Safe and Sample-efficient Reinforcement Learning: Learn how to drive in crowded and dynamic environment

Hongyi Chen[1] and Changliu Liu[2]

***Abstract*— Reinforcement Learning (RL) algorithms have found limited success beyond simulated applications due to safety constraints and limited samples in real-world. This paper presents a safe and efficient RL framework based on Safe Set Algorithm (SSA). Using SSA, we project the unsafe actions to safe actions to avoid risky states during learning process. Moreover, we integrate parameters tuning, exploration strategies and learning from demonstration into SSA based RL framework to speedup training and improve sample efficiency. To test it, we design a autonomous driving task in crowded and dynamic environment, which exceed the difficulty that can be solved by RL algorithm alone. The testing results show that our framework can guarantee safety 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 very hard to develop applicable RL algorithms for a real production system. Dulac-Arnold et.al [1] presented nine unique challenges that explain why RL in the real-world are more difficult than RL in research. Our paper focuses on the two challenges we think that are fundamentally necessary to be addressed: satisfying safety constraints and learning from limited samples. Ideally, 0-Safety violation should be guaranteed during system operation and the learning phase as the failures would be expensive (damage the real robot systems) and dangerous (hurt humans in the environment). To solve the safety challenge, Geibel et.al [2] proposed to shape the reward with the probability (risk) of reaching an "error state", Achiam et.al [3] developed Constrained Policy Optimization (CPO), as well as forms with adaptive penalties for safety costs based on the Lagrangian approach to constrained optimization. While these methods allowed for more general learning strategies, no guarantees could be derived from the algorithms. Recently, Cheng et.al [4] combines the Barrier Function Method (BFM), a safe control algorithm that provide hard safety guarantee, with RL and achieve no safety violation in training. This method is only tested in simple stationary environment like Inverted Pendulum and Car Following. In fact, most of the current RL research only test their algorithms in obstacle-free environments, which are far from the unstructured or crowded real-world cases. To be more specific, RL shows promising results in autonomous driving tasks like lane-following [5], but their testing environment is over-simplified without any static or dynamic obstacles. In reality, autonomous vehicle needs to learn how to drive safely in crowded environment like parking lot or city roads with many pedestrians and vehicles moving around.

People evaluate the sample efficiency by looking at the amount of data necessary to achieve a certain performance threshold [1]. As almost all real-world systems are slow-moving, it's time-consuming to collect massive sample data for agent training. What's worse, it's possible that the agent may converge prematurely to a local optimum due to sparse reward problem. In other word, sample efficiency challenge make it hard to deploy applicable RL algorithms quickly in real-world systems. One commonly used method is encouraging exploration, like adding action noise, adding parameter space noise (PSN) [6] and using random network distillation (RND) [7], to solve the sparse reward problem and find optimal solution. However, it's extremely risky to allow RL policy to explore freely in crowded dynamic environment as it would fail before converging to optimal solution. To avoid unnecessary exploration, imitation learning is also a commonly used approach. In DeepMind'spaper [8], they leverages very small amounts of demonstration data to massively accelerate learning for autonomous driving. However, perfect expert demonstration is hard to access and people usually get suboptimal demonstrations whose trajectories may take longer than necessary to solve the problem [9].

In this work, we aim to design a safe and sample-efficient reinforcement learning algorithm that can train the agent to drive in a crowded and dynamic environment, which simulate the driving task in crowded real-world environment. The Safe Set Algorithm (SSA) we adopt is able to safe guard any reinforcement learning algorithm by projecting the unsafe action to safe action in risky state and guarantee safety in theory, but it doesn't have any idea about how to solve the problem or how to improve sample efficiency. By incorporating SSA into normal exploration strategies like PSN and RND, these unsafe exploration can be transformed into safe exploration as SSA filter out these unsafe exploration actions. The same applies to imitation learning, since getting a safe expert demonstration is sometimes difficult in real-world, we could save our effort to learn from the safe actions generated from SSA to speedup training. Besides, we find we could tune the parameters inside SSA to change the safe action projection direction and generate the safe action that achieve better efficiency in crowded environment.

