Autonomous Robots (2024) 48:12
https://doi.org/10.1007/s10514-024-10164-6

Guiding real-world reinforcement learning for in-contact manipulation tasks with Shared Control Templates

Abhishek Padalkar[1] · Gabriel Quere[1] · Antonin Raffin[1] · João Silvério[1] · Freek Stulp[1]

Received: 23 August 2023 / Accepted: 20 April 2024 / Published online: 4 June 2024
© The Author(s) 2024

Abstract

The requirement for a high number of training episodes has been a major limiting factor for the application of *Reinforcement Learning* (RL) in robotics. Learning skills directly on real robots requires time, causes wear and tear and can lead to damage to the robot and environment due to unsafe exploratory actions. The success of learning skills in simulation and transferring them to real robots has also been limited by the gap between reality and simulation. This is particularly problematic for tasks involving contact with the environment as contact dynamics are hard to model and simulate. In this paper we propose a framework which leverages a shared control framework for modeling known constraints defined by object interactions and task geometry to reduce the state and action spaces and hence the overall dimensionality of the reinforcement learning problem. The unknown task knowledge and actions are learned by a reinforcement learning agent by conducting exploration in the constrained environment. Using a pouring task and grid-clamp placement task (similar to peg-in-hole) as use cases and a 7-DoF arm, we show that our approach can be used to learn directly on the real robot. The pouring task is learned in only 65 episodes (16 min) and the grid-clamp placement task is learned in 75 episodes (17 min) with strong safety guarantees and simple reward functions, greatly alleviating the need for simulation.

Keywords Guided reinforcement learning · Safe robot control · Robot learning · Constraint-based learning

1 Introduction

The potential of reinforcement learning (RL) in solving high-dimensional, highly non-linear problems is evident from its super-human level performance on Atari games (Mnih et al., 2013) and in mastering Go (Silver et al., 2016). RL can perform at its full potential when skills are learned from scratch, as evidenced by the example of AlphaGo Zero (Silver et al., 2017) (trained from scratch) outperforming AlphaGo (Silver et al., 2016) (pretrained with human games).

In robotics, RL has piqued the interest of researchers by learning intricate and impressive motor skills such as juggling (Ploeger et al., 2020), ball-in-cup (Schwab et al., 2019), in-hand manipulation (Andrychowicz et al., 2020), pick-and-place (Levine et al., 2018), or locomotion over highly variable terrain (Lee et al., 2020a). Although executing episodes on real robots is much more time-consuming than simulating Go games, it is still feasible and plays an important role in the acquisition of complex robot motor skills. However, the requirement for a large number of episodes has limited the application of RL in real-world object manipulation. Learning in simulation and then transferring skills to the real robot is limited by the accuracy of the simulation. Such limitations become even more apparent in tasks involving contacts, which are discussed in detail by Elguea-Aguinaco et al. (2023). In this article, we present a framework which enables learning directly on the real robot, safely and efficiently.

One of the aspects that sets human manipulation skills in daily activities (e.g. pouring a drink, opening a door) apart from the intricate motor skills mentioned before is that these

✉ Abhishek Padalkar
Abhishek.Padalkar@dlr.de

Gabriel Quere
Gabriel.Quere@dlr.de

Antonin Raffin
Antonin.Raffin@dlr.de

João Silvério
Joao.Silverio@dlr.de

Freek Stulp
Freek.Stulp@dlr.de

1 Robotics and Mechatronics Center (RMC), German Aerospace Center (DLR), Münchener Str. 20, 82234 Weßling, Germany

activities often involve interactions with objects that have a dedicated purpose (e.g. bottles, glasses, doors, sockets). Such objects are specifically designed with purposeful interactions in mind, i.e. they *afford* such interactions (Gibson, 1979; Leidner et al., 2012; Khetarpal et al., 2020). We consider it inefficient to use RL to re-learn such interactions—and the constraints arising from them—from scratch in thousands of episodes. What is known should not be learned.

In other words, the robot should not use RL to learn *what to do* with an object—this is already known and intrinsic to the object—but rather *how to do it well*, i.e. efficiently and without making many mistakes. This is especially true for tasks involving contact, such as insertion tasks, where the goal state is known but optimal behaviors (e.g. interaction forces) are difficult both to hardcode and demonstrate. In these cases, specifying partial constraints can greatly reduce the number of episodes needed to learn the task.

This stance raises the question of how such purposeful interactions and constraints involved should be represented and incorporated in the RL process. In our research on shared control (Quere et al., 2020), we have serendipitously discovered that representations for shared control are also ideally suited for guiding RL, for several reasons. First, shared control aims at enabling users to control high-dimensional robotic systems with low-dimensional input commands. For RL, this approach substantially reduces the action space. Second, user preferences vary, and shared control must foster agency and empowerment by providing freedom of movement within the reduced action space. For RL, this means that there is room for exploration within such representations. Third, shared control often limits the range of motion for safety reasons. For RL, this also means that exploration becomes safe. This has the following advantages:

1. Safer learning, as constraints are enforced during learning;
2. Faster learning, as constraints need not be learned;
3. Simplified reward function design, as constraints need no longer be implicitly represented in the reward.

Together, these advantages greatly facilitate skill acquisition on real robots via RL. Indeed, our key message in this article is that representing constraints implicitly in shaped reward functions, can require as much human design effort as explicitly modeling these constraints, but that the latter leads to safer and faster learning.

1.1 Contributions

In this paper, we investigate how RL can be guided by our framework for shared control—Shared Control Templates (SCTs) (Quere et al., 2020), see Sect. 3—thereby making it safer and more efficient. As shown in Fig. 1, we

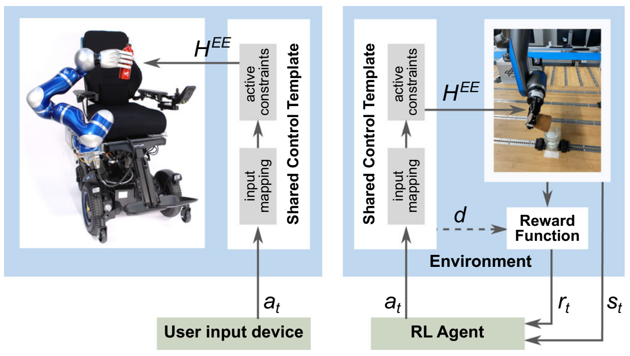


Fig. 1 In previous work, we developed Shared Control Templates (SCTs) to support human users in executing tasks of daily living, such as pouring liquids or opening drawers (Quere et al., 2020). In this context (left illustration), users provide commands to control the robot, which has many degrees of freedom. Efficiency and safety is improved with Input Mappings—which map low-dimensional user commands $\boldsymbol{a}$ to the many degrees of freedom of the robot $\boldsymbol{H}^{EE}$—and Active Constraints—which limit the range of motion, for instance to avoid collisions. In this work, we use SCTs to analogously achieve efficiency and safety in RL (right illustration). We also demonstrate how transition functions in the SCT's Finite State Machine (not depicted above; see Fig. 4 for details) can be used to automatically generate shaped reward functions and further speed up learning. The resulting framework is called *Reinforcement Learning with Shared Control Templates* (RL-SCT). RL-SCT is evaluated on the two tasks illustrated in Fig. 2

replace the human user by an RL agent to complete the task autonomously. Our method, *Reinforcement Learning with Shared Control Templates* (RL-SCT), introduced in Sect. 4, allows us to model task space constraints and knowledge about the motion required for task completion flexibly, learn the complete task model from the robot-environment interactions, and optimize secondary costs associated with quality of the task. The main contributions of this paper are:

1. A framework that leverages task knowledge represented in a shared control method to make reinforcement learning in the real world possible by explicitly constraining both state and actions spaces,
2. an approach to simplify the generation of shaped rewards based on prior information about task progress and completion encoded in SCTs, and
3. showing that applying SCTs to RL leads to more sample efficient and safer learning on a real robot even for tasks involving contact with environment (Fig. 2).

Our approach is validated using a simulated KUKA IIWA robot on a pouring task and DLR's SARA robot on pouring and grid-clamp insertion tasks, the latter requiring contacts with the environment (Sect. 5). Our results show that RL-SCT avoids unnecessary mistakes during the learning process and enables faster learning for the tasks where the object interaction model is fully known. For tasks where only partial interaction modeling is possible, e.g. tasks with contacts such

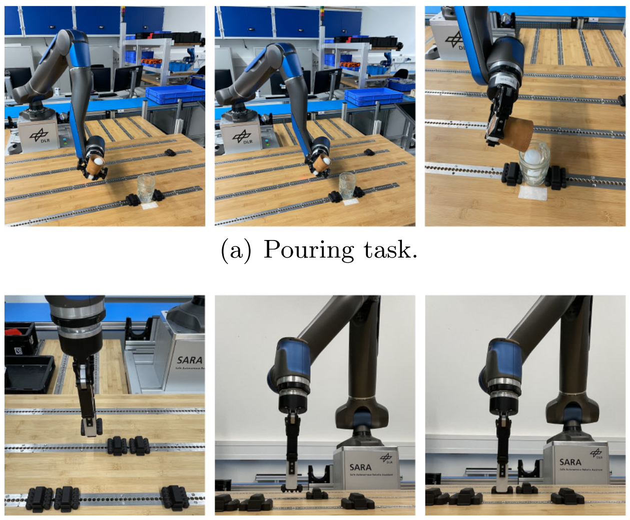


Fig. 2 Safer and more efficient RL with RL-SCT is demonstrated on the SARA robot (Iskandar et al., 2020) on two tasks: pouring and grid-clamp insertion. The latter involves contacts with the environment that are typically difficult to model in simulation. Our approach allows the robot to successfully learn the task while minimizing the interaction forces

as the grid-clamp insertion, we show that RL-SCT makes learning faster, safer—both by avoiding collisions and minimizing interaction forces when they are required to complete the task—and hence possible directly on the real robot.

