Null Space Based Efficient Reinforcement Learning with Hierarchical Safety Constraints

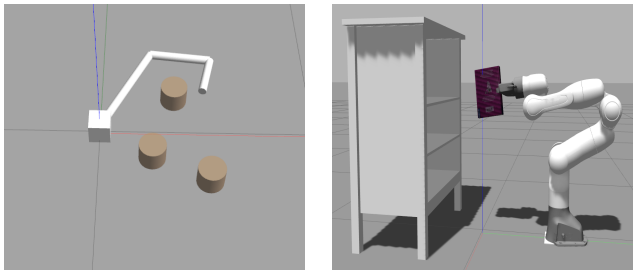
Quantao Yang[1], Johannes A. Stork[1], and Todor Stoyanov[1]

***Abstract*— Reinforcement learning is inherently unsafe for use in physical systems, as learning by trial-and-error can cause harm to the environment or the robot itself. One way to avoid unpredictable exploration is to add constraints in the action space to restrict the robot behavior. In this paper, we propose a null space based framework of integrating reinforcement learning methods in constrained continuous action spaces. We leverage a hierarchical control framework to decompose target robotic skills into higher ranked tasks ($e.g.$, joint limits and obstacle avoidance) and lower ranked reinforcement learning task. Safe exploration is guaranteed by only learning policies in the null space of higher prioritized constraints. Meanwhile multiple constraint phases for different operational spaces are constructed to guide the robot exploration. Also, we add penalty loss for violating higher ranked constraints to accelerate the learning procedure. We have evaluated our method on different redundant robotic tasks in simulation and show that our null space based reinforcement learning method can explore and learn safely and efficiently.**

I. Introduction

Trial-and-error learning (such as Reinforcement Learning) poses challenges when applied to physical robotics. Reinforcement learning (RL) involves performing a number of exploratory actions, often with a degree of randomness, which can lead to damage of the robot or its environment. This problem has been previously addressed by learning in simulation [1], [2], [3], safety exploration [4], [5], imitation learning [6], [7] and learning from demonstration (LfD) [8], [9]. However, some of these solutions do not guarantee safety, while others face difficulties when transferring from simulated to real environments. Another challenge applying RL on robot system is sampling efficiency because it is unfeasible for the robot to interact with its surroundings to collect millions of experience samples. Therefore, it is imperative to address these problems in order to enable RL on real physical robots.

Classical approaches to safe RL aim to reduce undesirable exploration results by limiting policy updates between iterations [10], [11]. However, it is hard to determine the best update step and these methods still might over time result in visiting unsafe states. In contrast, we propose a null space based framework that enables safe RL exploration in constrained continuous action spaces. We leverage a hierarchical framework that can guarantee constraint satisfaction in the least-square sense during the robot's trial-and-error exploration, where constraints are used to describe safety conditions. We base our work on Safe-To-Explore State Space (STESS) [4], which restricts the robotic operational space to a collision-free space. This is accomplished by decomposing a robotic skill ($e.g.$, putting a book into a cabinet) into different prioritized tasks. Higher ranked safety tasks are prioritized over lower ranked RL task, which ensures safe and efficient RL in the redundant null space of higher ranked tasks. We demonstrate the ability of our approach to satisfy constraints during RL trial-and-error by the comparison of our method with baseline algorithm on three simulated tasks. Our experimental setups for two robot systems are shown in Fig. 1.



(a) (b)

Fig. 1. An illustration of the experimental setup for two robot system: (a) a 3-DOF planar robot reaching a target while the robot end effector avoiding three cylinder obstacles and (b) 7-DOF redundant Franka Panda putting a book inside the top layer of the cabinet.

The main contributions of this paper are:

- A null space based hierarchical method that integrates RL algorithms in the safe action space by eliminating constraint violations during RL exploration and enables collision-free high dimensional robotic control tasks in continuous action spaces;
- A novel approach that constructs several constraint phases that can switch between multiple operational spaces and restrict the robot end effector movement with the different stack of constraint tasks;
- Evaluation of three RL algorithms' performance with our hierarchical control framework on three simulated experiments that demonstrates our null space based framework can generalize to different RL algorithms.

