Safe Reinforcement Learning of Dynamic High-Dimensional Robotic Tasks: Navigation, Manipulation, Interaction

Puze Liu1, Kuo Zhang1, Davide Tateo1, Snehal Jauhri1, Zhiyuan Hu1, Jan Peters1−4 and Georgia Chalvatzaki1−3

Fig. 1: Top: Sequence of real-world human-robot interaction. The human casually looks over the phone ignoring the robot that tries to deliver safely a cup of water. While the human reaches for an orange, the robot smoothly avoids collisions, maintaining the glass in an upward position. Bottom: The simulated digital twin of the real scene that illustrates the signed distance fields of 0.1m (red points) of the human, the robot, and the table, that are used as constraint models in our Safe Reinforcement Learning algorithm.

Abstract—Safety is a fundamental property for the real-world deployment of robotic platforms. Any control policy should avoid dangerous actions that could harm the environment, humans, or the robot itself. In reinforcement learning (RL), safety is crucial when exploring a new environment to learn a new skill. This paper introduces a new formulation of safe exploration for robotic RL in the tangent space of the constraint manifold that effectively transforms the action space of the RL agent for always respecting safety constraints locally. We show how to apply this approach to a wide range of robotic platforms and how to define safety constraints that represent dynamic articulated objects like humans in the context of robotic RL. Our proposed approach achieves state-of-the-art performance in simulated high-dimensional and dynamic tasks while avoiding collisions with the environment. We show safe real-world deployment of our learned controller on a TIAGo++ robot, achieving remarkable performance in manipulation and human-robot interaction tasks.

I. INTRODUCTION

Safe deployment of general-purpose robotic systems in the real world is an overarching goal of safe learning methods [1]. We envision robots learning complex high-dimensional tasks in dynamic, unstructured environments following the paradigm of Deep Reinforcement Learning (RL) [2]. In the standard RL setting, the agent interacts with an environment of unknown dynamics, collecting experience for learning a suitable behavior (policy). The exploration of a Markov Decision Process (MDP) for collecting experience can lead to hazardous situations that can damage the robot or the environment [3]. In this work, we investigate the problem of safe exploration in Deep RL for different robotic tasks with high-dimensional state and action spaces, such as robotic manipulation. We also focus on environments that can dynamically change, e.g., due to the dynamic behavior of a human in a Human-Robot Interaction (HRI) setting.

Safe learning for control and RL is a field of increasing interest in light of the different application areas where autonomous systems should operate [4]–[7]. Three main approaches for SafeRL are particularly relevant to our work. First is the Safe set approach, whose objective is to keep the agent in the set of states considered safe [3], [8]–[13]. The Safe set approaches rely either on a predefined safety set and backup policies or, starting from an initial safe policy, expand the safe set during learning. The second line of work formalizes the problem as a Constrained Markov Decision Processes (CMDP) [4]. The objective of CMDP is to learn an optimal policy where the expected (discounted) cumulative constraint violation does not exceed a threshold. To solve the problem under this formulation, many different approaches have been developed based on Lagrangian optimization [14]–[18], or using an additional reward signal to penalize constraint violations [19], [20]. More recent approaches [21], [22] use state augmentation, which transforms the CMDP problem into an unconstrained one by incorporating the constraint information into the policy.
Safe exploration methods address the safety problem at each step when interacting with the environment [23]. In this setting, we define as safe the behavior of an agent that complies with a set of constraints. These constraints are usually predefined, approximated, learned, or inferred from an initial safe policy. The algorithm directly samples a safe action or modifies one to enforce constraint satisfaction. These approaches are based on either Lyapunov functions [7], [24], [25], Hamilton-Jacobi reachability [26], Control Barrier Functions [27], [28], constraint linearization [29], [30] or planning [31], [32]. While some of these methods have already proven to be effective in some controlled tasks [26], [28], the objective of this work is to lay the foundations for extending the applicability of safe exploration techniques to the complexity of real-world constraints for safe robotic RL.