2 Related work

As contact dynamics are often difficult to model, tasks involving contacts are interesting for model-free RL (Buchli et al., 2011; Kalakrishnan et al., 2011). However, learning in-contact tasks with RL is challenging, as exploration can be unsafe due to collisions, and the number of episodes required to learn a task can be prohibitively high. Simulation and sim-to-real methods resolve these issues, but come with their own set of challenges when simulating in-contact tasks (Salvato et al., 2021). RL-SCT enables RL to be performed directly on the real robot, by ensuring safety through Active Constraints, and improving efficiency by providing low-dimensional action spaces and generating shaped reward functions.

In this section, we first describe RL approaches that are conceptually similar to (components of) SCT-RL. In Sect. 2.5, we then describe learning approaches that have been applied to in-contact manipulation on real robots.

2.1 Safe and efficient reinforcement learning with state-dependent action masking

In Reinforcement Learning, Markov Decision Process (MDP) is formalized as (S, A, T, R), with state space S, action space A, transition function T, and reward function R. To speed up learning, several approaches to reduce and modify these spaces and functions have been proposed, including action masking, action manifolds, and reward shaping.

To improve the safety during exploration and/or to speed up learning, various approaches have been proposed that constrain the actions that can be performed in specific states. An example in the discrete domain is *action masking* (Huang & Ontañón, 2020), where the set of actions A'_{s_i} that is known to be invalid or suboptimal in certain states is excluded from the action space in those states, i.e. $A'_{s_i} \subset A$, where A'_{s_i} denotes the set of masked actions at s_i. As fewer actions need to be explored, action space shaping typically speeds up learning (Kanervisto et al., 2020).

In robotics, similar concepts have been explored to also ensure safety during exploration. Cheng et al. (2019) proposed to use *Control Barrier Functions* (CBFs) (Taylor et al., 2020) to avoid unsafe states for the robot while an RL agent explored the state-action space.

Liu et al. (2022) demonstrate that a constrained RL problem can be converted into an unconstrained one by performing RL on the tangent space of the constrained manifold. Their results obtained in simulation are promising, but the approach is limited to problems where the constraints are differentiable, which restricts its application domain. RL-SCT is free of differentiability assumptions, making it flexible by design.

2.1.1 Affordances in RL

Khetarpal et al. (2020) apply the theory of affordances (Gibson, 1979) to RL, which allows faster learning by action space reduction and precise learning of transition models. First, 'intent' is defined as the desired outcome of an action. Affordances then capture the subset of the state-action space where the intent is achieved. Such subset represents a partial model of the environment. Intents and affordances are a principled two-step procedure for defining state-dependent actions masks. This approach is also noteworthy for combining action masking with a decomposition of the overall policy into different intents and affordances, see also Sect. 2.3. Within RL-SCT approach, each state can be considered to have an intent for a subphase of the task, and the Active Constraints implement the continuous action masking. SCT-RL adds to this the Input Mappings, the automatic generation of shaped rewards, and the representation of subtasks in a finite state machine. The method in (Khetarpal et al., 2020) is demonstrated on grid worlds and continuous 2D planes, and has, to the best of our knowledge, not yet been scaled up to the high-dimensional action spaces of robots.

Affordance-based learning has also been used to generate high-level skill sequences for domestic tasks (Cruz et al., 2014, 2016, 2018). A human guides the agent with instructions adhering to the affordances. In SCT-RL, we rather consider low-level actions that control end-effector positions, for instance for in-contact manipulation.

2.2 Efficient reinforcement learning with action manifolds

An alternative to defining state-dependent action masking/constraints is to define a reduced action space for the entire state space.

Luck et al. (2014) combine policy search and dimensionality reduction in an expectation-maximization framework and use probabilistic PCA for obtaining a latent space during the learning process.

Kolter et al. (2007) present a method that uses a (possibly inaccurate) simulator to identify a low-dimensional subspace of policies. An instance on a real system is learned using these low-dimensional policies using much less data.

Padalkar et al. (2020a, 2020b) proposed a partial task specification in the robot task space using the Task Frame Formalism (Mason, 1981; Bruyninckx & De Schutter, 1996), learning the remaining specification of the task with RL. This method successfully demonstrated a significant reduction in robot-environment interactions when cutting vegetables with a light-weight robot. Although this work ensures safety in the directions in which motion is fully specified, it does not guarantee the safety in the directions explored by the RL policy, a problem addressed in RL-SCT.

Reinhart and Steil (2015) propose a two step solution with a skill memory, an organization of motion primitives in low-dimensional, topology preserving embedding space and policy search leveraging the low-dimensional skill parameterization. Skill parameterization can be predefined or automatically discovered with parameterized self-organizing maps.

He and Ciocarlie (2022) present a framework that discovers a synergy space and learns a multi-task policy that operates on this low-dimensional action space. Learned synergies can be used across multiple manipulation tasks.

In this paper, we refer to action manifolds as *Input Mappings*, a term used in SCTs due to their role in shared control, where user *inputs* (e.g. through a joystick) are *mapped* to robot actions (Dragan & Srinivasa, 2013).

Manifold learning has also been applied to reducing the state space (rather than the action space as described above), for instance using PCA and GPLVM (Bitzer et al., 2010; Tosatto et al., 2021; Curran et al., 2016; Parisi et al., 2017). In DeepRL this is known as *state representation learning* (Raffin et al., 2019), and the automatic learning of the state space representations is the main motivations for using DeepRL. A full overview of this field is beyond the scope of this paper, as our focus in this work is not on learning state space manifolds.

2.3 RL with sequential tasks and/or task decompositions

An early RL approach to representing subtasks is the *options* framework, where the agent uses macro actions that span multiple time steps to represent subpolicies for subtasks (Sutton et al., 1999; Stolle & Precup, 2002). Our work differs in that the subtasks are represented in the SCT, and the SCT is part of the environment, rather than the agent. The RL agent does not represent subpolicies or subtasks internally, as will be highlighted in Fig. 4.

Daniel et al. (2013) propose an approach to learning sequencing motor primitives while simultaneously improving individual motor primitives.

Kroemer et al. (2015) propose to learn a probabilistic multiphase model of the task, a motion primitive for each task phase, and a RL policy for sequencing the motion primitives that use the task model. Although these methods solve multiphase tasks; they do not take constraints in the environment into account, and hence exploratory actions can lead the robot into potential hazardous situations during the learning process. In RL-SCT, we propose to learn a single policy for all phases of the task, whereas the above-mentioned approaches use different motion primitives for different phases.

Krishnan et al. (2019) propose an approach where inverse RL learns subtasks with subgoals and local cost functions from a latent space representation of demonstrations obtained with unsupervised learning. Such latent space and decomposed representation is used to accelerate RL.

Koert et al. (2020) present an approach where a robot learns and improves to combine skills for sequential tasks with human input during learning to accelerate the learning. It is an interactive framework where the human advises the robot on planned high-level action, provides feedback on the outcome of the action, and provides subgoal rewards. This approach uses human-guided RL for solving decision-making problems while choosing predefined low-level policies, whereas we propose to learn control policies with RL that generate inputs for underlying MDP simplified by SCTs by incorporating the task knowledge.

Khetarpal et al. (2020) developed the theory of affordances (Gibson, 1979) for RL agents, which allows faster learning by action space reduction and precise learning of transition models. With intent defined as the desired outcome of an action, affordances capture the subset of the state-action space where the intent is achieved. Such subset represents a partial model of the environment. In RL-SCT, each SCT-state models an intent and defines IM and AC to achieve the intent with state dependent action space reduction.

2.4 Reward shaping

In reward shaping, a sparse terminal reward R is transformed into a dense, immediate reward R' by adding knowledge about the task and its subphases (Ng et al., 1999). It does not change S, A, or T. Shaped rewards are more informative, and thus speed up learning. In this paper, we argue that action space shaping is more effective that reward shaping, and contrary to common belief, often not more difficult to design.

2.5 Reinforcement learning of manipulation tasks with contacts

Zhao et al. (2022) presented an approach of meta-reinforcement learning by encoding human demonstrations for different tasks in a latent space and using the latent space variables to generalize the RL policy for different types of insertion tasks. The policy can be trained offline with data collected from sources like human demonstrations, replay buffers from previous experiments and data collected from hand-coded solutions.

Vecerik et al. (2019) train a neural network to extract features from images during insertion task, with the same neural network being used to compute a binary reward. The features are used as input to the RL policy. Critic and actor are pre-trained to mimic human demonstrations. Davchev et al. (2022) present an approach to learn insertion task by learning a residual policy to support an imitation learning policy learned with Dynamic Motion Primitives. Kozlovsky et al. (2022) presented an approach to learn asymmetric impedance matrices to learn a policy in simulation and then transfer the solution to the real robot. Lee et al. (2020b) combine an RL policy with a model-based solution in the region where the task model is uncertain to learn the insertion task with a binary reward. The poses of the objects used as state are estimated by a vision system. All the above approaches do not include force measurements in the state and do not optimize interaction forces while executing the task. We address this limitation in RL-SCT by introducing interaction forces in the agent state and a secondary cost in the reward to minimize interaction forces.

Apolinarska et al. (2021) applied RL for assembly of timber joints. They use human demonstrations for initializing the policy. The policy is learned in simulation with domain randomization and then transferred to the real robot. Luo et al. (2019) also presented an approach to learn assembly tasks by learning a variable impedance controller. They use end-effector force/torque readings filtered by a low-pass filter and directly inject them in the second layer of the neural network representing the policy in order to provide direct haptic information to the policy. These approaches use interaction force in the state given to the RL policy but do not optimize interaction cost.

Kim et al. (2021) presented a method to learn impedance parameters using RL for insertion tasks and then transferred the learned policy to a real robot which presents a need for fairly accurate simulation. Beltran-Hernandez et al. (2020) proposed a method to learn force controllers on a position-controlled robot using an end-effector force/torque sensor with RL. Both of these approaches use force data in the state and try to optimize interaction forces. Beltran-Hernandez et al. (2020) do not use any method to guide or constrain RL exploration and hence results in collision during the learning.