The key contributions of this paper are summarized below:

- We propose the SSA based safe RL training algorithm.

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

[2]Changliu Liu is with Faculty of Electrical and Computer Engineering, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, USA cliu6@andrew.cmu.edu

We show that this safe learning algorithm can guarantee safety even in crowded and dynamic environment.

- We propose three techniques to improve the sample-efficiency in learning: tuning parameters in quadratic programming solver, combining with safe exploration and learning from SSA demonstration. The numerical results show that we can solve the task with substantially fewer episodes and interactions.

II. Problem Formulation

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

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

where $f : X \times U \to X$ is a function that maps the vehicle state and control at current time step to the vehicle state in the next time step. For simplicity, this paper considers deterministic dynamics. Moreover, it is assumed that we can access the ground truth form of f.

Safety Specification The safety specification requires that the system state should be constrained in a closed subset in the 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 $\phi_0 : R^{n_x} \to R$, i.e. $X_S = \{x|\phi_0(x) \leq 0\}$. X_S and ϕ_0 are directly specified by users. For metrics that can provide a good global summary, we count the number and the percentage of safety violations, which is collisions in our case, during the training[10].

Sample Efficiency Specification To evaluate the data efficiency of a particular method, we use the measurement method in [1], that is measuring the amount of data necessary to achieve a certain performance threshold:

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

where D_i is the data used for training RL agent and R_{min} is the desired performance threshold.

Simulation Environment We design a dynamic obstacles crowded environment with sparse reward, shown in Figure 1. The entire space is a 2 × 2 square area, x-axis and y-axis range from [-1, 1]. In this environment, our goal is to move the triangle vehicle, starting from the bottom, to the green area on the top while avoiding moving obstacles in between. Our agent can only get a huge positive reward after reaching goal, and negative reward for collision and get zero reward otherwise. The gray dots are moving obstacles, whose starting position, velocity and acceleration are randomized at every episode. We use double integrator dynamic model for ego vehicle and assume the vehicle can sense the correct positions and velocities of approaching obstacles. Also, we allow the vehicle to across from left to right or from right to left, that is if vehicle's x-coordination is large than 1, like 1.05, it will be mod to -0.95.

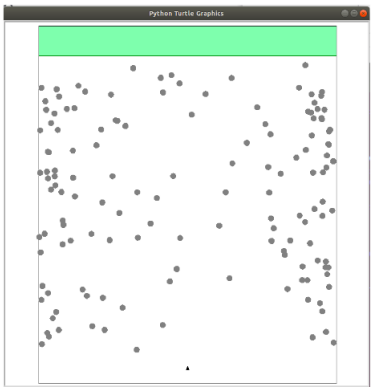

Fig. 1. Simulation Environment

Problem The core problem of this paper is to achieve safe and efficient RL learning in crowded dynamic environment. For safety, we need to monitor and modify the nominal control to ensure forward invariance in a subset of the safe set X_S and finite time convergence to that subset. For sample efficiency, we need to make sure RL agent wouldn't converge prematurely to a local optimum and find the optimal solution with fewer training episodes and environment interactions.

A. Safe Set Algorithm

In SSA, we define a safety index ϕ which maps safe states to low values and unsafe states to high values

$$\phi = d_{min}^2 - d^2 - k \cdot \dot{d}^2 \tag{3}$$

where d_{min} is the safety distance threshold define by user, d is the shortest distance from robot to obstacle at current time, k is a constant factor. Whenever the systems drive into unsafe state or in region of attraction, that is ϕ is positive, SSA will generate a safe control that minimize the safety index and push the systems back to safe region. Given the affine dynamic system $\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + g(\boldsymbol{x})\boldsymbol{u}$, we add a safety constraint about the gradient of safety index $\dot{\phi} \leq \eta$, where

$$\dot{\phi} = \frac{\partial \phi}{\partial x} f + \frac{\partial \phi}{\partial x} g\ \boldsymbol{u} = L_f \phi + L_g \phi\ \boldsymbol{u} \tag{4}$$