One of the most promising ways to impose safe learning is by Acting on the TAgent Space of the COnstraint Manifold (ATACOM) [30], a method that effectively maps the action space of an RL agent to the tangent space of the constraints, i.e., ensuring local constraint satisfaction. To allow such approach to be effectively used for safe exploration in RL of high-dimensional robotic tasks of different complexity, like navigation, manipulation, HRI, we extend ATACOM towards three key directions. (a) We generalize ATACOM to nonlinear affine control systems, describing a wide range of robotic systems. (b) We show how to define complex constraints that involve the uncontrollable environmental state in ATACOM to allow the agent to explore safely even with dynamic objects in the domain. (c) We provide an in-depth discussion about the tangent space of the constraint manifold and show how to increase numerical stability.

Thanks to the generalized formulation of ATACOM, we can transform the action space of RL agents for acting on the constraint manifold of both hand-designed (e.g., joint limits) and learned constraints (e.g., for avoiding contact with humans), resolving various classes of constraints for different tasks (e.g., differential drive constraint), while ensuring the deployability of the learned policy at test time, since ATACOM is always active. We provide experimental evaluations on different robotic simulated tasks for learning navigation, manipulation, and HRI, and we show the superior performance of the generalized ATACOM agent in terms of safety and task success-rate against representative baselines from the safe exploration literature. Crucially, we demonstrate the real-world applicability of ATACOM in manipulation and HRI tasks with complex learned constraint manifolds (e.g., human manifold in Fig. 1), showcasing the ability to preserve local safety during deployment without significant performance losses due to sim-to-real gaps.

Related Work in Safe Learning for Robotics

Both safety and adaptability concern all robotic tasks, from manipulation [33], [34], & navigation [35], [36], to locomotion [37] & HRI [38]–[41]. These two requirements are the main focus of the vast literature on SafeRL [1]. Many different formulations and solutions have been proposed to face this problem, e.g., uncertainty-aware model-based RL in robot navigation tasks [42]–[44], offline learning for finding unsafe zones and recovery policy for a surgical robot is proposed in [45] and for locomotion in [46]. Many works adopt the idea of adding a safety layer on the RL-policy, that can adapt an unsafe action to a safe one in conjunction with reachability analysis [32], [47]. An optimization layer was used in [48] to train a robot-reaching task in the real world. An ensemble of policies is used in [49], from which the most likely safe policy is transferred to a real robot playing air hockey through episodic interaction. An RL framework that filters suboptimal actions in the domain of HRI is introduced in [50], but was not demonstrated on a real-world task. In this paper, we introduce a generalized framework for safe exploration in RL of robotic tasks that operates on the tangent space of the constraint manifold [30] satisfying both handcrafted and learned constraints [30]. We showcase real-world performance in challenging tasks, like HRI, demonstrating that our framework can enable safe robot learning of various tasks across different application areas.

Problem Statement

Safe Reinforcement Learning (SafeRL) applies to problems modeled as a CMDP [4] defined by the tuple $<S, A, P, \gamma, R, C>$, where $S$ is the state space, $A$ is the action space, $P : S \times A \times S \to [0, 1]$ is the transition kernel, $\gamma \in (0, 1]$ is the discount factor, $R : S \times A \to \mathbb{R}$ is the reward function, and $C := \{c_i : S \to \mathbb{R} | i \in \mathbb{N}\}$ is a set of constraint functions. We approach the problem of safety in RL through the glance of safe exploration that prevents constraint violations throughout the learning process. Therefore, we formalize our problem as follows,

$$\max_{\pi} \mathbb{E}_{\tau \sim \pi} \sum_{t=0}^{T} \gamma^t r(s_t, a_t)$$

s.t. $c_i(s_t) \leq 0, \quad i \in \{1, 2, ..., N\}, \quad t \in \{0, 1, ..., T\}$

where $\tau = \{s_0, a_0, \ldots, s_T, a_T\}$ is the trajectory under policy $\pi$, and $a_t \sim \pi(\cdot, s_t)$ is the action sampled from policy $\pi$. The objective is to maximize the discounted cumulative reward, while satisfying all constraints at each step.