Table 1 compares the approaches for learning contact tasks to RL-SCT. We address the various shortcomings of this related work in RL-SCT by leveraging pre-specified task knowledge to improve sample efficiency and simplifying reward function design. RL-SCT facilitates direct learning on real robots hence alleviating the need for simulation. Simplified reward functions in RL-SCT allow us to optimize secondary costs, e.g. interaction force costs, right from the beginning of learning, producing significantly lower forces during learning and resulting in learned policies that generate minimal interaction forces with the environment.

3 Shared control templates

In most applications of shared control, the aim is to map low-dimensional input commands to task-relevant motion on a high-dimensional (robotic) system. An example is EDAN ("EMG-controlled Daily AssistaNt") (Vogel et al., 2020), which consists of an electric wheelchair with an articulated arm. EDAN typically takes a 3D input signal extracted from surface electromyography. It maps these inputs to 6D end-effector movements, which are then mapped to the movement of the overall 11 degree-of-freedom system (excluding the degrees of freedom in the hand) through whole-body motion control (Quere et al., 2020).

Performing tasks such as opening doors and pouring liquids with EDAN raises the following challenges: (1) The 6D end-effector action space is too complex for a user to command in unison. A typical user command is at most 3D. (2) There are many constraints that should not be violated, e.g. not tilting a full bottle too much before starting to pour. (3) All tasks consist of multiple phases, e.g. grasp bottle, move towards mug, tilt bottle, pour, etc. (4) The state and action spaces are continuous.

To address these challenges, we have proposed Shared Control Templates (SCTs) (Quere et al., 2020), which consist of several components.

Table 1 Comparison with state-of-the-art approaches on learning in-contact tasks using RL. f_{ext} denotes contact forces with the environment. (D demonstrations, C constraints and residual learning, S pre-trained in simulation)

Criteria	On real robot	Guidance	f_{ext} as state	f_{ext} in reward
(Zhao et al., 2022)	✓	D	–	–
(Apolinarska et al., 2021)	✓	D,S	✓	–
(Vecerik et al., 2019)	✓	D	–	–
(Beltran-Hernandez et al., 2020)	✓	-	✓	✓
(Lee et al., 2020b)	✓	C	–	–
(Kozlovsky et al., 2022)	✓	C	–	-
(Davchev et al., 2022)	✓	C	–	–
(Kim et al., 2021)	✓	S	✓	✓
(Luo et al., 2019)	✓	–	✓	–
(Buchli et al., 2011)	✓	D	–	✓
(Kalakrishnan et al., 2011)	✓	D	✓	✓
RL-SCT (our approach)	✓	C	✓	✓

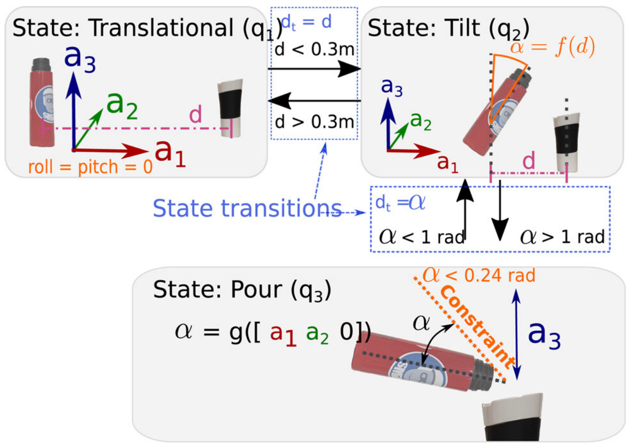


Fig. 3 The Shared Control Template (SCT) for pouring water. Different phases of a task are modeled as different SCT states (q_i) in a Finite State Machine ('Translational', 'Tilt', 'Pour'). In each state, an Input Mapping maps the 3D user input commands a_1, a_2, a_3 to 6D end-effector motions. Active Constraints (shown in orange text) limit the range of motion

Input Mappings (IMs)

In previous work, the user provides commands through electromyography or a joystick (Quere et al., 2020). An Input Mapping converts these user inputs—which in this paper are 3D, following Quere et al. (2020)—to phase-dependent 6D end-effector motions.

In the *Translational* and *Tilt* states in Fig. 3 for example, all 3 inputs a_1, a_2, a_3 are mapped to the 3 translational Cartesian DoFs. In the *Pour* state, a_3 is mapped to translation in Z-direction and the vector $[a_1\ a_2\ 0]$ is mapped to the tilt angle of the thermos via a scalar projection.

Formally, an IM is a function which computes a desired displacement $\Delta \boldsymbol{H} \in \text{SE}(3)$ in task space from an n-dimensional input $\boldsymbol{a}_t$ with $n \leq 6$ at time step t (dropping the subscript i in q_i from now on):

$$\begin{aligned} \text{map}_q : \mathbb{R}^n &\rightarrow SE(3) \\ \boldsymbol{a}_t &\mapsto \Delta \boldsymbol{H}. \end{aligned} \tag{1}$$

The displacement computed from Eq. (1) is then applied on the end-effector pose H_t:

$$\begin{aligned} \text{displace}_q : SE(3), SE(3) &\rightarrow SE(3) \\ \boldsymbol{H}_t, \Delta \boldsymbol{H} &\mapsto \boldsymbol{H}^{\text{im}}_{t+1}. \end{aligned} \tag{2}$$

Active Constraints (ACs)

Active Constraints (Bowyer et al., 2013) constrain the end-effector pose that results from applying an IM. As illustrated in Fig. 3 during the phase *Pour*, the tilt angle α of the bottle is constrained, to avoid excessive pouring. Another example of AC in the *Tilt* phase is the value of the tilting angle being a function of the distance from the target.

Formally, after applying the IM to obtain $\boldsymbol{H}^{\text{im}}_{t+1}$, geometric constraints can be enforced using AC. An AC (Quere et al., 2020) applies a projection of the form,

$$\begin{aligned} \text{project}_q : SE(3) &\rightarrow SE(3) \\ \boldsymbol{H}^{\text{im}}_{t+1} &\mapsto \boldsymbol{H}^{\text{ac}}_{t+1}, \end{aligned} \tag{3}$$

where $\boldsymbol{H}^{\text{ac}}_{t+1}$ is the constrained end-effector pose. This constraint could for example be the arc-circle path traced by the door handle when opening a door or orientation constraints depending on the end-effector position when approaching an object, as in the *Tilt* phase in Fig. 3.

Finite State Machine (FSM) and Transition Functions

Different phases of a task require different IMs and ACs, as illustrated in Fig. 3. For this reason, these phases are represented as states in a finite-state machine (FSM). The FSM triggers transitions from one state to the next by monitoring transition functions, which measure when certain distances $d_t \in \mathbb{R}$ drop below pre-specified thresholds.

During the approach in the first state in the pouring task for instance, d_t is the distance between thermos and mug tip positions at t, i.e. $d_t = ||\boldsymbol{x}_{\text{th},t} - \boldsymbol{x}_{\text{mug},t}||$. In the last state, when actually pouring, the distance is the tilting angle $d_t = \alpha_t$. Both are highlighted as blue rectangles in Fig. 3.

A key aspect of SCTs is that they facilitate task completion through shared control by defining task-relevant IMs and ACs, but the user always remains in control, i.e. determines the speed of movement, the amount of water that is poured, etc.

SCTs have previously been used to automate tasks, where trajectories are generated through local optimization (Bustamante et al., 2021); our aim here is to optimize the overall policy that generates the trajectory.

4 Shared control templates for reinforcement learning

The key insight that we propose is that *components that facilitate human control of the robot through shared control are conceptually very similar to those used to facilitate reinforcement learning*. Our aim is to demonstrate this conceptually and empirically.

Figure 4 shows how SCTs are integrated in the RL agent-environment interface. In this section, we describe how the main components of SCTs—Input Mappings, Active Constraints, and the Transition Functions between the FSM states—are conceptually similar to state-dependent action space shaping (see Sect. 2.1), action manifolds (see Sect. 2.2), and reward shaping (see Sect. 2.4), respectively.

4.1 Action space shaping with SCTs

Using IMs to map 3D user commands to 6D Cartesian commands, the action space A of the MDP is reduced to 3D. Although IMs can accept inputs up to 6D, more than 3 inputs are rarely needed in most tasks (Quere et al., 2020).

The application of AC is conceptually equivalent with continuous state-dependent action masking, in that not all actions defined by the IM have an effect[1].

Typically, in RL, constraints are encoded *implicitly* in the reward function, through reward terms that encourage actions that do not violate them. A direct consequence of such approach is that an agent will need to violate these constraints in order to learn them. In many robotic tasks, violating

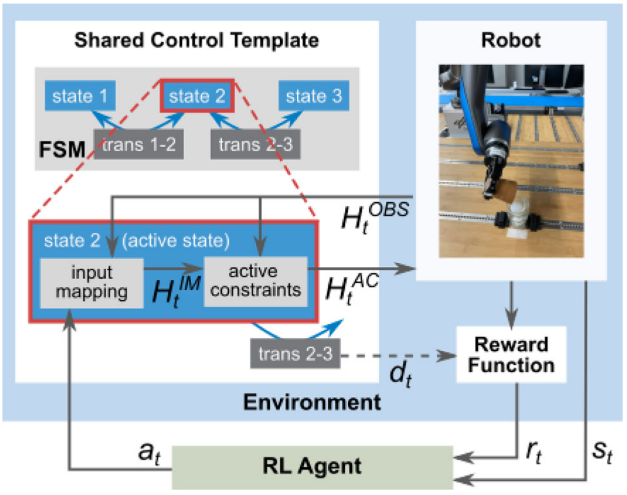


Fig. 4 Agent-environment interface in SCT-RL components. The input a_t to the IM of an SCT is computed by the RL policy, s_t is the state for the RL agent. d_t is the distance function to determine transitions in the FSM; it can be used as a shaped reward function. All time-varying variables are annotated with the same t, in practice the agent-environment loop and the controllers using the end-effector poses H may run at different frequencies

constraints can lead to physical damage. SCTs address the above issues by precluding the robot from violating constraints through the definition of Active Constraints.