Except this safety constraint, we also have velocity and acceleration limits. With all these constraints, SSA solve this optimization problem through quadratic programming (QP), which project the originally unsafe control $\boldsymbol{u^r}$ to new safe control $\boldsymbol{u}$

$$\begin{aligned} \boldsymbol{u} &= \min_{\boldsymbol{u} \in U} ||\boldsymbol{u} - \boldsymbol{u^r}||^2 \\ s.t. L_f \phi &+ L_g \phi\ \boldsymbol{u} \leq \eta \end{aligned} \tag{5}$$

The safe set algorithm discussed in equation 5 is for continuous time systems. One applied, it ensures that the constraint $\phi(x) \leq 0$ is satisfied at all time if the initial state satisfies the constraint. The result is proved in [11]. However, in reality, many systems are controlled in discrete time, which means that the controller may not be able to respond immediately to potential violations of safety.

The problems in vanilla SSA lie in low sample efficiency: too many failures, meaning the vehicle neither collide nor

reach the goal, in the reward sparse environment as SSA can only guarantee safety, see Figure 6. Except the triangle vehicle in green area, all other red (which means the vehicle is adopting the safe action from SSA in this timestep) and black vehicles fail to reach the goal before the time limits. To make it work in real-world task, we need to improve the sample efficiency by introducing parameter tuning, safe exploration and safe expert.

B. Quadratic Programming (QP) Parameter Tuning

Vanilla SSA would output safe control that drive the systems to the safest direction, which is usually not an efficient direction to achieve the goal and may even drive the systems back and forth between multi obstacles before it finds safe control for all obstacles, see Figure 2, 3, 4. The default parameters in QP solver are identity matrixs, thus we can rewrite the objective into this equation

$$\min_{\boldsymbol{u}} ||\boldsymbol{u}-\boldsymbol{u^r}||^2 = \min_{\boldsymbol{u}} \boldsymbol{u}^{\mathbf{T}} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{u} - 2\boldsymbol{u}^{\mathbf{T}} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{u^r} \quad (6)$$

The default L2 norm in SSA objective function may not generate safe action for all surrounding dynamic obstacles as it only consider those whose ϕ values are positive. Thus we decide to tune the parameters in QP solver which can project the $\boldsymbol{u^r}$ to the direction that is safest to approaching obstacles, even these ϕ values are negative and meet the safety constraints, see Figure 5. We define the approaching obstacles are those whose distances are smaller than a threshold distance value.

$$\min_{\boldsymbol{u}} ||\boldsymbol{u}-\boldsymbol{u^r}||^2 = \min_{\boldsymbol{u}} \boldsymbol{u}^{\mathbf{T}} \begin{bmatrix} \alpha & \sigma \\ \sigma & \beta \end{bmatrix} \boldsymbol{u} - 2\boldsymbol{u}^{\mathbf{T}} \begin{bmatrix} \alpha & \sigma \\ \sigma & \beta \end{bmatrix} \boldsymbol{u^r} \quad (7)$$

In detail, with the positions of approaching obstacles $(x_i, y_i), i = 1, 2, ..., n$ and the position of our system (x_0, y_0), we first solve the line $l1 : -sin(\theta)x + cos(\theta)y = 0$ that maximize $J(\theta)$ by finding its global maximum point through derivative.

$$\max_{\theta} J(\theta) = \max_{\theta} \sum_{i=1}^{n} d_i \quad (8)$$

where d_i means the distance of obstacle $(x_i - x_0, y_i - y_0)$ to the line. The direction of line l_1 is the eigenvector $\boldsymbol{x}_1$, which is the direction we want the $\boldsymbol{u^r}$ to be projected to, and $J(\theta)$ is its corresponding eigenvalue λ_1, meaning how much we want to project $\boldsymbol{u^r}$ to $\boldsymbol{x}_1$ direction. The line that is perpendicular to $\boldsymbol{x}_1$ is the second eigenvector $\boldsymbol{x}_2$, which has smallest total distance λ_2. We can build the QP parameter matrix as follows:

$$[\boldsymbol{x}_1, \boldsymbol{x}_2] \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} [\boldsymbol{x}_1, \boldsymbol{x}_2]^{-1} = \begin{bmatrix} \alpha & \sigma \\ \sigma & \beta \end{bmatrix} \quad (9)$$