This paper is organized as follows: we introduce related works in Section II and background knowledge of hierarchical control and safety constraints in Section III respectively. Next, our null space RL formulation as a lower ranked task is described in Section IV. Experimental results are presented in Section V. Discussion and some possibilities for future work follow in Section VI.

*Funded by Wallenberg Artificial Intelligence, Autonomous Systems and Software Program (WASP).

[1]Autonomous Mobile Manipulation Lab (AMM), Örebro University, Sweden.

978-1-6654-1213-1/21/$31.00

II. Related Work

RL for robotics is categorized into the problem of a continuous action and state space. Safety during exploration is one of the main challenges and safe RL [12], [13], [14] is a significant branch of RL in continuous high dimensional tasks.

One approach is to incorporate a constraint term into RL which results in Constrained Markov Decision Process (CMDP) [15], where agents must satisfy auxiliary constraint costs. However, this kind of constrained RL approach is still not effective for continuous high dimensional control tasks. Trust region methods [10] ensure safe exploration by limiting policy updates between iterations in terms of Kullback-Leibler (KL) penalty. Although trust region methods can reduce the unintended exploration, it is hard to determine the best update step and the agent still might explore harmful behaviors. Constrained Policy Optimization (CPO) [16] approximately enforces the constraints in a trust region of every policy update. In comparison, our algorithm takes advantage of a hierarchical control framework to satisfy constraints during exploration process and safe exploration is guaranteed in the null space of the prioritized tasks.

A recent method trains the policy for constraint as an advisor of the actor [17], which is together denoted as Actor-Advisor method. The advisor learns from the collected experiences to prevent the actor from violating the constraint. However, actor and advisor might tend to induce different regions of the state space, which is sample inefficient. Zhu et al. propose Dynamic Actor-Advisor Programming (DAAP) [18], where actor and advisor are intertwined in policy updates and the advisor is trained simultaneously without any prior knowledge. But it requires two separate sets of reward for minimizing the cost and constraint violation.

Constraint-aware learning by demonstration has also been shown to perform well on the robotic system, in which the task or the constraint is learned first and a null space policy is learned separately [19]. In [8], Armesto et al. propose a constraint-aware learning method that is divided into two problems, learning the constraint and then learning an acting policy in the null space of the constraint. It is validated on a redundant robot that their method can generalize learned null space policies across different constraints that were not known during training. By comparison, constraints are not learned separately and can be defined in advance for our hierarchical control framework.

The prior work constrains the agent to explore in Safe-To-Explore-State-Spaces (STESS) [4], which decomposes a robotic skill into prioritized elemental tasks and a normalized Radial Basis Function (RBF) [4] network is used to represent the learning policy. We continue with STESS framework in this paper and further construct several phases for different period constraints. To be specific, constraint phases can be switched when the robot enters different exploration spaces. Our null space based RL framework is validated on REINFORCE [20], Normalized Advantage Function (NAF) [21] and Deep Deterministic Policy Gradient (DDPG) [22].

III. Hierarchical Control and Safety Constraints

In this work, we leverage the prior hierarchical stack-of-tasks (SoT) [23] motion control framework to restrict RL exploration in the null space of higher prioritized tasks ($e.g.$, joint limits and obstacle avoidance). Each task is described by a mapping from joint space to an operational space.

The vectors $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}} \in \mathbb{R}^n$ represent joint configuration, joint velocity and joint acceleration respectively, where n is degree of joint freedoms. According to [24], we define task Jacobians via the derivatives of pre-defined task functions $\boldsymbol{e}(\boldsymbol{q})$. The task evolution is given by

$$\begin{aligned} \dot{\boldsymbol{e}}(\boldsymbol{q}) &= \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}, \\ \ddot{\boldsymbol{e}}(\boldsymbol{q}) &= \boldsymbol{J}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}(\boldsymbol{q})\dot{\boldsymbol{q}}, \end{aligned} \tag{1}$$

where $\boldsymbol{J} \in \mathbb{R}^{m \times n}$ denotes the task Jacobian $\boldsymbol{J}(\boldsymbol{q}) = \frac{\partial \boldsymbol{e}(\boldsymbol{q})}{\partial \boldsymbol{q}}$ and $\dot{\boldsymbol{J}}$ is the corresponding task Jacobian derivative.