4.2 Reward Shaping with SCTs

In RL, the main role of the reward function R is usually to provide feedback about whether the task has been achieved, e.g. was the exit to the maze found, was water poured into the glass, etc. We call this the *primary reward*. Designing a primary reward function is easiest if this reward is sparse (e.g. 1 or 0) and terminal (i.e. r_T^{prim} is given at the end of an episode) (Chatterji et al., 2022). However, this is also the least informative type of reward function.

Primary reward shaping with the SCT transition distance
Reward shaping is the process of redesigning the sparse primary reward function so that it becomes dense and/or immediate, to make the reward function more informative and speed up learning. The process of designing an SCT is similar to that of shaping a reward function; we now explain how a shaped reward function can be generated automatically from an SCT.

In an SCT, the transition distance functions that yield d_t are designed in such a way that it monotonically decreases as the robot moves towards the next phase of the task. As this constitutes a gradient towards the overall task, d_t can be used directly as a immediate primary rewards in addition to

[1] The underlying implementation is slightly different, in that the action is projected into the future in state space, and the resulting end-effector position is projected back onto a constraint if the constraint is violated (Quere et al., 2020).

a terminal sparse reward, as follows:

$$r_t = r_T^{\text{prim}} \qquad \text{Sparse} \qquad (4)$$

$$r_t = r_T^{\text{prim}} + k(d_{t-1} - d_t), \qquad \text{Shaped} \qquad (5)$$

where k is a constant and r_T^{prim} a terminal reward which is zero except at $t = T$. Note that the term $d_{t-1} - d_t$ encourages progress towards the next SCT state, and thus task completion, by rewarding a decrease in transition distance.

Secondary costs

While the primary reward provides feedback about task completion, in robotics it is also desirable for movements to have low accelerations and low force interactions. As such measures can commonly be measured at each time step, we add M such measures to the immediate rewards as

$$r_t^{\text{secon}} = -\sum_{i=1}^{M} \boldsymbol{v}_{i,t}^\top \boldsymbol{R}_i \boldsymbol{v}_{i,t}, \qquad (6)$$

where $\boldsymbol{v}_{i,t}$ is a vector representing a physical quantity whose magnitude is to be minimized during learning, and $\boldsymbol{R}_i$ is a diagonal matrix with positive entries representing a weighting factor. In this work we focus on two types of secondary rewards that penalize actions $\boldsymbol{a}_t$ and interaction forces $\boldsymbol{f}_t$, with which the overall cost function becomes:

$$r_t = \underbrace{\overbrace{r_T}^{\text{Sparse}} + \overbrace{k(d_{t-1} - d_t)}^{\text{Shaped part}}}_{\text{Primary}} \underbrace{-\boldsymbol{a}_t^\top \boldsymbol{R}_a \boldsymbol{a}_t - \boldsymbol{f}_t^\top \boldsymbol{R}_f \boldsymbol{f}_t}_{\text{Secondary}}. \qquad (7)$$

5 Evaluation

We first evaluated RL-SCT against RL without SCT in simulation on a pouring task with a KUKA IIWA (Fig. 5), and perform the same task directly on the real 7-DoF SARA robot (shown in Fig. 2a). To showcase the safe and efficient learning properties of our method, we also learn a grid-clamp placement task (Fig. 2b), showing how RL-SCT allows tasks involving contacts with the environment to be learned directly on the real robot. These experiments on the real robots aim at demonstrating the ability of RL-SCT to learn the tasks directly on the real robot, safely and efficiently.

As RL-SCT performs action space shaping and reward shaping in the environment (see Fig. 4), the underlying policy representation and RL algorithm used by the agent are unaffected by applying RL-SCT. To highlight this, we apply both Soft Actor Critic (SAC) (Haarnoja et al., 2018) and Truncated Quantile Critics (TQC) (Kuznetsov et al., 2020) from the open-source implementation in Stable-Baseline3

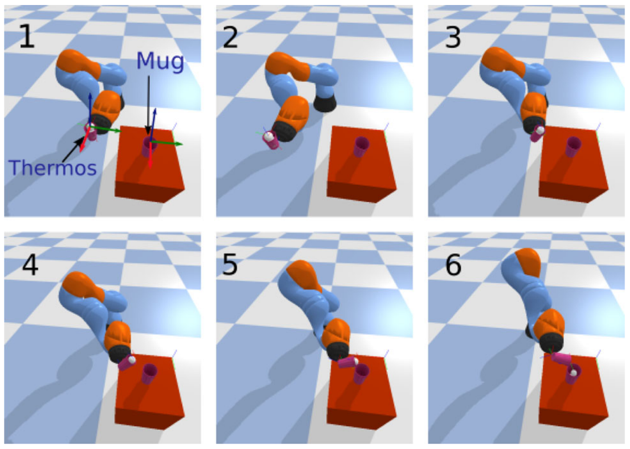


Fig. 5 Experimental setup for the pouring task with simulated KUKA IIWA holding the thermos and the target mug placed on the table with thermos tip and mug tip coordinate frames

(Raffin et al., 2021). The parameters for these algorithms are provided in “Appendix A”.

In all experiments, the policy is a feed-forward neural network with 2 hidden layers with 256 neurons in each hidden layer. The weights of the neural network are initialized to random values.

5.1 Pouring task

The pouring task consists of transferring the liquid from a container (thermos) attached to the end-effector of the robot to a target container (mug) placed in the environment. The SCT for the pouring task is visualized in Fig. 3 and explained in Table 2. In our experiments, the liquid is replaced by two ping-pong balls with 4 cm diameter, as illustrated in Figs. 2a and 5.

The SCT for this task was taken *as is* from our previous work on assistive robotics (Quere et al., 2020), where it was shown that using the SCT for shared control enables users to complete the pouring task more than twice as fast, on average. SCTs were an essential component for winning the Cybathlon Challenges for Assistive Robots in 2023 (Jaeger et al., 2023; Vogel et al., 2023).

The state for this task is $\boldsymbol{s}_t = \boldsymbol{x}_{\text{th},t}$ where $\boldsymbol{x}_{\text{th},t}$ is the 6D pose of the thermos tip expressed in the mug tip frame. In RL-SCT, as shown in Fig. 4, a policy generates an action $\boldsymbol{a}_t \in \mathbb{R}^3$ which acts as input to the SCT, which in turn generates a desired end-effector pose (Sect. 3). When running RL without SCT in the ablation study, the policy generates the 6D end-effector velocity as action $\boldsymbol{a}_t$ which is used to compute the target end-effector pose. The end-effector poses are given as references to the Cartesian position controller, which runs at 100Hz for the simulated KUKA IIWA robot and 8KHz for the SARA robot.

Table 2 Comparison of knowledge modeled using typical RL reward functions and RL-SCT for a pouring task

Task state	Classical RL reward function	RL-SCT
Translational	Reward for moving towards target, penalty for spilling (by directly penalizing rotational motion) and penalty for collision	**IM:** 3 inputs mapped to translational motion (No rotational motion needed). **AC:** Stay above table to avoid collisions Transition distance: Distance of mug tip from thermos tip
Tilt	Reward for moving and tilting towards the target simultaneously, penalty for spilling liquid (by penalizing translational and rotational motion in undesired directions), penalty for collision	**IM:** 3 inputs mapped to translational motion (Rotational motion is controlled by AC). **AC:** Stay above table to avoid collisions, tilt towards the target depending on distance Transition distance: Distance of mug tip from thermos tip
Pour	Reward for tilting towards the mug, penalty for spilling liquid (by penalizing translation to avoid spillage outside the target), and penalty for collision	**IM:** 1 input mapped to Z-axis translational motion, the others to the tilting motion (no horizontal motion). **AC:** Stay above table and mug to avoid collision, Transition distance: Tilting angle

On the real robot, the pose of the mug in the robot base frame was fixed and known beforehand. The pose of the thermos, grasped by the robot gripper, was calculated from the forward kinematics of the robot.

5.1.1 Reward functions

We evaluate our approach by comparing the above mentioned four scenarios. The baseline uses RL without SCTs, with a designed reward function, which is either shaped (RL-Shaped) or sparse (RL-Sparse). Our proposed method based on SCTs is also evaluated with a shaped (RL-SCT-Shaped) and sparse (RL-SCT-Sparse) reward function.

Sparse reward function

For the pouring task, the sparse reward function used in both RL-Sparse and RL-SCT-Sparse is

$$r_T = \begin{cases} 200, & \text{on successful termination} \\ -10, & \text{on termination due to collision} \\ 0, & \text{otherwise,} \end{cases} \tag{8}$$

$$r_t = r_T - \boldsymbol{a}_t^\top \boldsymbol{a}_t, \tag{9}$$

where $\boldsymbol{a}_t^\top \boldsymbol{a}_t$ a secondary cost component related to the action magnitude.

The task is considered successful only if both balls are successfully poured into the mug. In the experiments with the real robot, a human observing the task provided the feedback about the success of the task. The task is considered failed if the robot collides with the mug or the table, one or both balls are spilled out of the thermos or a pre-defined time limit in terms of time steps per episode is reached. In the event of collision with the table or target mug, the episode is terminated.

This sparse reward function is not very informative, as the primary reward consists only of three discrete rewards given only at the end of the episode.

Shaped reward function from the SCT

The reward function for RL-SCT-Shaped is automatically derived from d_t in the SCTs, as described in Sect. 4.2, i.e.

$$r_t = r_T + 200(d_{t-1} - d_t) - \boldsymbol{a}_t^\top \boldsymbol{a}_t, \tag{10}$$

where r_T is the same as in Eq. (9).

Hand-designed shaped reward function

We also hand-designed a shaped reward function for the pouring task. Our aim here is to show the intricacy of the design process for manually shaped reward functions, and its similarity to the process of designing an SCT.