C. Safe Exploration

In real-world reward sparse and crowded dynamic environment, it's very hard to find a sequence of actions that lead to positive reward and generalize to related situations, thus RL agent needs long training time period before converge to optimal policy or even converge prematurely to a local optimum. For example, in this Figure 6, the vehicle explores and fails many times at the beginning of training. The samples collected from these failed episodes cause the policy converge to a local optimum and the vehicle just move at the bottom line. Without exploration or other methods, the agent will keep repeating the problem and collecting garbage data. However, Traditional exploration techniques are not particularly useful for solving safety-critical tasks, as them may select action that cause damage to the systems, which need to be avoided in real-world. But, with the help of SSA, we do exploration safety to improve the suboptimal policy and sample efficiency. We use the following two exploration strategies with SSA to monitor and modify the exploration actions to keep the vehicle stat inside the safe set.

Parameter Space Noise (PSN)[6] This method adds noise directly to the agent's parameters, which can lead to more consistent exploration and a richer set of behaviors. At the start of each episode, we create a copy of RL policy and slightly perturb the weights. Suppose we parameterize policy π_θ as a neural networks with weights θ. Then the exploration policy is $\pi_{\widetilde{\theta}}$, where

$$\widetilde{\theta} = \theta + N(0, \sigma^2 I) \quad (10)$$

Random Network Distillation (RND)[7] Another exploration strategy we adopt is modifying the reward function to encourage the agent to visit novel states. We measure novelty via the prediction error of two neural networks, which are initialized with different random weights. In detail, we create two neural networks that take the state as input and output a 128-dimensional vector, then we train one of the networks to predict the output of the other. The prediction error in modified reward function is as follows:

$$\widetilde{r}(s, a) = r(s, a) + ||f_{\theta_1}(s, a) - f_{\theta_2}(s, a)||_2^2 \quad (11)$$

D. Safe Learning

Besides the safe exploration, another techniques that people use to improve sample efficiency is learning from demonstration instead of training from scratch. In DeepMind's paper[8], they leverages very small amounts of demonstration data to massively accelerate learning for autonomous driving. Thus, we regard the safe action generated by SSA as demonstration from expert and update RL policy with a mix of demonstration data from SSA and self-generated data. To be more specific, SSA is part of the environment in default SSA+RL algorithm, which means the action generated by RL policy is projected by SSA when $\phi \geq 0$. Now, we separate the SSA from the environment and regard it as our safe expert, see Figure . Unlike traditional learning from demonstrations, we don't need to prepare demonstration data and pre-train the RL agent. Instead, SSA would generate demonstrations automatically during the interaction with environments and we train the agent with the demonstrations online. In practice, people use techniques like prioritized replay mechanism [12] to automatically control the ratio between demonstration and self-generated data to improve performance. But this is not the focus of our paper, we simply