A desired performance measure of (1) can be attained via control law $\ddot{\boldsymbol{e}}^*(\boldsymbol{q})$, where we use simple PD controllers of the form

$$\ddot{\boldsymbol{e}}^*(\boldsymbol{q}) = -K_p \boldsymbol{e}(\boldsymbol{q}) - K_d \dot{\boldsymbol{e}}(\boldsymbol{q}), \tag{2}$$

where K_p and K_d are the positive definite gain matrices. In case of only one equality task, finding joint space controls corresponds to solving the following least square Quadratic Program (QP)

$$\arg\min_{\ddot{\boldsymbol{q}}} \frac{1}{2} \|\boldsymbol{J}\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}\dot{\boldsymbol{q}} - \ddot{\boldsymbol{e}}^*\|^2, \tag{3}$$

which can be solved analytically via the pseudo-inverse of $\boldsymbol{J}$. Considering inequality tasks, a more general constraint is given by

$$\ddot{\boldsymbol{e}}(\boldsymbol{q}) \leq \ddot{\boldsymbol{e}}^*(\boldsymbol{q}). \tag{4}$$

If the constraint in (4) is infeasible, it can be formulated as a least squares function by introducing the slack variable $\boldsymbol{w}$ in the decision variables

$$\begin{aligned} &\min_{\ddot{\boldsymbol{q}}, \boldsymbol{w}} \frac{1}{2} \|\boldsymbol{w}\|^2 \\ &\text{subject to} \quad \boldsymbol{J}\ddot{\boldsymbol{q}} \leq \ddot{\boldsymbol{e}}^* - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}} + \boldsymbol{w}. \end{aligned} \tag{5}$$

To form a hierarchical stack of tasks with $p = 1, \ldots, P$ priority levels, we stack all task Jacobians of equal priority p in a matrix $\boldsymbol{A}_p$ and the corresponding upper bounds in a vector $\boldsymbol{b}_p$ to form one constraint of the form $\boldsymbol{A}_p\ddot{\boldsymbol{q}} \leq \boldsymbol{b}_p$ for each hierarchy level, where

$$\boldsymbol{A}_p = \begin{bmatrix} \boldsymbol{J}_1 \\ \vdots \\ \boldsymbol{J}_n \end{bmatrix}, \boldsymbol{b}_p = \begin{bmatrix} \ddot{\boldsymbol{e}}_1^* - \dot{\boldsymbol{J}}_1\dot{\boldsymbol{q}} \\ \vdots \\ \ddot{\boldsymbol{e}}_n^* - \dot{\boldsymbol{J}}_n\dot{\boldsymbol{q}} \end{bmatrix}. \tag{6}$$

To avoid large acceleration, a regularization term $\lambda\|\ddot{\boldsymbol{q}}\|^2$ is incorporated into the objective function

$$\begin{aligned} &\min_{\ddot{\boldsymbol{q}}, \boldsymbol{w}_p} \frac{1}{2}(\|\boldsymbol{w}_p\|^2 + \lambda\|\ddot{\boldsymbol{q}}\|^2) \\ &\text{subject to} \quad \boldsymbol{A}_i\ddot{\boldsymbol{q}} \leq \boldsymbol{b}_i + \boldsymbol{w}_i^*, \quad i = 1, \ldots, p-1 \\ &\qquad\quad \boldsymbol{A}_p\ddot{\boldsymbol{q}} \leq \boldsymbol{b}_p + \boldsymbol{w}_p. \end{aligned} \tag{7}$$

The main idea of the hierarchical framework above is to solve lower prioritized tasks ($e.g.$, policy search) optimally in the null space of higher prioritized tasks. In this way the RL exploration will not violate higher ranked tasks and therefore a safe behavior is guaranteed in the least-square sense.

IV. Efficient Reinforcement Learning with Safety Constraint Control Tasks

We propose a null space based hierarchical method that integrates RL algorithms in the safe action space by eliminating constraint violations during RL exploration. We leverage a hierarchical control framework that can guarantee constraint satisfaction during the robots trial-and-error exploration, where constraints are used to describe safety conditions. We can limit exploratory actions from arbitrary operational spaces to a collision-free subset of the original state space, which is done by decomposing a robotic skill ($e.g.$, putting a book into a cabinet) into prioritized tasks defined in different operational spaces.