First, the pouring task is divided in two phases, (1) transport the thermos near the mug without spilling the liquid, and (2) pour the liquid by tilting the thermos around the appropriate axis. The translation phase takes the thermos to a fixed distance near the target mug, and the pouring phase rotates the thermos avoiding any translation. These phases need to be identified correctly, and give the reward for not tilting the thermos in the first phase and reward for tilting around the correct axis and not around the other axes in the second phase.

The implementation of this reward function for RL-Shaped is given by Eq. (11)–(13), where j_t is the distance between the thermos tip and the mug tip, and ϕ_t^x, ϕ_t^y and ϕ_t^z are the angular positions, expressed as Euler angles (roll, pitch and yaw) of the thermos tip in the mug tip frame (orientations of the tip frames are depicted in Fig. 5).

$$r_t^{\text{prim}} = r_t^{\text{dist}} + r_t^{\text{tilt}}/4 - \boldsymbol{a}_t^\top \boldsymbol{a}_t + r_T, \tag{11}$$

$$r_{\text{dist}} = \begin{cases} 15(j_{t-1} - j_t), & \text{if } j_t > 0.04 \\ 15(j_{t-1} - j_t) + 0.2, & \text{otherwise} \end{cases} \tag{12}$$

$$r_t^{\text{tilt}} = \begin{cases} -|\delta\phi_t^x| - |\delta\phi_t^y| - |\delta\phi_t^z|, & \text{if } j_t > 0.04 \\ 20\delta\phi_t^x - |\delta\phi_t^y| - |\delta\phi_t^z|, & \text{otherwise} \end{cases} \tag{13}$$

In the design of the shaped reward function, we recognize many similarities to SCT design, e.g. the definition of phases, transition thresholds between the phases, definition of constraints, etc. Table 2 provides a full analysis of the similarities.

5.1.2 Results

Figures 6 and 7 show the rate of success, spillage and collision with the environment in simulation both with SAC and TQC. The curves show the mean and standard deviation obtained from 10 different learning sessions. The results from the different combinations of reward types and presence/absence of SCTs give important insights about the influence of SCT on the learning process, and failures during learning.

Using SAC (Fig. 6) RL-Sparse is not able to learn the task in 1400 (maximum) episodes $\boxed{\text{A}}$. RL-Shaped converges within 1200 episodes $\boxed{\text{B}}$, but also shows very high spill rate and collision rate due to unconstrained exploration $\boxed{\text{C}}$. RL-SCT-Sparse converges within 600 episodes $\boxed{\text{D}}$, shows very low spill rate and no collisions $\boxed{\text{E}}$ due to the constraints. RL-SCT-Shaped converges within 250 episodes $\boxed{\text{G}}$ showing the best performance overall, with small initial spill rate $\boxed{\text{F}}$ and no collisions.

Using TQC (Fig. 7) RL-Sparse is not able to learn the task in 1400 (maximum) episodes $\boxed{\text{H}}$. RL-Shaped converges within 700 episodes $\boxed{\text{I}}$, but also shows very high spill rate and collision rate due to unconstrained exploration $\boxed{\text{J}}$. RL-SCT-Sparse converges within 1000 episodes $\boxed{\text{K}}$ and shows very low spill rate and no collisions $\boxed{\text{L}}$ due to the constraints. RL-SCT-Shaped converges within 250 episodes $\boxed{\text{M}}$ showing the best performance overall, with small initial spill rate $\boxed{\text{N}}$ and no collisions. Figure 9 summarizes these results.

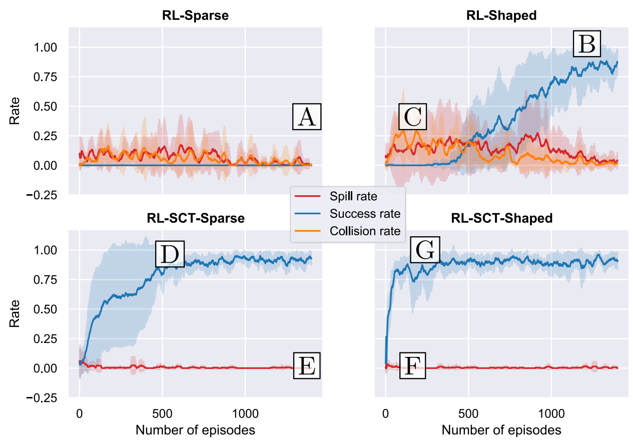


Fig. 6 Success, spill and collision rates vs number of episodes, in different experimental settings in simulation using SAC. Each point shows the average of 10 learning sessions, together with one standard deviation

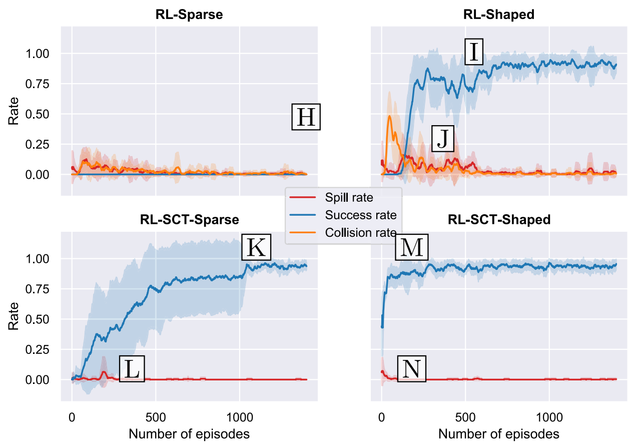


Fig. 7 As Fig. 6, but using TQC

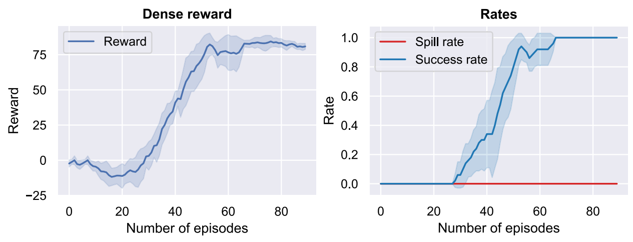


Fig. 8 Success rate, spill rate and reward for RL-SCT-Shaped on the real robot using SAC. Each point shows the mean and one standard deviation over 5 learning sessions

The results on the real robot are shown in Fig. 8. As we wanted to avoid collisions as much as possible during training, and running more than 1000 episodes multiple times takes prohibitively long on the real robot, we only ran RL-SCT-Shaped with SAC. Figure 8 shows the results of 5 independent learning sessions. The learning agent achieves a success rate of 1 within 65 episodes amounting to 9700 time steps, on average. This corresponds to 16 min of training time, without considering the time taken for resetting the environment.

5.1.3 Discussion

From the learning curves and the summary in Fig. 9, we derive the following conclusions. As expected Shaped (yellow/green) outperforms Sparse (red/blue) by several orders of magnitude wrt. convergence speed as shaped rewards are more informative (top two graphs). The boxplots further confirm that SCTs speed up learning (blue vs. red and green vs.

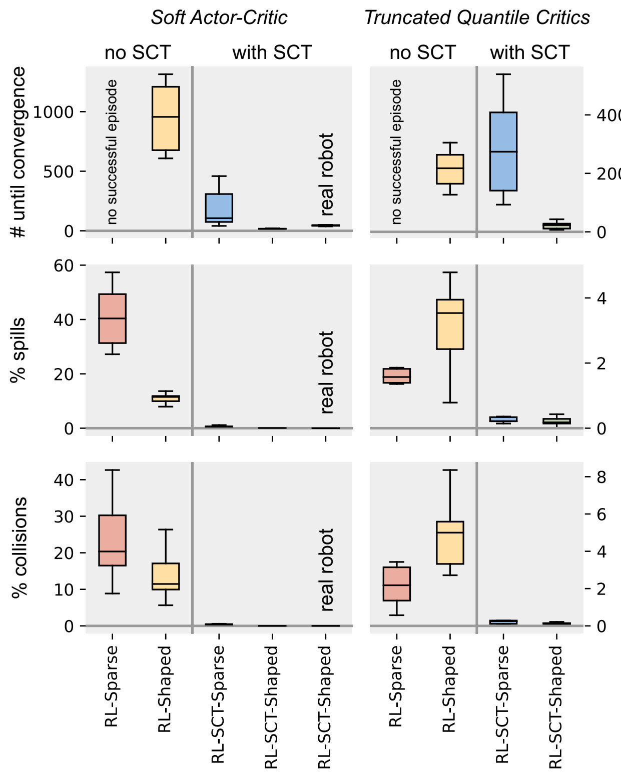


Fig. 9 Boxplots summarizing the liquid pouring experiments. Left: SAC (including real robot experiment), Right: TQC. '# until convergence': number of episodes until 10 subsequent episodes achieve the task. '% spills': number of episodes in which a spill occured. '% collisions': number of episodes in which a collision occured

blue); indeed RL-Sparse never converges within 1400 episodes in any of the 10 learning sessions.

From an RL perspective, speed of convergence and the rewards achieved after convergence are the most important measures of success. From a robotics perspective, safety is just as important, and this aspect is highlighted in the lower two rows of the boxplots. We observe that with SCTs (boxplots to the right of the vertical gray lines), there are hardly any collisions or spills; especially the 0 spills and 0 collisions on the robot are of importance. With SAC, the median percentage (over 10 learning sessions) of episodes that involves collision is 12 and 20 for sparse/shaped rewards respectively (bottom left graph, red/yellow) . On a real robot, this would lead to an unacceptable amount of wear-and-tear, which is why we do not run experiments on the robot without SCT. This confirms that SCT-RL leads to safer learning, enabling RL directly on the real robot.