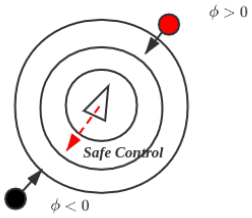

Fig. 2. Vanilla SSA: step 1

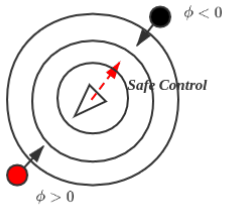

Fig. 3. Vanilla SSA: step 2

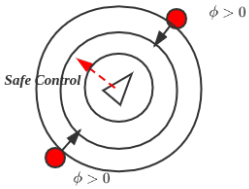

Fig. 4. Vanilla SSA: step 3

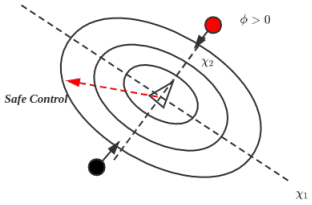

Fig. 5. QP Tuned SSA

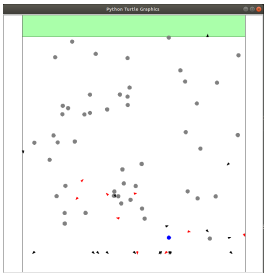

Fig. 6. Fail case1

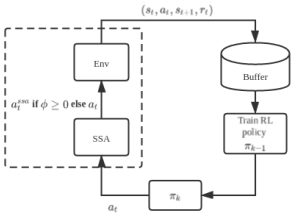

Fig. 7. SSA+RL Default Learning

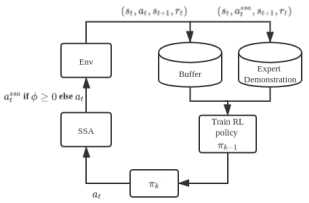

Fig. 8. SSA+RL Safe Learning

set the fixed ratio to make sure that demonstration data plays an important role in updating the policy. In default SSA+RL learning, see Figure 7, we regard the environment and the SSA as a whole environment, that is the policy action a_t would be modified by SSA and the agent doesn't realize it. While in SSA+RL safe learning, we spin off the SSA into a separate module, later we store the self-generated data and demonstration data into two buffers. When updating the policy, we mix the training data from two buffers based on the ratio.

III. Experiments

We test our algorithm in our simulation environment with 50 dynamic obstacles, shown in Figure 1. We adopt the commonly used actor-critic RL algorithms Twin Delayed Deep Deterministic Policy Gradients (TD3)[13] as our baseline RL policy. TD3 is built upon the DDPG algorithm and tries to address the overestimation bias problem with DDPG. Instead of using one critic network Q_θ and its corresponding target $Q_{\theta'}$, TD3 uses two critics Q_{θ_1}, Q_{θ_2} and two corresponding targets $Q_{\theta'_1}$, $Q_{\theta'_2}$. For both critics, the target update is computed using the minimum of two target networks (Clipped Double Q-Learning), which successfully reduces the overestimation bias in off-policy RL. For safety experiments, we train the RL policy with different combinations for 50 episodes as we notice SSA can greatly speedup the RL training compare to the baseline RL model and 50 episodes are enough for the agent to converge. Success means the vehicle arrives at goal area within 1000 timesteps while failure means that the vehicle doesn't reach goal within time limit. Collision means the vehicle collide with obstacles. For sample efficiency experiments, we also train the different variation of SSA+RL and stop only when the performance reach the threshold reward R_{min} or the episodes number are larger than 1000. In detail, the reward of reaching goal is 2000, the penalty of collision is -500 and the reward of failure is 0. The threshold reward R_{min} requires the agent to achieve at least 1900 on average for the past 20 episodes. We count the number of episodes and the number of interacting with environment, the biggest interaction number allowed within a single episode is 1000. Please visit .... for access to the code.

A. Safety Results

(1) Normal RL policy gradually converges to a local optimum, which is staying at the bottom positions where obstacles can't get to, after repeated collisions. See figures 9, 10 and 11 for detailed trajectory. The green dot and red dot mean the starting and ending position. The darker dot mean more recent position while the ligher dot mean older position. We can see from Fig 10 that the vehicle does exploration and try to drive to the goal area, but it is pushed back by obstacles or by the random actions generated by policy. The second failure case in Fig 11 is even worse, the vehicle stops exploration quickly and stays at the bottom line. Thus, baseline RL model has high failure rate and collision rate. On the other hand, SSA helps the agent to avoid risky states and reduce the collision rate from 31.7% to 0.8%. In our dynamic crowded environment, we find SSA can't achieve 0-safety violation discussed in other papers using simple testing environment [4]. First of all, SSA can only theoretically guarantee safety for continuous time systems while our system is controlled in discrete time, which means that the controller may not be able to respond immediately to potential violations of safety. Second, the dynamic obstacles may move in an unpredictable way, and the vehicle can be surrounded by multi obstacles and fails to find a safe control in this situation. Besides, we notice the successful rate increased from 2% to 50.2%. The main

reason is that by avoiding collisions, SSA prevent the RL policy from converging to local optimum and allow for more exploration which increases the possibility of reaching the goal. Combining with QP parameter tuning, the failure rate drops from 51% to 30% and success rate increases from 50.2% to 68.6% in Table I. We think this may due to fewer detours when meeting multi-obstacles situations, and projection guidance lead the vehicle to a more efficient way which also ensure safety. The effects of these QP tuned safe actions will accumulate and improve the overall success rate and reward.