II. NOVEL FORMULATION OF ATACOM FOR MOBILE ROBOTS AND MANIPULATORS

In this section, we provide a novel and more general formulation of the ATACOM method. Our proposed formulation allows applying ATACOM for learning a wide variety of mobile robotics and manipulation tasks. Moreover, we handle some critical aspects of the original ATACOM, improving its numerical stability, learning performance, and safe-space structure. Finally, we show how to model complex real-world collision-avoidance constraints using learned Signed Distance Function (SDF)s, that allow safe learning of complex tasks, as in the domain of HRI. We first introduce the original design of ATACOM. Then, we introduce our reformulation to it that allows its generalization to a broader class of problems.

A. Original ATACOM Formulation

ATACOM [30] is a method for safe exploration in the tangent space of the constraints’ manifold. It converts the
constrained RL problem to a typical unconstrained one, while handling both equality and inequality constraints. This method allows us to utilize any model-free RL algorithm, while maintaining the constraints below a designated tolerance. In ATACOM, the state space $S$ is separated into two sets, the controllable state space $Q \subset \mathbb{R}^n$ and the uncontrollable state space $X \subset \mathbb{R}^m$, i.e., $s = [q^T x^T]^T \in \mathbb{R}^{n+m}$. ATACOM prescribes that all $k$ constraints are defined on the controllable variable $c(q) \leq 0$, where $c : \mathbb{R}^n \rightarrow \mathbb{R}^k$ is differentiable. ATACOM constructs a constraint manifold by introducing the slack variable $\mu \in \mathbb{R}^k$ into the constraint

$$M = \{(q, \mu) : c(q, \mu) = c(q) + \frac{1}{2} \mu^2 = 0, q \in Q, \mu \in \mathbb{R}^k\}$$

(1)

The tangent-space bases of the constraint manifold are determined by computing the null space $N(q, \mu) \in \mathbb{R}^{(n+k) \times n}$ of the Jacobian matrix $J(q, \mu) = \left[ \frac{\partial}{\partial q} c(q, \mu)^\top, \frac{\partial}{\partial \mu} c(q, \mu)^\top \right]^\top \in \mathbb{R}^{k \times (n+k)}$. To simplify the notation, we use $\bar{c}$, $N$, and $J$, without explicitly writing the dependency on the input. The velocity of the controllable state can be determined by

$$\begin{bmatrix} \dot{q} \\ \dot{\mu} \end{bmatrix} = Na - K_c J^\dagger \bar{c}$$

(2)

with the action $a = n(q, x)$ sampled from the policy. The second term on the right-hand side, with the pseudoinverse of the Jacobian $J^\dagger$ and the gain $K_c$, is the error correction term that forces the agent to stay on the manifold, and it is necessary when using time discretization.

### B. Safe Exploration with Generalized ATACOM on Nonlinear Affine Control Systems

The original formulation of ATACOM assumes a holonomic system, i.e., it assumes that we can set an arbitrary derivative of each generalized coordinate describing the mechanical system. However, many robotics systems of interest are subject to non-holonomic constraints, i.e., non-integrable constraints, such as the differential drive and the bicycle model, which prevent imposing arbitrary velocities or accelerations on the system’s state variables.

To extend the applicability of ATACOM to a broader class of systems, we reformulate it for nonlinear affine control systems [51]. In this setting, we assume that the system’s velocity of the generalized coordinates can be expressed as

$$\ddot{q} = f(q) + G(q) a$$

(3)

with the control action vector $a$, and two arbitrary (nonlinear) vector functions of the current state variable $f(q), G(q)$.