In comparison to SAC, we see a much lower rate of spills and collisions without SCTs (red/yellow) with TQC, i.e. between 2 and 5%. This is because in many learning sessions, the robot learns to not move at all. The robot does not receive the reward for completing the task then, but it also does not get the penalties for collisions. The reason for

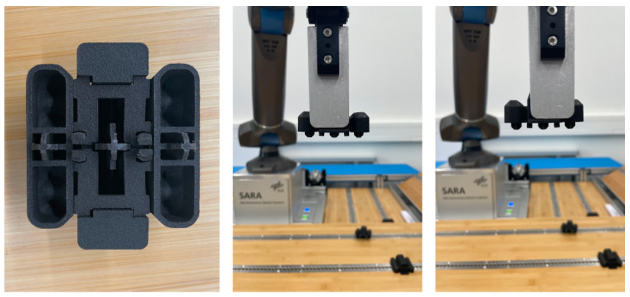


Fig. 10 Bottom view of a grid-clamp (left) and uncertainty when grasping grid-clamps (middle/right)

the low rates is not that collisions are avoided during the movement; rather there is hardly any movement at all. With RL-Shaped, we see more collisions and spills for TQC than for RL-Sparse. This is because the reward gradient in the shaped reward leads to more movement than the sparse reward which is never received.

The results show that RL-SCT can learn the multi-phase pouring task on the real robot safely (no collisions) and efficiently (convergence in less than 100 episodes). Furthermore, safe exploration is not sacrificed when using a sparse reward function.

In these experiments, reward shaping was essential to making RL without SCTs feasible within 1400 episodes. Therefore, if the RL expert needs to invest time in designing a complex multi-phase shaped reward function as in Eq. (11) to make learning feasible, we argue that this time is better invested in designing *explicit* multi-phase constraints. This will speed up learning even more, and, critically, ensure safety during learning, as constraints no longer need to be violated in order to learn them. What is known need not be learned.

5.2 Grid clamp insertion task

In order to evaluate the ability of RL-SCT to learn tasks involving contacts, a grid-clamp placement task was learned on the SARA robot (Fig. 2b). Grid clamps are used in DLR's Factory of Future setup to reconfigure variable workstations. The robot learns to insert the grid-clamp into the grid holes on the table, similar to a peg-in-hole task.

During the insertion process, 5 peg-like heads, which secure the grid-clamp on the table, are needed to be inserted simultaneously and snapped into the holes on the table, as shown in Fig. 10 (top). The task is challenging due to the kinematic inaccuracy mainly in the horizontal plane (x–y), the pose of the grid-clamp (estimated using forward kinematics of the robot) and the holes on the table. Additional inaccuracies arise from the grasp of the grid-clamp, as shown in Fig. 10, as well as the inherent inaccuracy of the Cartesian impedance controller and table pose calibration.

In our experiments, we place the grid-clamps at 6 different locations to be picked up by the robot, with a user resetting the setup every 6th episode. This way we ensure that the robot learns under uncertainties arising from both the grasp and the robot configuration. During the experiments, the grid-clamp is grasped and transported above the hole using a hand designed grid-clamp pick up skill. The RL policy takes over when the grid-clamp is above the target hole.

It is possible to complete the task successfully despite the above mentioned uncertainties by appropriately reacting to the contact forces. While the task can still be executed successfully with high contact forces, minimizing them is critical for safe long-term operation of the robot. We therefore minimize contact forces by including them as a secondary reward, with the primary reward ensuring the task completion.

The SCT used for learning this task has the following design:

- **Phases:** one phase is used, governing the translational motion of the peg towards the hole.
- **Constraints:** the rotational degrees of freedom are constrained (we assume negligible uncertainty on the hole orientation) and the robot is free to move in translational DoFs within a cuboid constraint of size 0.8 cm × 0.8 cm × 10 cm.
- **Transitions:** the transition distance d_t is the z-axis distance between the grid-clamp and the hole.

For the grid-clamp insertion task (Sect. 5.2), the 6D state given to the policy is $\boldsymbol{s}_t = [\boldsymbol{x}_t^\top \ \boldsymbol{f}_t^\top]^\top$ where $\boldsymbol{x}_t \in \mathbb{R}^3$ is the position of the grid-clamp (attached to the robot gripper) expressed in the hole frame and $\boldsymbol{f}_t \in \mathbb{R}^3$ is the force measured at the grid-clamp frame by the integrated force/torque sensor.

5.2.1 Reward function

Using the outlined SCT, the policy mainly has to learn the contact dynamics during insertion. Particularly, the robot is encouraged to reduce the contact force by introducing a secondary cost component for the measured force $\boldsymbol{f}_t$ at the grid-clamp frame. The reward function is thus given by:

$$r_T = \begin{cases} 50, & \text{on successful termination} \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

$$r_t = r_T + 300(d_{t-1} - d_t) - \frac{1}{12}\boldsymbol{f}_t^\top \boldsymbol{f}_t - \boldsymbol{a}_t^\top \boldsymbol{a}_t. \tag{15}$$

We determine task success from the z coordinate of the end-effector; if it drops below a threshold, this indicates the clamp has been placed successfully. To evaluate the effect of the interaction force cost on the interaction forces during learning, two sets of experiments were conducted: 1) using the interaction force cost in the reward function ($\boldsymbol{f}_t^\top \boldsymbol{f}_t$) and 2) without using interaction force costs.

5.2.2 Results

In both cases, with and without using interaction force cost in the reward function, the robot learns the grid-clamp insertion task in less than 70 episodes amounting to 9820 time steps (≈ 17 min) as shown in Fig. 11. It achieves 100% success in 70 episodes $\boxed{\text{O}}$. Figure 11 (center) shows the average interaction force per episode, computed, for episode i, as $\boldsymbol{f}_i = \sum_{n=1}^{N_i} \boldsymbol{f}_{i,n}^\top \boldsymbol{f}_{i,n}/N_i$, where $\boldsymbol{f}_{i,n}$ is the interaction force measured at step n and N_i is the number of steps in the episode.

Comparison of the average interaction forces, $\boxed{\text{P}}$ and $\boxed{\text{Q}}$, shows that the robot uses significantly less force during the learning process when the reward function contains the secondary cost term associated with interaction forces. This happens without significantly affecting other components of learning, particulary the time required to achieve 100% success rate $\boxed{\text{O}}$ and the speed of task completion in terms of time steps $\boxed{\text{R}}$ when the policy is learned. Figure 12 shows the comparison of the rewards gained over number of episodes. For both cases, learning in terms of reward converges in ≈70 episodes.

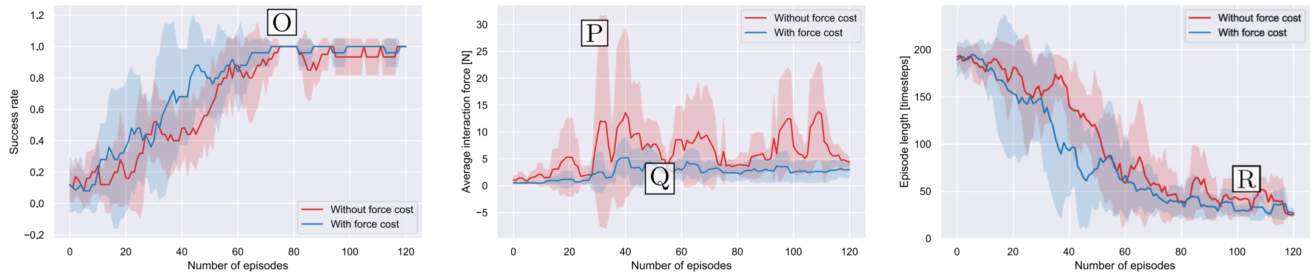


Fig. 11 Learning performance of the SAC policy on the grid-clamp insertion task for RL-SCT-Shaped on the real robot with interaction force cost. The plots shows learning performance with and without force interaction cost in the reward function. Each point shows the mean and one standard deviation over 5 learning sessions

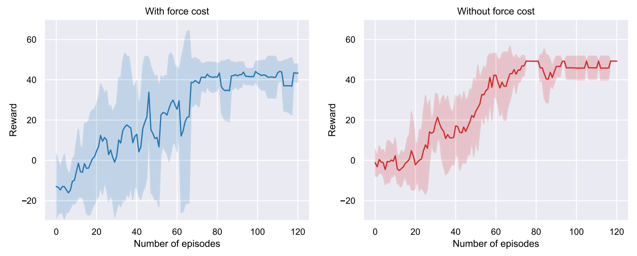


Fig. 12 Comparison of rewards achieved by learning agent with reward functions with and without interaction force cost. Each point shows the mean and one standard deviation over 5 learning sessions

5.2.3 Discussion

RL-SCT can also effectively learn a task involving contact forces directly on the real robot as demonstrated by learning the grid-clamp insertion. Notably, through the definition of a transition-distance-based primary reward, the reward curves (both with and without force cost) converge quickly (Fig. 12). The secondary reward penalizing interaction forces then allows the learning of a policy that uses an optimal force profile.

It is worth emphasizing that both tasks were learned not only in an small amount of time but also with safety for both the robot and the environment, with no collisions observed. This was ensured by the SCT constraints in task space. Moreover, we highlight that, despite achieving 100% success rate after 70 episodes, in the setting without force cost the robot keeps exploring, resulting in subsequent failures (Fig. 11-left). We observed that continued exploration with inadequate force behaviors often leads the robot to apply too high contact forces on the environment, increasing the likelihood of failure even after the task has been learned. Such undesired force profiles can be seen in Fig. 11-center after the 80–90 episode range.

6 Conclusion

We proposed a framework—RL-SCT—to guide reinforcement learning with constraints that are represented as Shared Control Templates. We have demonstrated that the properties that users expect from shared control—empowerment through freedom of movement, safety by enforcing constraints, low-dimensional input commands to facilitate control—are properties that are also advantageous for robot reinforcement learning.

