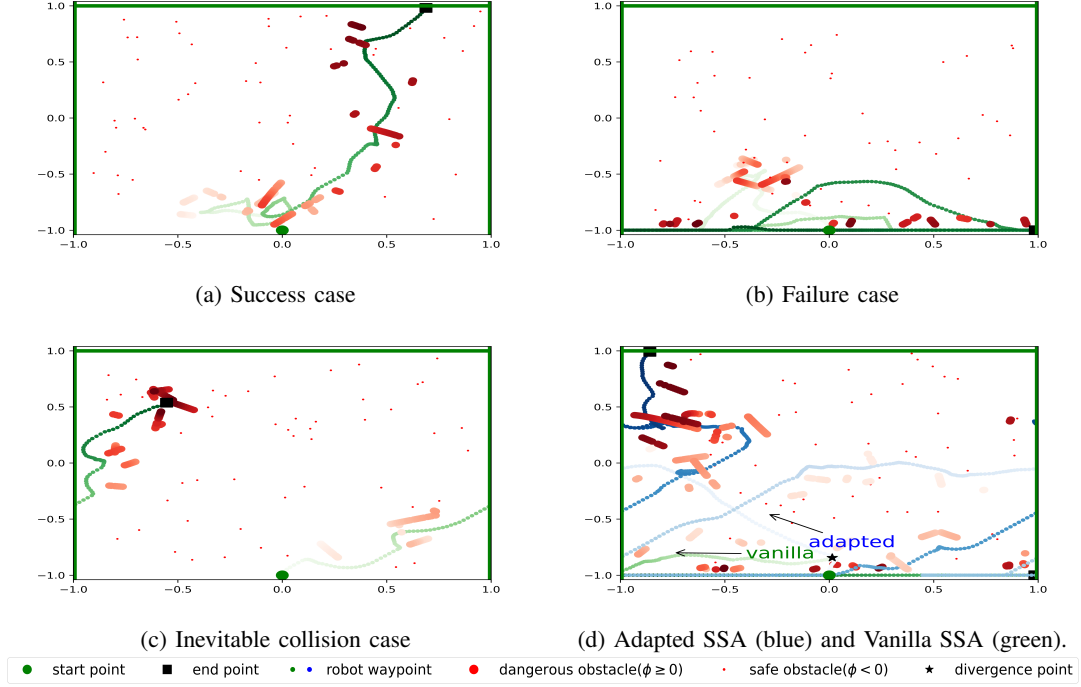


Fig. 3: Trajectory plots of four cases. For both vehicle and obstacles, the darker color means more recent position while the lighter color means older position. The vertical green lines are boundaries that will wrap around. The top green line represents the goal area.

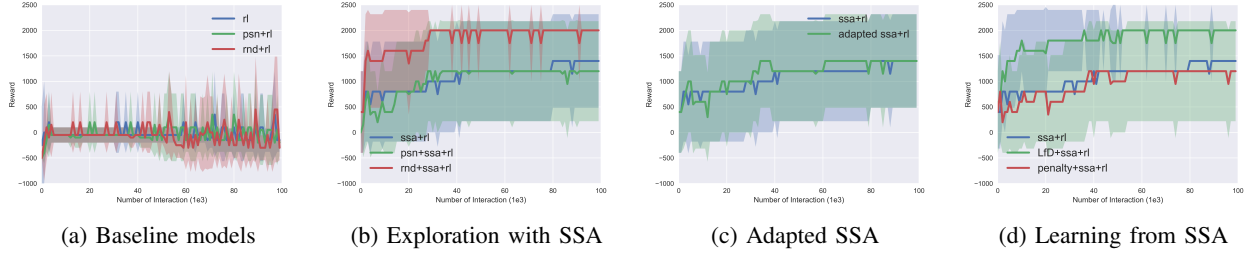


Fig. 4: Average performance of baseline models and our proposed models over 10^6 steps interactions.