(2) The normal exploration strategies don't improve the performance in our crowded environment which mean the random explorations would only cause collision. However, combining with SSA, we can transform the random explorations into safe explorations by filtering out the harmful actions. The SSA-Based RND exploration can greatly reduce the failure rate and increase the success rate, which solve the problem of being over-conservative in RL training in risky environment, thus improve the sample efficiency. However, not every exploration method can work well with SSA, like PSN+SSA+RL get slightly worse performance than SSA+RL.

(3) Learning from SSA demonstration generates the best results, we think there are two possible reasons: First, there is a mismatch between the action that agent think environment takes and the real action that environment takes due to unsafe action modification in the default SSA+RL learning. Because of this error training data, the training efficiency will be lower. Secondly, learning from the SSA demonstration can greatly accelerate RL agent training by avoiding repeated trial-and-error, RL agent could better learn how to take safe action and achieve higher success rate.

TABLE I
EXPERIMENTAL RESULTS OF RL AND VANILLA SSA+RL

	Model	Success	Failure	Collision	Reward
SSA	RL	2%	66.3%	31.7%	-118.6
	SSA+RL	50.2%	51%	0.8%	1000
	QP Tuned SSA+RL	68.6%	30%	1.4%	1365
Exploration	PSN+RL	3.6%	76.8%	19.6%	-26
	RND+RL	5.2%	49.5%	47.8%	-133.8
	PSN+SSA+RL	40.4%	58.3%	1.3%	800
	RND+SSA+RL	71.4%	27.2%	1.4%	1421
Learning	Learning+SSA+RL	89.8%	9.4%	0.8%	1792

B. Sample Efficiency Results

We set the maximum training episodes to 1000 and maximum environment interaction steps to 10^6, We pad those who take longer to reach the reward threshold with 1000 episodes and 10^6 interactions. For each model's training, we repeat the experiments with different seeds for 10 times and calculate the average samples needed over 10 experiments for robustness. Also, we use the same hyper parameters for all models.

(1) We notice that models without SSA, including RL, PSN+RL, RND+RL, wouldn't be able to meet the reward threshold requirement within time limit. The possible reason is the designed environment is too difficult for the agent to find a solution. On the other hand, the SSA enabled models are capable to solve the task while the learning speed of individual model differs. From the table II, the QP parameter tuning only gives a small margin over the default SSA+RL. The reason behind is that both algorithms fail to find the solution and have 10^6 times interactions in some experiments, and QP parameter tuning has better sample efficiency on other experiments that solve the problem. For the

(2) Compared to the SSA+RL model, exploration enabled SSA+RL models use substantially fewer episodes and environment interactions. Especially the RND based SSA+RL (RND+SSA+RL) and safe learning based SSA+RL (Learning+SSA+RL) meet the threshold within 30 episodes on average, using $38\times$ and $28\times$ fewer environment interactions compared to SSA+RL. The high sample efficiency of exploration encouraged SSA+RL is that it's usually difficult for vanilla SSA+RL to find a sequence of actions that solve the problem in a reward sparse environment. By encouraging visiting new states or adding consistent exploration noise in parameter space, the agent has more chance to avoid getting stuck in the local optimum and converge to optimal solution faster. For safe learning based SSA+RL, even we don't have expert demonstration in advance, the SSA generated $< s_t, a_t, s_{t+1}, r_t >$ safe data is good enough to bootstrap the RL agent, rather than learning from repeated collisions. From Fig 13, 14, we find that the rnd exploration and safe learning techniques have great training stability, that the agent can converge to find the solution in all 10 experiments. While other models have big deviations as they get 0 reward sometimes.

We also adjust the difficulty of the environment, that is change to 30 obstacles instead of 50.