A. Problem Formulation

In the framework of reinforcement learning, the agent interacts with the environment which is modeled as a Markov Decision Process (MDP). An MDP is described by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the set of actions, $\mathcal{T}$ denotes the transition probability from state $\boldsymbol{s}_t$ to the next state $\boldsymbol{s}_{t+1}$ under action $\boldsymbol{a}_t$, and $r \in \mathcal{R}$ is immediate reward with that transition. The goal of reinforcement learning problem is to learn the optimal parameters θ^* of the policy π_θ that maximize the expected return

$$G(\theta) = \underset{\tau \sim \pi_\theta}{E}[\sum_{t=0}^{\infty} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1})], \tag{8}$$

τ is the state-action trajectory and $\gamma^t \in [0, 1]$ is the discount rate at time t.

In this paper, the state space consists of a set of the robot's joint positions $\boldsymbol{q}$, velocities $\dot{\boldsymbol{q}}$ and task error functions $\boldsymbol{e}$, the action space is the second derivative of task error functions $\ddot{\boldsymbol{e}}$. The immediate reward is negatively shaped according to current error function values.

B. Reinforcement Learning in Null Space

A hierarchy of control tasks with safety constraints having higher priority than the RL task are defined in advance. We use the framework introduced in the Section III to guarantee safety during learning process by limiting RL exploration action to a collision-free subset of the original action space. We consider model-free RL and create a control task for RL agent in the stack.

RL task is ranked lower than prioritized constraint tasks in the hierarchical controller. According to (6), all desired task behavior $\ddot{\boldsymbol{e}}_\pi^*$ are stacked in the vector $\boldsymbol{b}_p$ of the same hierarchy p and also the RL agent commands the second derivative $\ddot{\boldsymbol{e}}$ of control task in the error space. The hierarchical control framework satisfies all constraints sequentially starting from the highest priority by quadratic programming method [24] and each lower constraint task is solved in the null space of the previous higher ranked constraints that are left unchanged.

We take advantage of STESS [4] to enable safe exploration of lower ranked RL task in the null space of higher ranked tasks.All tasks are solved in the acceleration space and the objective function can be formulated as

$$\begin{aligned} & \min_{\ddot{\boldsymbol{q}}} \lambda \|\ddot{\boldsymbol{q}}\|_2 \\ \text{subject to} \quad & \boldsymbol{A}\ddot{\boldsymbol{q}} \leq \boldsymbol{b}, \\ & \boldsymbol{J}\ddot{\boldsymbol{q}} = \ddot{\boldsymbol{e}}_\pi^* - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}, \end{aligned} \tag{9}$$

where $\lambda \in \mathbb{R}_+$ is a small factor to avoid large accelerations. $\boldsymbol{A}$ is upper constraints matrix and $\boldsymbol{b}$ is the corresponding upper boundary for all higher prioritized constraints. The inequality provides all upper constraints that need to be satisfied by the feasible optimal solution $\ddot{\boldsymbol{e}}_\pi^*$.

Safe RL task has lower priority than constraints in the controller and thus can learn the policy in the remaining collision-free low dimensional search space. However, sometimes it is difficult or even impossible for the agent to learn a skill in a complicated environment only using single set of constraints. We also propose a novel approach that can switch constraint phases when the robot enters different exploration spaces. Multiple phases for constraint can accelerate learning speed for complicated tasks. The null space based RL training is described in Algorithm 1.