Following the original ATACOM derivation, we consider the constraint with uncontrollable state, $c(q, x) \leq 0$. The constraint manifold can be defined as $M = \{(q, x, \mu) : c(q, x, \mu) = 0\}$. Assuming that the velocity of the state variables $\dot{x}$ involved in the constraints are known or estimated, and using the dynamical system in Eq. (3), we write the time derivative of the constraint function $c(q, x, \mu)$ as

$$\frac{\partial}{\partial t} c(q, x, \mu) = J_q q + J_x \dot{x} + J_\mu \mu$$

$$= J_q f(q) + J_q G(q) a + J_x \dot{x} + J_\mu \mu$$

$$= F(q, x, \dot{x}, \mu) + J_{\mu} a + J_{\mu} \mu,$$  

(4)

with $J_q = \frac{\partial}{\partial q} c(q, x, \dot{x}, \mu)$, $J_x = \frac{\partial}{\partial x} c(q, x, \mu)$, and $J_\mu = \frac{\partial}{\partial \mu} c(q, x, \mu)$ the Jacobian matrices w.r.t. the $q$, $x$, and $\mu$ variables respectively, $F(q, x, \dot{x}, \mu) = J_q f(q) + J_x \dot{x}$, and $J_G = J_q G(q)$. Again, we drop the explicit dependency of the variables $q$, $\mu$ and $x$ to simplify the notation.

We can now compute the safe action by imposing zero velocity of constraint violation. Setting the right-hand side of Eq. (4) to 0 and solving for $a$ and $\mu$ we obtain

$$\begin{bmatrix} a \\ \mu \end{bmatrix} = N(q, \mu) a - J(q, \mu) \bar{c},$$

(5)

where $J(q, \mu)$ is the concatenation of the $J_G$ and $J_\mu$ matrices, and $N(q, \mu)$ is the null space of $J(q, \mu)$. As done in ATACOM, it is also possible to add an error correction term $-K_c J^\dagger \bar{c}$ to the applied control action computed in (5): this term makes the system more responsive and able to deal with equality constraints. By the special choice of slack variable function discussed in Section II-C, we can ensure $J_G(q, \mu)$ is full rank and invertible when $\mu$ is well-defined. We will introduce how the null space matrix is obtained in Section II-C.

With the extended formulation in Eq. (5), we can compute the control action for a wide variety of common robotics kinematics. As an example, we will consider the differential drive kinematics, for which the mobile robot can only move forward/backward along the heading direction and rotate around its center axis. The differential drive kinematics can be cast as a nonlinear affine control system as follows

$$q = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}, \quad f(q) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad G(q) = \begin{bmatrix} \cos \theta & 0 \\ \sin \theta & 0 \\ 0 & 1 \end{bmatrix}, \quad a = \begin{bmatrix} v \\ \omega \end{bmatrix}$$

(6)

with the Cartesian coordinates $x$ and $y$, the current yaw angle $\theta$, the angular velocity $\omega$, and the speed $v$ along heading direction. Given this definitions, it is easy to derive a safe control action using Eq. (5). A similar derivation holds for other kinematics models, e.g. the bicycle kinematics.

Our newly proposed formulation presents a clean and general way to handle systems controlled in velocity/acceleration/jerk or any other arbitrary derivative: this objective can be easily achieved by adding all non-controlled derivatives as state variable $q$ and defining appropriately the $f(q)$ and $G(q)$ functions.

### C. Robust Tangent Space Bases

An ATACOM agent explores the tangent space of the constraint manifold. Therefore, obtaining smooth and consistent bases of the tangent space is essential for training the RL policy. The original method [30] constructs the constraint manifold by introducing the slack variables in quadratic form and determines the unique tangent space bases by QR decomposition of the Jacobian matrix and Reduced Row Echelon Form (RREF). However, this approach may derive inconsistent bases and suffers numerical stability issues. In this work, we further investigate the numerical problems and introduce a new type of slack variable and a new way of determining the tangent space bases that keep the consistency.
a) Projected Tangent Space Bases: The tangent space bases are usually determined by computing the null space of the Jacobian matrix using QR [52] or SVD [53] decomposition. One desired property is to have continuously varying tangent space bases. However, there is no continuous function that generates the null space [54]. QR/SVD-based methods use the Householder transformation, which applies a sign function \( \text{sgn}(\cdot) \) during the Tridiagonalization. The null space bases can flip the direction when a diagonal element changes sign. However, [54] proved the continuity of the projection-based method to generate tangent space bases under certain conditions. Following their idea, the projected null space of the Jacobian matrix is determined by

\[
N = (I - J^T J)^{-1} J Z \tag{7}
\]

where \( J \) is the Jacobian of the constraint manifold, \( Z = [I, 0]_k^T \in \mathbb{R}^{(n+k) \times n} \) are the augmented bases that combine the normalized action-space bases \( I_0 \in \mathbb{R}^{n \times n} \) with \( 0_0 \in \mathbb{R}^{n \times k} \). Eq. (7) projects the action bases onto the tangent space of the constraint manifold. Notice that the first \( n \) dimensions of the augmented state correspond to the original action space, and that the project bases are not orthogonal.