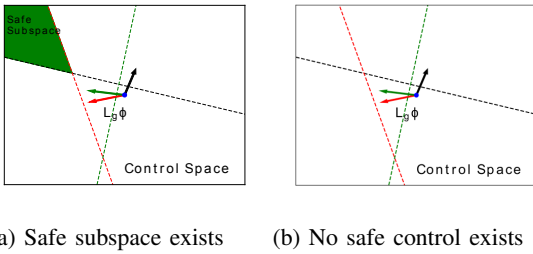


Fig. 5: **SSA feasibility when meeting multiple obstacles.**

The exploration strategies PSN and RND only improve the success rate slightly from 2% to 3.6% and 5.2% respectively. But the failure rate of PSN-enabled model increases to 76.8% and the collision rate of RND-enabled model increases 47.8%. That is mainly because these exploration strategies have large portion of harmful exploration, i.e., unsafe controls that don't meet the safety constraint $\dot{\phi} \leq -\eta\phi$. The RND+SSA+RL model can reduce the failure rate to 27.2% and increase the success rate to 71.4%, which mitigates the problem of being overly conservative in clustered environment. Moreover, in table II, this model uses substantially

fewer episodes and $38\times$ fewer environment interactions to meet R_{min} compared to the SSA+RL model. By visiting new states, the agent has more chance to avoid getting stuck in the local optimum and converge to optimal policy faster. However, PSN+SSA+RL gets worse performance than SSA+RL, which may because PSN is unable to sufficiently explore in our challenging environments.

H4: For LfD+SSA+RL, it takes only 26 episodes and 8464 steps to meet R_{min} , using $28\times$ fewer steps compared to SSA+RL model in table II. From training plot fig. 4d, this model learns the optimal controller in all experiments, achieving faster convergence and greater training stability than other models. Moreover, LfD+SSA+RL achieves highest success rate 89.8% and lowest collision rate 0.8% in table I, which proves that learning from SSA demonstration can best maintain safety. This is due to two reasons: Firstly, there is a mismatch between the control that RL agent generates and the real control that environment takes due to control modification in the default SSA+RL framework. This will introduce errors to the training data and lower the training efficiency. Secondly, the RL policy could better learn

how to take safe control with SSA demonstration, instead of learning from scratch or the reward penalty. To validate the second point, we give the agent a negative reward penalty $-||a_t^{ssa} - a_t||_2^2$ to negatively reinforce unsafe control following the idea in [22]. The results show that penalty+SSA+RL model has lower success rate 43.8%, slightly higher collision rate 1.0% and takes more interactions steps compared to SSA+RL model in table I and table II. This is because learning safe control from the reward penalty is too hard for the agent, which reduces the learning efficiency.

H5 We compare our results with the recent safe RL methods from three different categories, including: constrained policy optimization (CPO) [6], probabilistic shields+RL [12], CBF+RL [13]. CPO will converges to constraint-satisfying policies in the end, but not consistently constraint-satisfying throughout training. Probability shield is applied in discrete MDPs, in which they can calculate the safety violation probability for all possible actions and states within the finite horizon. But this method is hard to check all possible actions in continuous systems and will inevitably be suboptimal. The CBF used in [13] is only tested in simple environments with linear barrier function $h(x) = p^T x + q$. Moreover, the CBF approach in general has limitations than the SSA approach: 1) enforce control constraint everywhere, which is not necessary; 2) our SSA method uses a design rule [18][20] to nonlinearly synthesize ϕ from ϕ_0 which ensures that there always exists a feasible control input to satisfy the safety constraint on control, while the counterpart in CBF is the exponential CBF which is still a linear function on ϕ_0 . Nonetheless, the design rule can also be applied to CBF and we added a comparison of CBF+RL using the same ϕ . In table III, CPO gets high collision rate since this method still relies on trial-and-error to enforce constraints. Probabilistic Shields+RL can only guarantee safety with some probability and is too restrictive to the agent. The barrier function in CBF+RL is too simple to correctly evaluate the safety in higher dimensional spaces. The collision rate of CBF using safety index ϕ is 1.0% higher than that of SSA+RL. The reason is that CBF adds constraints for all nearby obstacles even for safe ones, which restrict the control space and may fail to find safe control for dangerous obstacles.

TABLE III: Safety comparison with other methods.

Model	Success	Failure	Collision	Reward
CPO	7.4%	6.8%	85.8%	-281
Probabilistic Shields+RL	13.8%	80.4%	5.8%	287
CBF+RL	74.6%	21.8%	3.6%	1474
CBF+Safety Index ϕ +RL	52.8%	45.4%	1.8%	1047

VI. CONCLUSION

In this work, we propose to use SSA to improve safety during RL policy training, and introduce three strategies, including SSA adaption, exploration under safety constraint and learning from SSA demonstrations, to improve the learning efficiency. We further validate the proposed framework in a clustered dynamic environment. The results show that SSA can greatly reduce the safety-violation except for the situations that no safe control exists, and achieve better safety

performance statistically compare to other safe RL methods. Combining with the three strategies, the agent can solve the task with substantially fewer episodes and interactions.