Our experiments show that the explicit representation of constraints leads to faster learning, and without the need to design complicated reward functions to represent these constraints. Particularly, we demonstrated that RL-SCT facilitates reinforcement learning on real robots. In a pouring task (without contacts between robot and environment) we showed that RL-SCT allows the robot to learn the task in 16 min without dangerous interactions with the environment. Given the importance of safety during in-contact tasks, we also applied our approach to a grid-clamp insertion task in the presence of position uncertainties to learn a policy which succeeds at the task while minimizing contact forces. Similarly to the pouring task, the robot was able to quickly learn the task in ≈ 17 min, exhibiting low contact forces when compared to a baseline which did not account for contact force minimization. In view of the difficulty to accurately model contacts in simulation (and the subsequent reality gap) our results gain special relevance as we show that RL-SCT can be used to learn directly on the robot safely and efficiently, while minimizing interaction forces.

Despite the successful results obtained, some limitations of RL-SCT should be highlighted. On the one hand, our approach is tailored to the learning of tasks involving the use of objects with known constraints. It is less suited for learning intricate motor skills, such as those required for juggling, ball-in-cup, or locomotion. On the other hand, the design of SCTs can be cumbersome, especially for new tasks. However, we argue that designing SCTs leads to safer and faster learning than the classical approach of carefully designing shaped rewards, which also often takes a significant amount of time (and trial and error, with all the potentially dangerous interactions it entails). At the same time, we believe that re-using SCTs from already existing tasks in new ones is a promising way to mitigate the design effort.

In future work we will investigate methods to extract the required SCT components from demonstrations (Quere et al., 2021, 2024), namely constraints and nominal solutions to complete a given task. Having an initial policy that can be extracted from demonstrations and which only fails in specific conditions can help further simplify the reward functions used by RL-SCT. Motion primitive representations which capture aleatoric and epistemic uncertainties (Huang et al., 2019; Silvério & Huang, 2023) are promising approaches to build on, to achieve such goals.

A Parameters for SAC and TQC

Parameters for iiwa robot simulation (pouring task)
SAC parameters: `learning rate`=0.0008, `buffer size` =1000000, `discount factor`=0.95, `soft update coefficient`=0.02, `training frequency` =8, `gradient steps`=8, `learning starts at` =1000, `batch size`=256.

TQC parameters: `learning rate`=0.0008, `buffer size`=1000000, `discount factor`=0.95, `soft update coefficient`=0.02, `training frequency`

=8, `gradient steps`=8, `learning starts`=1000, `batch size`=256.

Parameters for real SARA robot (pouring task and grid clamp task)

SAC parameters: `learning rate`=0.001, `buffer size` =1000000, `discount factor`=0.98, `soft update coefficient`=0.02, `training frequency` =8, `gradient steps`=8, `learning starts at` =1000, `batch size`=256.

Acknowledgements The research reported in this paper has been (partially) supported by the German Research Foundation DFG, as part of Collaborative Research Center (Sonderforschungsbereich) 1320 Project-ID 329551904 "EASE—Everyday Activity Science and Engineering", University of Bremen (http://www.ease-crc.org/). The research was conducted in subproject R04: Cognition-enabled execution of everyday actions. This work was also supported in part by the European Union's Horizon Research and Innovation Programme under Grants 101070596 (euROBIN) and 101136067 (INVERSE).

Author Contributions A.P., J.S., F.S. designed the experiments and wrote the manuscript. A.P. implemented the SCT-RL framework and implemented and conducted the experiments. G.Q. provided the SCT implementation and wrote the SCT description. A.R. provided support for Stable Baselines 3. G.Q. and A.R. reviewed the manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

Andrychowicz, O. M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., & Schneider, J. (2020). Learning dexterous in-hand manipulation. *The International Journal of Robotics Research, 39*(1), 3–20.

Apolinarska, A. A., Pacher, M., Li, H., Cote, N., Pastrana, R., Gramazio, F., & Kohler, M. (2021). Robotic assembly of timber joints using reinforcement learning. *Automation in Construction, 125*, 103569.

Beltran-Hernandez, C. C., Petit, D., Ramirez-Alpizar, I. G., Nishi, T., Kikuchi, S., Matsubara, T., & Harada, K. (2020). Learning force control for contact-rich manipulation tasks with rigid position-controlled robots. *IEEE Robotics and Automation Letters, 5*(4), 5709–5716.

Bitzer, S., Howard, M., & Vijayakumar, S. (2010). Using dimensionality reduction to exploit constraints in reinforcement learning. In *2010 IEEE/RSJ international conference on intelligent robots and systems* (pp. 3219–3225). IEEE.

Bowyer, S. A., Davies, B. L., & Baena, F. R. (2013). Active constraints/virtual fixtures: A survey. *IEEE Transactions on Robotics, 30*(1), 138–157.

Bruyninckx, H., & De Schutter, J. (1996). Specification of force-controlled actions in the "task frame formalism": A synthesis. *IEEE Transactions on Robotics and Automation, 12*(4), 581–589.

Buchli, J., Stulp, F., Theodorou, E., & Schaal, S. (2011). Learning variable impedance control. *International Journal of Robotics Research, 30*(7), 820–833.

Bustamante, S., Quere, G., Hagmann, K., Wu, X., Schmaus, P., Vogel, J., Stulp, F., & Leidner, D. (2021). Toward seamless transitions between shared control and supervised autonomy in robotic assistance. *IEEE Robotics and Automation Letters, 6*(2), 3833–3840.

Chatterji, N., Pacchiano, A., Bartlett, P., & Jordan, M. (2022). On the theory of reinforcement learning with once-per-episode feedback. 2105.14363.

Cheng, R., Orosz, G., Murray, R. M., & Burdick, J. W. (2019). End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 3387–3395).

Cruz, F., Magg, S., Weber, C., & Wermter, S. (2014). Improving reinforcement learning with interactive feedback and affordances. In *4th international conference on development and learning and on epigenetic robotics* (pp. 165–170). IEEE.

Cruz, F., Magg, S., Weber, C., & Wermter, S. (2016). Training agents with interactive reinforcement learning and contextual affordances. *IEEE Transactions on Cognitive and Developmental Systems, 8*(4), 271–284.

Cruz, F., Parisi, G. I., & Wermter, S. (2018). Multi-modal feedback for affordance-driven interactive reinforcement learning. In *2018 international joint conference on neural networks (IJCNN)* (pp. 1–8). IEEE.

Curran, W., Brys, T., Aha, D., Taylor, M., & Smart, W. D. (2016). Dimensionality reduced reinforcement learning for assistive robots. In *2016 AAAI fall symposium series*.

Daniel, C., Neumann, G., Kroemer, O., & Peters, J. (2013). Learning sequential motor tasks. In *2013 IEEE international conference on robotics and automation* (pp. 2626–2632). IEEE.

Davchev, T., Luck, K. S., Burke, M., Meier, F., Schaal, S., & Ramamoorthy, S. (2022). Residual learning from demonstration: Adapting dmps for contact-rich manipulation. *IEEE Robotics and Automation Letters, 7*(2), 4488–4495.

Dragan, A. D., & Srinivasa, S. S. (2013). A policy-blending formalism for shared control. *The International Journal of Robotics Research, 32*(7), 790–805. https://doi.org/10.1177/0278364913490324

Elguea-Aguinaco, Í., Serrano-Muñoz, A., Chrysostomou, D., Inziarte-Hidalgo, I., Bøgh, S., & Arana-Arexolaleiba, N. (2023). A review on reinforcement learning for contact-rich robotic manipulation tasks. *Robotics and Computer-Integrated Manufacturing, 81*, 102517.

Gibson, J. J. (1979) The ecological approach to visual perception. Houghton Mifflin Harcourt (HMH).

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning, PMLR* (pp. 1861–1870).

He, Z., & Ciocarlie, M. (2022). Discovering synergies for robot manipulation with multi-task reinforcement learning. In *2022 international conference on robotics and automation (ICRA)* (pp. 2714–2721). IEEE.

Huang, S., & Ontañón, S. (2020). A closer look at invalid action masking in policy gradient algorithms. CoRR arXiv:2006.14171

Huang, Y., Rozo, L., Silvério, J., & Caldwell, D. G. (2019). Kernelized movement primitives. *International Journal of Robotics Research, 38*(7), 833–852.

Iskandar, M., Ott, C., Eiberger, O., Keppler, M., Albu-Schäffer, A., & Dietrich, A. (2020). Joint-level control of the dlr lightweight robot sara. In *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 8903–8910). IEEE.

Jaeger, L., Baptista, R. D., Basla, C., Capsi-Morales, P., Kim, Y. K., Nakajima, S., Piazza, C., Sommerhalder, M., Tonin, L., Valle, G., & Riener, R. (2023). How the cybathlon competition has advanced assistive technologies. *Annual Review of Control, Robotics, and Autonomous Systems, 6*(1), 447–476.

Kalakrishnan, M., Righetti, L., Pastor, P., & Schaal, S. (2011). Learning force control policies for compliant manipulation. In *2011 IEEE/RSJ international conference on intelligent robots and systems* (pp. 4639–4644). https://doi.org/10.1109/IROS.2011.6095096

Kanervisto, A., Scheller, C., & Hautamäki, V. (2020). Action space shaping in deep reinforcement learning. CoRR arXiv:2004.00980

Khetarpal, K., Ahmed, Z., Comanici, G., Abel, D., & Precup, D. (2020). What can i do here? A theory of affordances in reinforcement learning. In *International conference on machine learning, PMLR* (pp. 5243–5253).

Kim, Y. G., Na, M., & Song, J. B. (2021). Reinforcement learning-based sim-to-real impedance parameter tuning for robotic assembly. In *2021 21st international conference on control, automation and systems (ICCAS)* (pp. 833–836). IEEE.

Koert, D., Kircher, M., Salikutluk, V., D'Eramo, C., & Peters, J. (2020). Multi-channel interactive reinforcement learning for sequential tasks. *Frontiers in Robotics and AI, 7*, 97.

Kolter, J. Z., & Ng, A. Y. (2007). Learning omnidirectional path following using dimensionality reduction. In *Robotics: Science and systems* (pp. 27–30).

Kozlovsky, S., Newman, E., & Zacksenhouse, M. (2022). Reinforcement learning of impedance policies for peg-in-hole tasks: Role of asymmetric matrices. *IEEE Robotics and Automation Letters, 7*(4), 10898–10905.