b) The SoftCorner Slack Variable: In the original manifold construction, the mapping between the constraint \( c(q) \) and \( \mu \) in Eq. (1) is not unique, causing an ambiguity in the tangent space bases. As an example, Fig. 2 (right) illustrates the constraint manifold \( q + \mu^2 = 0 \) for the inequality constraint \( q < 0 \), where the arrow shows the tangent space basis. The basis of the tangent space at configuration \( q_0 \) leads to an opposite direction along the \( q \)-axis because the slack variable \( \mu \) is different, as shown by the points \( (q_0, \mu_1) \) and \( (q_0, \mu_2) \). However, as the slack variable is not observed by the agent, this inconsistency of the basis will cause failures during training. Instead, we can formulate a bijective mapping between the original constraint and the slack variable, such as the exponential form \( \exp(\beta \mu) \). Here, \( \beta \) is a positive scalar controlling how strong the original action space shrinks as the constraint function approaches the limit: lowering the \( \beta \) parameter makes the action space more sensitive to the value of the constraint, as the green curve shown in Fig. 2 (left). However, this formulation morphs the action space even in states that are far away from constraint violations, as shown in Fig. 2 (right). To avoid this issue, we introduce the SoftCorner slack variable parametrization

\[
e(q, x) + \frac{1}{\beta} \log(1 - \exp(\beta \mu)) = 0, \tag{8}
\]

where \( \beta \) here has the same interpretation as in the exponential parametrization. In addition, the SoftCorner slack variable maintains the original action space when the states are far from violating the constraint. This slack definition is not sensitive to the metric spaces in which the constraints are defined, such as the combination of the joint space and Cartesian space constraints.

D. Collision Avoidance with Learned SDFs

Collision avoidance constraints are formulated as maintaining a sufficient distance margin between two objects. Typical approach approximate obstacle with primitive shapes, such as spheres. However, this approach is not feasible for complex or dynamic shapes, such as a shelf or a human, as the computation load grows quadratically with the number of primitives. The SDF is a prominent representation for expressing distance w.r.t. a given surface by defining a function that precomputes the distance of an arbitrary query point in the Cartesian space. As SDFs provide a smooth differentiable function, we will discuss how to employ them together with ATACOM.

We rely on Regularized Deep Signed Distance Fields (ReDSDF) [55], which approximates the distance fields of objects with complex shapes or even articulations, such as humans or robots, and provides distances in a regularized form. ReDSDF uses a neural network to approximate the distance of a query point \( p \) w.r.t the center of the articulated object, specified by the joint configuration \( q_o \). Unlike DeepSDF [56] that focuses on reconstructing the object’s surface as close as possible to the zero level-set, the training in ReDSDF integrates an intuitive distance inductive bias, allowing the learning of distance fields at any scale.

The ReDSDF approximates the closest distance of a query point \( p \in \mathbb{R}^3 \) in Cartesian space w.r.t the object. To integrate the collision avoidance constraints in ATACOM, we define multiple Point of Interest (PoI) located at relevant positions of the robot. We can compute the Cartesian position of the PoIs \( p_i \) given a robot configuration \( q \) using forward kinematics \( p_i = FK_i(q), \, i \in [1, 2, ..., N] \). We formulate the collision avoidance constraints as

\[
c_i(q, q_o) : \delta_i - d(p_i(q), q_o) \leq 0, \quad i \in (0, 1, ..., N), \tag{9}
\]

where \( \delta_i \) are thresholds assigned for each PoI and \( d(p_i(q), q_o) \) is the distance computed by the ReDSDF model. A key advantage of using ReDSDF is that we can compute the gradient w.r.t the robot configuration \( q \) as \( \nabla q c_i = -\nabla p_i \cdot \nabla q p_i \), with \( \nabla p_i \) the gradient of the distance field and \( \nabla q p_i \) the Jacobian of the PoI w.r.t the robot configuration, which can be computed based on the adjoint matrix [57]. Note that we don’t include the velocity for the distance constraint in (9) as the velocity of the obstacles is considered in (5).
III. EXPERIMENTAL EVALUATION