Algorithm 1: Training the policy in the null space of higher ranked constraints

Input : Stack of tasks with different priorities
Output: Trained null space policy

```
Initialize the policy π and initialize replay buffer B;
for each episode=1,M do
    Initialize a random noise ϵ for action exploration;
    Start environment and enter constraint phase C_0;
    for each step=1,N do
        Select action a_t ~ π(s_t) + ϵ;
        Execute action a_t and receive reward r_t;
        if constraint phase condition C_i satisfied then
            Switch and enter constraint phase C_i;
        end
        Store transition tuple (s_t, a_t, r_t, s_{t+1}) in B;
        if reach the goal or constraint violated then
            Reset the environment;
        end
    end
    Train and update the policy π' with sampled
     transitions from B;
end
```

Prior to training the policy, a stack of tasks formulated in (7) are defined with different priorities and the policy π is initialized with random weights. The agent starts each episode from a default constraint phase C_0 that restricts the initial movement of the robot. Given the state in each

step, the policy π generates an action $\boldsymbol{a}_t$ and an Ornstein-Uhlenbeck noise [25] is added in the exploration episodes. A reward is received after action execution and each transition $(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{s}_{t+1})$ is stored in the replay memory $\mathcal{B}$. The agent should switch and enter constraint phase C_i if the switching condition is satisfied. The policy is trained at the end of each episode. We validated our null space RL methods including REINFORCE, DDPG and NAF.

V. Experiments

We have evaluated our approach on three simulation tasks in the Gazebo simulator [26]: two redundant reaching tasks and a putting book task. Performance of null space based RL on three tasks is evaluated to show that the hierarchical constraints can lead the agent to learn safely and efficiently. Reaching tasks can be achieved by calculating the remaining distance between the end effector and the object, while the task of putting a book inside a cabinet is devised into two consecutive phases with different exploration constraints. Our implementation is available at `https://github.com/yquantao/panda_demos/tree/develop`.

A. Experimental Setup

Our three experiments are performed on a 3-DOF arm and Franka panda manipulator respectively. In all experiments each robot joint is controlled by a velocity controller and the velocity commands are generated by the desired task behavior $\ddot{\boldsymbol{e}}^*_\pi$ from RL agent. We use PyTorch to train all three RL methods and Adam [27] is the optimizer with a fixed learning rate 1×10^{-3}. Two reaching experiments consist of 200 episodes and putting book task was trained within 100 episodes. In all experiments each episode has 300 steps. Each episode is terminated and restarts from the initial joint configuration if the robot finishes the goal or violates any constraint.

B. Redundant Reaching Tasks

Two redundant reaching tasks are implemented with a planar 3-DOF arm and 7-DOF Franka panda manipulator respectively. Each reaching task is executed with surrounding planes which restrict the action space of the end effector. Reward is shaped as the addition of proportional distance between the end effector and the target point and the error value of constraint violation

$$r = -\alpha_0 \times \|\boldsymbol{p} - \boldsymbol{p}_{goal}\|_2 - \alpha_1 \times e, \tag{10}$$

where α_i is a positive proportional number, $\boldsymbol{p}$ and $\boldsymbol{p}_{goal}$ are positions of the robot end effector and the target point respectively, e is the absolute error function value of the constraint that is violated. If the end effector reaches the target point within 0.02m in our experiment, a positive large reward will be assigned to the agent.

To ensure safe exploration, some prioritized constraints are set in the controller. Table I lists all constraints for the 3-DOF planar reaching task, including joint limits, the end effector collision avoidance with robot body, surrounding planes and 3 cylinder obstacles. For 7-DOF reaching task similar prioritized constraints are set. RL task has lower priority 2 in the hierarchical controller [24] and the generalization of the proposed null space learning framework is validated with three RL algorithms: REINFORCE, DDPG and NAF.

TABLE I
Constraints set for the first experiment

Constraint	Description	Priority
joint1 limit	limits for $q_0, \dot{q}_0, \ddot{q}_0$	0
joint2 limit	limits for $q_1, \dot{q}_1, \ddot{q}_1$	0
joint3 limit	limits for $q_2, \dot{q}_2, \ddot{q}_2$	0
self collision-free	end effector outside of robot body	0
back plane	-0.6m in y direction of world frame	1
front plane	0.6m in y direction of world frame	1
left plane	-0.6m in x direction of world frame	1
right plane	0.6m in x direction of world frame	1
3 obstacles	end effector outside of 3 cylinders	1

The comparison of cumulative reward for 3-DOF reaching task using null space based RL methods is shown in Fig. 2(a), where learning curves are plotted with mean and variance of three independent runs with the same set of parameters. It presents that the null space NAF method achieves the reward more efficiently and the average cumulative reward is higher than the null space DDPG and REINFORCE. As comparison, Fig. 2(b) shows the training results of three baseline methods. We can see that all three baseline methods learn slowly without our null space based higher prioritized constraints.