REFERENCES

- [1] C. Paduraru et al. “Challenges of Real-World Reinforcement Learning: Definitions, Benchmarks Analysis”. In: *Machine Learning Journal* (2021).
- [2] J. García, Fern, and o Fernández. “A Comprehensive Survey on Safe Reinforcement Learning”. In: *Journal of Machine Learning Research* (2015), pp. 1437–1480.
- [3] P. Geibel and F. Wyszotki. “Risk-Sensitive Reinforcement Learning Applied to Control under Constraints”. In: *Journal of Artificial Intelligence Research* (2005), 81–108.
- [4] A. Ray, J. Achiam, and D. Amodei. “Benchmarking safe exploration in deep reinforcement learning”. In: ().
- [5] M. Heger. “Consideration of Risk in Reinforcement Learning”. In: *Machine Learning Proceedings*. Morgan Kaufmann, 1994, pp. 105–111.
- [6] J. Achiam et al. “Constrained Policy Optimization”. In: *Proc. Int. Conf. Machine Learning*. 2017, 22–31.
- [7] A. Tamar, D. Di Castro, and S. Mannor. “Policy Gradients with Variance Related Risk Criteria”. In: *Proc. Int. Conf. Machine Learning*. 2012.
- [8] P. Abbeel, A. Coates, and A. Ng. “Autonomous Helicopter Aerobatics through Apprenticeship Learning”. In: *The International Journal of Robotics Research* (2010).
- [9] P. Vidal et al. “Learning on real robots from experience and simple user feedback”. In: *Journal of Physical Agents* (2013).
- [10] C. Gehring and D. Precup. “Smart exploration in reinforcement learning using absolute temporal difference errors”. In: *Int. Conf. Autonomous Agents and Multiagent Systems*. 2013.
- [11] M. Hasanbeig, A. Abate, and D. Kroening. “Cautious reinforcement learning with logical constraints”. In: *AAMAS* (2020).
- [12] N. Jansen et al. “Shielded Decision-Making in MDPs”. In: *ArXiv abs/1807.06096* (2018).
- [13] R. Cheng et al. “End-to-End Safe Reinforcement Learning through Barrier Functions for Safety-Critical Continuous Control Tasks”. In: *Proc. AAAI Conf. Artificial Intelligence*. 2019, pp. 3387–3395.
- [14] M. Plappert et al. “Parameter Space Noise for Exploration”. In: *Proc. Int. Conf. Learning Representations*. 2018.
- [15] Y. Burda et al. “Exploration by random network distillation”. In: *Proc. Int. Conf. Learning Representations*. 2019.
- [16] T. Hester et al. “Learning from Demonstrations for Real World Reinforcement Learning”. In: *arXiv preprint arXiv:1704.03732* (2017).
- [17] J. Buckman et al. “Sample-Efficient Reinforcement Learning with Stochastic Ensemble Value Expansion”. In: *Proc. Int. Conf. on Neural Information Processing Systems*. 2018.
- [18] C. Liu and M. Tomizuka. “Control in a safe set: Addressing safety in human-robot interactions”. In: *Dynamic Systems and Control Conference*. 2014, V003T42A003.
- [19] T. Wei and C. Liu. “Safe Control Algorithms Using Energy Functions: A Unified Framework, Benchmark, and New Directions”. In: *Proc. IEEE Conf. Decision and Control*. 2019, pp. 238–243.
- [20] W. Zhao, T. He, and C. Liu. “Model-free Safe Control for Zero-Violation Reinforcement Learning”. In: *5th Annual Conference on Robot Learning*. 2021.
- [21] S. Fujimoto, H. van Hoof, and D. Meger. “Addressing Function Approximation Error in Actor-Critic Methods”. In: *Proc. Int. Conf. Machine Learning*. 2018, pp. 1582–1591.
- [22] W. Saunders et al. “Trial without Error: Towards Safe Reinforcement Learning via Human Intervention”. In: *Proc. Int. Conf. Autonomous Agents and MultiAgent Systems*. 2018.