To evaluate the performance of generalized ATACOM, we designed a series of tasks with different complexity. First, we designed static goal-reaching tasks in which the robot has to avoid collisions with objects of different geometric complexity, e.g., a table and a shelf, as shown in Fig. 3. We then designed two dynamic tasks for navigation and for HRI. In the robot navigation scenario, we assume a differential-drive mobile robot and test the performance of ATACOM in respecting the control constraints while avoiding collisions with a blindly moving mobile robot. In the HRI task, we consider a shared-workspace scenario in which a human moves dynamically while the robot attempts to deliver a cup appropriately to the desired goal location without colliding with the human or “spilling” the content of the cup. In the following, we provide our empirical evaluation against baselines. All algorithms and environments are implemented within the MushroomRL framework [58]; technical and implementation details can be found on the website https://irosalab.com/saferobotrl/.

A. Manipulation Tasks

The first two experiments are goal-reaching tasks in the proximity of the obstacles, defined as a table `TableEnv` or a shelf `ShelfEnvSim` (Fig. 3). To ensure safety, we define 9 query-PoIs distributed along the links of the moving arm for computing the distance constraints with ReDSDF w.r.t. the tables or shelf. We also define distance constraints for avoiding the ground, self-collisions, and joint limits. Both the `TableEnv` and the `ShelfEnvSim` contain 34 constraints. The reward is defined as $r(s_t, a_t) = -\rho_d d_{goal} - \rho_\angle d_{\angle} - \rho_a ||a_t|| + \mathbb{I}_s$, with the distance to the goal position $d_{goal}$ and orientation $d_{\angle}$, the action penalty $||a_t||$, the scaling factor $\rho_d, \rho_\angle, \rho_a$, and a task indicator $\mathbb{I}_s$. The task indicator encourages the robot to stay on top of the table or inside the shelf. For the manipulation task, we apply velocity control $u = \dot{q}$ on the joints with control frequency 30 Hz. Each episode contains 500 steps. Each episode terminates if a collision is detected, with a penalty of $r_{term} = -1000$.

We compare our approach with three baselines, i.e., vanilla SAC [59], SafeLayer [29], and a hard-coded Linear Attractor (LA). LA-ATACOM applies ATACOM on top of the action computed by LA. Learning the constraint as proposed in [29] is very challenging; thus, we use the constraint functions defined by our approach for a fair comparison. We conducted a hyperparameter search for each task on the learning rates of 5 random seeds, and then, ran 25 seeds with the best hyperparameters. We report the final discounted reward and the distance to the target before episode termination. We also compare the total number of episodes with early terminations due to damage events (collisions or violation of joint limits) and average episode steps throughout 2 million training steps, as shown in Fig. 4a and Fig. 4b. Training is not required in LA and LA-ATACOM, and only the final performance is reported. From the boxplots it is clear that learning with ATACOM high-dimensional manipulation tasks achieves better performance, as LA-ATACOM gets stuck in local minima due to joint limits and complex obstacle shapes.