The hierarchical controller ensures the elimination of constraint violations that have higher priorities than RL task. The constraint violation rates of two reaching tasks are shown in Table. II and the error threshold of constraint violation is 0.02m. In the first reaching task only one violation occurred with DDPG out of 200 episodes, while three baseline RL methods violate constraints nearly all the time. It can be seen that our null space RL framework is capable of reducing constraint violations significantly.

Training reward with three null space RL methods on Franka reaching task is shown in Fig. 2(c). Null space based NAF and REINFORCE can learn the task efficiently, by comparison DDPG learned quickly at the beginning but oscillated after 75 episodes. When our null space framework is removed, no baseline methods can learn the skill successfully. Same as the first reaching experiment, our null space based RL methods reduced constraint violations significantly compared with 100% violation rate of the baseline without higher ranked constraints depicted in Table. II. Especially null space based NAF decreased the violation rate from 100% to 1.0%.

TABLE II
Constraint violation rate for reaching tasks

Method	3-DOF reaching (%)		Franka reaching (%)	
	Null space RL	Baseline	Null space RL	Baseline
NAF	0.0	99.5	1.0	100.0
DDPG	0.5	98.5	7.0	100.0
REINFORCE	0.0	100.0	3.5	100.0

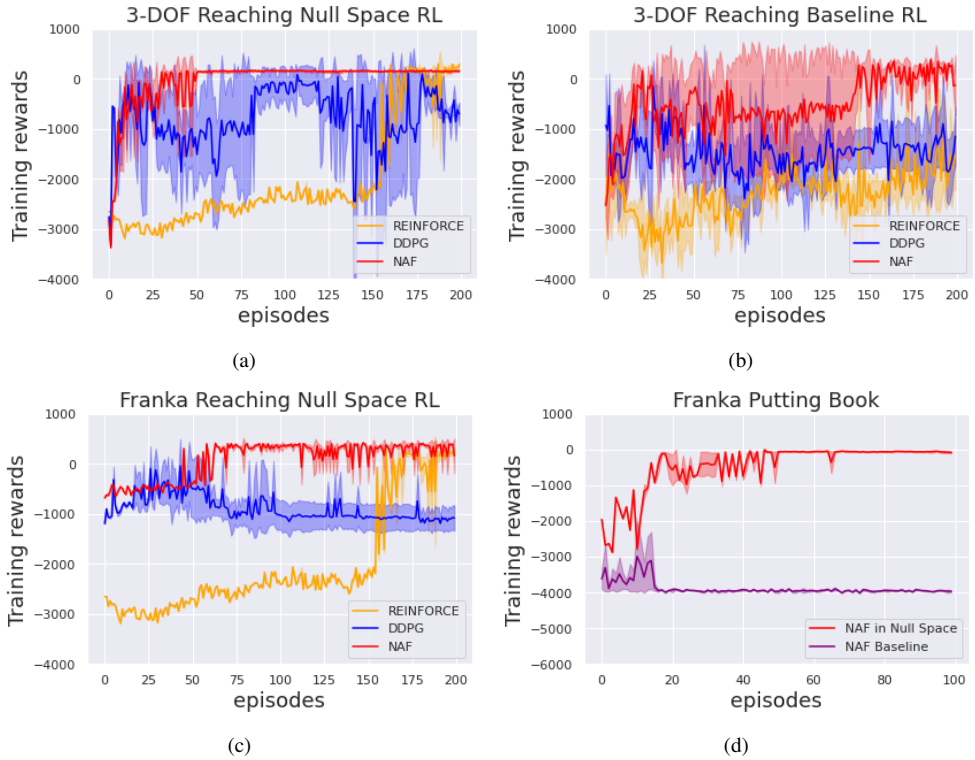
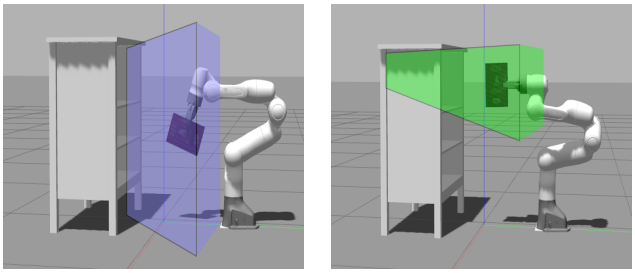