TABLE II
EXPERIMENTAL RESULTS OF RL AND VANILLA SSA+RL

	Model	Episodes	Interaction
SSA	RL	1000	10^6
	SSA+RL	254	2.4×10^5
	QP Tuned SSA+RL	245	2.3×10^5
Exploration	PSN+RL	1000	10^6
	RND+RL	1000	10^6
	PSN+SSA+RL	154	1.4×10^5
	RND+SSA+RL	23	6183
Learning	Learning+SSA+RL	26	8464

IV. CONCLUSION

With the success in massive simulation tasks, RL methods are still not capable to solve real-world tasks due to safety and sample efficiency issues. We propose to use SSA to guarantee safety during policy training, and adopt three strategies including QP parameter tuning, safe exploration and learning

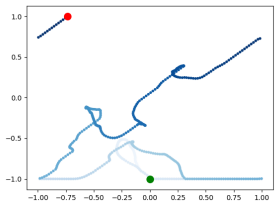

Fig. 9. Success case

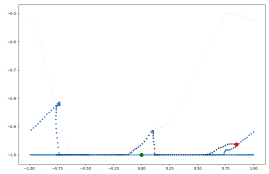

Fig. 10. Failed Case1

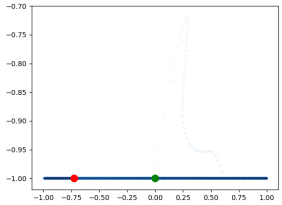

Fig. 11. Failed Case2

Fig. 12. Avg Performance of QP SSA

Fig. 13. Avg Performance of Exploration+SSA

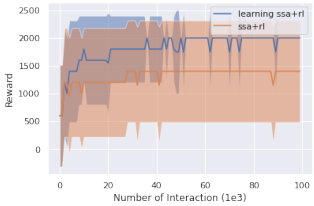

Fig. 14. Avg Performance of Learning+SSA

from SSA demonstration, to improve the sample efficiency. We further validate the proposed method in a crowded dynamic environment. The results show that SSA could avoid most of the risky states except for those inevitable collisions. Using the strategies we proposed, the agent can solve the task with substantially fewer episodes and interactions.

REFERENCES

[1] Gabriel Dulac-Arnold, Daniel J. Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *CoRR*, abs/1904.12901, 2019.

[2] Peter Geibel and Fritz Wysotzki. Risk-sensitive reinforcement learning applied to control under constraints. *J. Artif. Int. Res.*, 24(1):81–108, July 2005.

[3] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. *CoRR*, abs/1705.10528, 2017.

[4] Richard Cheng, Gábor Orosz, Richard M. Murray, and Joel W. Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. *CoRR*, abs/1903.08792, 2019.

[5] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day. *CoRR*, abs/1807.00412, 2018.

[6] Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y. Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. *CoRR*, abs/1706.01905, 2017.

[7] Yuri Burda, Harrison Edwards, Amos J. Storkey, and Oleg Klimov. Exploration by random network distillation. *CoRR*, abs/1810.12894, 2018.

[8] Todd Hester, Matej Vecerík, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Andrew Sendonaris, Gabriel Dulac-Arnold, Ian Osband, John P. Agapiou, Joel Z. Leibo, and Audrunas Gruslys. Learning from demonstrations for real world reinforcement learning. *CoRR*, abs/1704.03732, 2017.

[9] Ajay Mandlekar, Fabio Ramos, Byron Boots, Li Fei-Fei, Animesh Garg, and D. Fox. Iris: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4414–4420, 2020.

[10] Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerík, Todd Hester, Cosmin Paduraru, and Yuval Tassa. Safe exploration in continuous action spaces. *CoRR*, abs/1801.08757, 2018.

[11] *Control in a Safe Set: Addressing Safety in Human-Robot Interactions*, volume Volume 3: Industrial Applications; Modeling for Oil and Gas, Control and Validation, Estimation, and Control of Automotive Systems; Multi-Agent and Networked Systems; Control System Design; Physical Human-Robot Interaction; Rehabilitation Robotics; Sensing and Actuation for Control; Biomedical Systems; Time Delay Systems and Stability; Unmanned Ground and Surface Robotics; Vehicle Motion Controls; Vibration Analysis and Isolation; Vibration and Control for Energy Harvesting; Wind Energy of *Dynamic Systems and Control Conference*, 10 2014. V003T42A003.

[12] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay, 2016.

[13] Scott Fujimoto, H. V. Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *ArXiv*, abs/1802.09477, 2018.