B. Navigation: Differential-Drive Robot

In the `NavEnv`, we designed a navigation task for the differential-drive TIAGo++ robot (white) that moves in a room while avoiding the Fetch robot (Blue), as shown in Fig. 3. The Fetch robot moves to its randomly assigned target goal, using a hand-crafted policy that ignores TIAGo, and serves as a dynamic obstacle. We aim to train TIAGo to reach its randomly generated goal while avoiding collisions with the Fetch and the walls. The control actions are the
linear and angular velocities of the robot base. We follow a reward structure similar to the one in Section III-A. However, the orientation reward is defined as $-\text{sigmoid}(30(|d| - 0.2)) \cdot \frac{|\angle_{\text{goal}} - \theta|}{\pi}$, with the heading angle of the robot $\theta$ and the yaw angle to the goal $\angle_{\text{goal}}$. As shown in Fig. 4c, the ATACOM-TIAGo can reach the goal with lower error while experiencing significantly fewer collisions. Notably, the performance of SafeLayer in NavEnv is significantly lower compared to the static TableEnv and ShelfEnvSim. Indeed, the SafeLayer approach only corrects the actions when the constraints are violated. To ensure safety in a dynamic task, SafeLayer requires a larger safety threshold, limiting task performance. Instead, ATACOM reduces the feasible action space when approaching the constraint’s boundary. In this task, LA-ATACOM outperforms the learned policy as the moving obstacle prevents local minima and the hard-coded policy is a strong and efficient bias to reach a fixed target.

C. Human Robot Interaction

In the last task, we test our approach in a HRI-Sim environment (Fig. 1) with reward function similar to Section III-A. TIAGo should deliver a cup of water vertically to a target point while the human operates in a shared workspace. Like in TableEnv, we define constraints for avoiding collisions w.r.t the table, the robot, the ground, and the human using the pre-trained ReDSDF. To simulate human motion, we record a human using a motion capture system ($\sim$18,000 data points). During the recording, the human moves near the table arbitrarily without considering the robot. We convert the human motion to the SMPL representation [60]. To train a robust policy, we randomly initialize the human motion at the beginning of each episode. Note that given the inferior performance of the baselines in the previous tasks, we do not compare against them in this far more challenging task to save computational and energy resources. We evaluate the final policy with 10,000 steps, the converged ATACOM policy achieves a final distance to target $0.14 \pm 0.02$m and the number of episodes in which a spill happens is $4 \pm 3.16$. Throughout the training process (2 million steps), the total number of collisions were $9.2 \pm 8.1$. The small number of constraint violation during training confirm the effectiveness of the safe exploration strategy of ATACOM, that ends up learning a safe interaction policy.

D. Real Robot Validation

Given the encouraging results of ATACOM in our simulated results, we transfer our safe policies to the real world*. We create the ShelfEnvReal, in which we place two objects on the shelves of a bookcase (Fig. 3). We use motion capture to perceive the objects’ poses. The robot starts from a random configuration and reaches the two targets sequentially. The maximum reaching time for each target is 16.67s. Once the distance between the robot grasping frame and the object is smaller than 10cm for 2s, we count the task as a success. We conduct 25 trials with different combinations of object placements and initial arm configuration. As seen in our results in Table I, our policy is 100% safe.

<table>
<thead>
<tr>
<th>Target</th>
<th>Final Error (m)</th>
<th>Suc. Rate</th>
<th># Collisions</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.038 $\pm$ 0.010</td>
<td>90%</td>
<td>0</td>
<td>6.58 $\pm$ 1.01</td>
</tr>
<tr>
<td>2</td>
<td>0.047 $\pm$ 0.024</td>
<td>90%</td>
<td>0</td>
<td>6.47 $\pm$ 1.73</td>
</tr>
</tbody>
</table>

*Videos of the real-world experiments can be found in: https://irosalab.com/saferobotrl/

IV. Conclusions

This paper introduces a novel and more general formulation of Safe Exploration in RL by Acting on the TAgent Space of the COnstraint Manifold (ATACOM), i.e., by transforming the agent’s action space by mapping it on the nullspace of the constraints. We extend the original approach to non-linear control affine systems, hence, treating various robotic platforms and a wide variety of tasks, such as manipulation and navigation. Furthermore, we show how to integrate learned constraints with ReDSDF: this allows imposing arbitrary shapes and articulated structures as safety constraints. This flexibility in the constraint definition makes learning safe interactions with humans possible, even when deploying the learned policies in the real world, without sacrificing performance. The proposed method paves the way for possible breakthroughs in the deployment of RL methods in dynamic real-world tasks.
References