Fig. 2. Comparison of training rewards in three simulated tasks: (a) Training rewards for 3-DOF reaching task with three null space based RL methods; (b) Training rewards for 3-DOF reaching task using baseline RL methods (without null space framework); (c) Training rewards for Franka reaching task with null space based RL methods; (d) Training rewards for Franka putting book task with null space based NAF and NAF baseline method.



(a) (b)

Fig. 3. Illustration of two constraint phases: (a) Constraint phase C_0 that restricts the robot end effector inside the surrounding blue cuboid; (b) Constraint phase C_1 in which all four book corners should be kept inside the corresponding green exploration cuboid.

C. Putting Book Inside Cabinet Task

Based on the observed results achieved on reaching experiments, we chose NAF to validate our approach of multiple phases for different constrained exploration spaces. The goal of our third experiment is to put a book inside the top layer of a cabinet which is shown in Fig. 3.

1) Experimental Setting: To guide the robot to put a book into the cabinet, we devised two phases for different exploration spaces. The robot is set in the first phase C_0 default during which the end effector should be kept inside of the blue cuboid shown in Fig. 3(a). When all four corners of the book are above the top layer plane of the cabinet, the robot will enter the second phase C_1 and constraints are switched. In this phase, all book corners are constrained to stay inside the green cuboid to ensure no collision with the cabinet as shown in Fig. 3(b).

TABLE III

CONSTRAINT VIOLATION RATE FOR DIFFERENT ERROR THRESHOLD IN PUTTING BOOK TASK

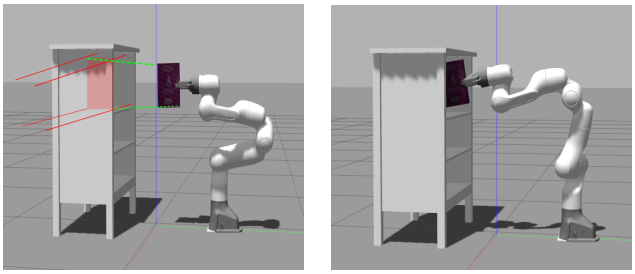
Method	Error threshold 0.02 (%)		Error threshold 0.05 (%)	
	Null space RL	Baseline	Null space RL	Baseline
NAF	2.0	100.0	1.5	100.0

The reward in this experiment is shaped as the sum of the distances between each book corner and a corresponding line drawn with red color in Fig. 4(a) and also the error value of constraint violation:

$$r = -\alpha_0 \times (\delta_0 + \delta_1 + \delta_2 + \delta_3) - \alpha_1 \times e, \qquad (11)$$

where α_i is the positive proportional number, δ_i is the position displacement of each book corner and its corresponding straight line and e is the absolute error value of the violated constraint. As each displacement δ_i tries to minimize the distance between a point and a line, it will allow the robot to place the book anywhere in an upright configuration inside the cabinet top layer. There is a wide range of configurations that would satisfy the goal, which means that the goal is more complex than reaching tasks. One successful example of putting the book inside the cabinet is shown in Fig. 4(b).

2) Results: The training reward for Franka putting book experiment is shown in Fig. 2(d). The robot succeeded in learning the task within 20 episodes efficiently compared with around 50 episodes in both reaching experiments shown in Fig. 2(a) and Fig. 2(c). It validates that the design of multiple constraint phases accelerates learning procedure in this complex task. We recorded constraint violations with



(a) (b)

Fig. 4. Two consecutive snapshot showing a successful task completion of putting a book inside the cabinet: (a) The reward is shaped according to the displacement between each book corner and the corresponding projected line and (b) Franka panda puts the book inside the top layer of a cabinet successfully.

two error threshold 0.02m and 0.05m respectively listed in Table. III. NAF baseline method kept violating constraints and failed in this experiment, while our null space based NAF reduced violation rate to 2.0% and 1.5% for two error threshold values.

VI. Discussion

In this paper, we present an approach for safe and efficient RL for simulated robot systems. A hierarchical control framework is leveraged to decompose target robotic skills into higher ranked tasks (e. g., joint limits and obstacle avoidance) and lower ranked RL task. By encoding prior knowledge in form of constraints, our method ensures safe RL exploration and RL algorithms can be learned in the null space of prioritized constraint tasks. Meanwhile, due to the restricted action space and multiple constraint phases, learning efficiency is improved. We evaluated our approach by simulated reaching point and putting book tasks demonstrating that null space based RL is able to learn high dimensional control tasks safely and efficiently.

In our last experiment the switching conditions of constraint phases are defined by hard coding manually. In future work we plan to look into letting the agent learn these conditions. Also how to transfer the learned model from simulation to the real physical system is another potential future work.

References

[1] W. Yuan, K. Hang, D. Kragic, M. Y. Wang, and J. A. Stork, "End-to-end nonprehensile rearrangement with deep reinforcement learning and simulation-to-reality transfer," *Robotics and Autonomous Systems*, vol. 119, pp. 119–134, 2019.

[2] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 1–8.

[3] 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 *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.

[4] J. Lundell, R. Krug, E. Schaffernicht, T. Stoyanov, and V. Kyrki, "Safe-to-explore state spaces: Ensuring safe exploration in policy search with hierarchical task optimization," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2018, pp. 132–138.

[5] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, "Learning-based model predictive control for safe exploration," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 6059–6066.

[6] P. Abbeel and A. Y. Ng, "Exploration and apprenticeship learning in reinforcement learning," in *Proceedings of the 22nd international conference on Machine learning*, 2005, pp. 1–8.

[7] O. Mees, M. Merklinger, G. Kalweit, and W. Burgard, "Adversarial skill networks: Unsupervised robot skill learning from video," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 4188–4194.

[8] L. Armesto, J. Moura, V. Ivan, M. S. Erden, A. Sala, and S. Vijayakumar, "Constraint-aware learning of policies by demonstration," *The International Journal of Robotics Research*, vol. 37, no. 13-14, pp. 1673–1689, 2018.

[9] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 297–330, 2020.

[10] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, 2015, pp. 1889–1897.

[11] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[12] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[13] A. HasanzadeZonuzy, A. Bura, D. Kalathil, and S. Shakkottai, "Learning with safety constraints: Sample complexity of reinforcement learning for constrained mdps," *arXiv preprint arXiv:2008.00311*, 2020.

[14] S. Miryoosefi, K. Brantley, H. Daumé III, M. Dudík, and R. Schapire, "Reinforcement learning with convex constraints," *arXiv preprint arXiv:1906.09323*, 2019.

[15] E. Altman, *Constrained Markov decision processes*. CRC Press, 1999, vol. 7.

[16] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," *arXiv preprint arXiv:1705.10528*, 2017.

[17] H. Plisnier, D. Steckelmacher, D. M. Roijers, and A. Now, "The actor-advisor: Policy gradient with off-policy advice." *CoRR*, vol. abs/1902.02556, 2019. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1902.html#abs-1902-02556

[18] L. Zhu, Y. Cui, and T. Matsubara, "Dynamic actor-advisor programming for scalable safe reinforcement learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 10 681–10 687.

[19] L. Armesto, J. Bosga, V. Ivan, and S. Vijayakumar, "Efficient learning of constraints and generic null space policies," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1520–1526.

[20] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[21] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," in *International Conference on Machine Learning*, 2016, pp. 2829–2838.

[22] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[23] T. Stoyanov, R. Krug, A. Kiselev, D. Sun, and A. Loutfi, "Assisted telemanipulation: A stack-of-tasks approach to remote manipulator control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–9.

[24] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The International Journal of Robotics Research*, vol. 33, no. 7, pp. 1006–1028, 2014.

[25] G. E. Uhlenbeck and L. S. Ornstein, "On the theory of the brownian motion," *Physical review*, vol. 36, no. 5, p. 823, 1930.

[26] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, pp. 2149–2154.

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980