Krishnan, S., Garg, A., Liaw, R., Thananjeyan, B., Miller, L., Pokorny, F. T., & Goldberg, K. (2019). Swirl: A sequential windowed inverse reinforcement learning algorithm for robot tasks with delayed rewards. *The International Journal of Robotics Research, 38*(2–3), 126–145.

Kroemer, O., Daniel, C., Neumann, G., Van Hoof, H., & Peters, J. (2015). Towards learning hierarchical skills for multi-phase manipulation tasks. In *2015 IEEE international conference on robotics and automation (ICRA)* (pp. 1503–1510). IEEE.

Kuznetsov, A., Shvechikov, P., Grishin, A., & Vetrov, D. (2020). Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In *International conference on machine learning, PMLR* (pp. 5556–5566).

Lee, J., Hwangbo, J., Wellhausen, L., Koltun, V., & Hutter, M. (2020a). Learning quadrupedal locomotion over challenging terrain. *Science Robotics, 5*(47), eabc5986.

Lee, M. A., Florensa, C., Tremblay, J., Ratliff, N., Garg, A., Ramos, F., & Fox, D. (2020b). Guided uncertainty-aware policy optimization: Combining learning and model-based strategies for sample-efficient policy learning. In *2020 IEEE international conference on robotics and automation (ICRA)* (pp. 7505–7512). IEEE.

Leidner, D., Borst, C., & Hirzinger, G. (2012). Things are made for what they are: Solving manipulation tasks by using functional object classes. In *International conference on humanoid robots (HUMANOIDS)*. https://elib.dlr.de/80508/

Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., & Quillen, D. (2018). Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research, 37*(4–5), 421–436.

Liu, P., Tateo, D., Ammar, H. B., & Peters, J. (2022). Robot reinforcement learning on the constraint manifold. In *Conference on robot learning, PMLR* (pp. 1357–1366).

Luck, K. S., Neumann, G., Berger, E., Peters, J., & Amor, H. B. (2014). Latent space policy search for robotics. In *2014 IEEE/RSJ international conference on intelligent robots and systems* (pp. 1434–1440). IEEE.

Luo, J., Solowjow, E., Wen, C., Ojea, J. A., Agogino, A. M., Tamar, A., & Abbeel, P. (2019). Reinforcement learning on variable impedance controller for high-precision robotic assembly. In *2019 international conference on robotics and automation (ICRA)* (pp. 3080–3087). IEEE.

Mason, M. T. (1981). Compliance and force control for computer controlled manipulators. *IEEE Transactions on Systems, Man, and Cybernetics, 11*(6), 418–432.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602

Ng, A. Y., Harada, D., & Russell. S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml* (pp. 278–287).

Padalkar, A., Nieuwenhuisen, M., Schneider, S., & Schulz, D. (2020a). Learning to close the gap: Combining task frame formalism and reinforcement learning for compliant vegetable cutting. In *ICINCO* (pp. 221–231).

Padalkar, A., Nieuwenhuisen, M., Schulz, D., & Stulp, F. (2020b). Closing the gap: Combining task specification and reinforcement learning for compliant vegetable cutting. In *International conference on informatics in control, automation and robotics* (pp. 187–206). Springer.

Parisi, S., Ramstedt, S., & Peters, J. (2017). Goal-driven dimensionality reduction for reinforcement learning. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 4634–4639). IEEE.

Ploeger, K., Lutter, M., & Peters, J. (2020). High acceleration reinforcement learning for real-world juggling with binary rewards. arXiv preprint arXiv:2010.13483

Quere, G., Bustamante, S., Hagengruber, A., Vogel, J., Steinmetz, F., & Stulp, F. (2021). Learning and interactive design of shared control templates. In *2021 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 1887–1894). IEEE.

Quere, G., Hagengruber, A., Iskandar, M., Bustamante, S., Leidner, D., Stulp, F., & Vogel, J. (2020). Shared control templates for assistive robotics. In *2020 IEEE international conference on robotics and automation (ICRA)* (pp. 1956–1962).

Quere, G., Stulp, F., Filliat, D., & Silverio, J. (2024). A probabilistic approach for learning and adapting shared control skills with the human in the loop. In *International conference on robotics and automation (ICRA)*.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*. https://elib.dlr.de/146386/

Raffin, A., Hill, A., Traoré, R., Lesort, T., Díaz-Rodríguez, N., & Filliat, D. (2019). Decoupling feature extraction from policy learning: assessing benefits of state representation learning in goal based robotics. SPiRL workshop ICLR.

Reinhart, R. F., & Steil, J. J. (2015). Efficient policy search in low-dimensional embedding spaces by generalizing motion primitives with a parameterized skill memory. *Autonomous Robots, 38*, 331–348.

Salvato, E., Fenu, G., Medvet, E., & Pellegrino, F. A. (2021). Crossing the reality gap: a survey on sim-to-real transferability of robot controllers in reinforcement learning. *IEEE Access, 9*, 153171–153187.

Schwab, D., Springenberg, T., Martins, M.F., Lampe, T., Neunert, M., Abdolmaleki, A., Hertweck, T., Hafner, R., Nori, F., & Riedmiller, M. (2019). Simultaneously learning vision and feature-based control policies for real-world ball-in-a-cup. arXiv preprint arXiv:1902.04706

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., & Dieleman, S. (2016). Mastering the game of go with deep neural networks and tree search. *Nature, 529*(7587), 484–489.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., & Chen, Y. (2017). Mastering the game of go without human knowledge. *Nature, 550*(7676), 354–359.

Silvério, J., & Huang, Y. (2023). A non-parametric skill representation with soft null space projectors for fast generalization. In *Proceedings of the IEEE international conference on robotics and automation (ICRA)* (pp. 2988–2994).

Stolle, M., & Precup, D. (2002). Learning options in reinforcement learning. In *Abstraction, reformulation, and approximation: 5th international symposium, SARA 2002 Kananaskis, Alberta, Canada August 2–4, 2002 proceedings 5* (pp. 212–223). Springer.

Sutton, R. S., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence, 112*(1–2), 181–211. https://doi.org/10.1016/s0004-3702(99)00052-1

Taylor, A., Singletary, A., Yue, Y., & Ames, A. (2020). Learning for safety-critical control with control barrier functions. In *Learning for dynamics and control, PMLR* (pp. 708–717).

Tosatto, S., Chalvatzaki, G., & Peters, J. (2021). Contextual latent-movements off-policy optimization for robotic manipulation skills. In *2021 IEEE international conference on robotics and automation (ICRA)* (pp. 10815–10821). IEEE.

Vecerik, M., Sushkov, O., Barker, D., Rothörl, T., Hester, T., & Scholz, J. (2019). A practical approach to insertion with variable socket position using deep reinforcement learning. In *2019 international conference on robotics and automation (ICRA)* (pp. 754–760). IEEE.

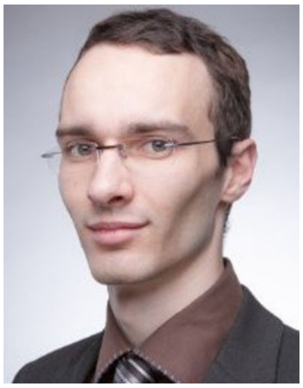
Vogel, J., Hagengruber, A., Iskandar, M., Quere, G., Leipscher, U., Bustamante, S., Dietrich, A., Höppner, H., Leidner, D., & Albu-Schäffer, A. (2020). Edan-an emg-controlled daily assistant to help people with physical disabilities. In *2020 IEEE/RSJ international conference on intelligent robots and systems, IROS 2020.*

Vogel, J., Hagengruber, A., & Quere, G. (2023). Mattias and edan winning at cybathlon challenges march 2023. https://www.youtube.com/watch?v=EoER_5vYZsU

Zhao, T.Z., Luo, J., Sushkov, O., Pevceviciute, R., Heess, N., Scholz, J., Schaal, S., & Levine, S. (2022). Offline meta-reinforcement learning for industrial insertion. In *2022 international conference on robotics and automation (ICRA)* (pp. 6386–6393). IEEE.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

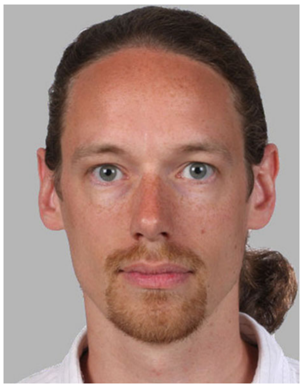
Abhishek Padalkar is currently working as a roboticist at the Institute of Robotics and Mechatronics, German Aerospace Center (DLR). He earned his Master of Science degree in Autonomous Systems from Hochschule Bonn-Rhein-Sieg, Bonn, Germany. The main focus of his research is to investigate reinforcement learning for learning tasks directly on the robots.



Gabriel Quere was born in France in 1993. After an engineering degree at Ensta Paris and a Master of Science at Université Paris-Saclay, he joined the Robotics and Mechatronic Center (RMC) of the German Aerospace Center (DLR), in the re-enabling robotics team. His research interest includes shared control for assistive robots and learning skills from demonstrations.

Antonin Raffin is a research engineer in robotics and machine learning at the German Aerospace Center (DLR). Previously, he worked on state representation learning in the ENSTA robotics lab (U2IS), where he created the Stable-Baselines library together with Ashley Hill. His research focus is now on applying reinforcement learning directly to real robots, for which he continues to maintain the Stable-Baselines3 library.

João Silvério is a group leader at the German Aerospace Center (DLR) since April 2022. He received his Ph.D. in Robotics (2017) from the University of Genoa (Genoa, Italy) and the Italian Institute of Technology, where he was also a postdoctoral researcher until May 2019. Between June 2019 and March 2022 he was a postdoctoral researcher at the Idiap Research Institute (Martigny, Switzerland). He is interested in machine learning for robotics, particularly imitation and reinforcement learning. Webpage: http://jpsilverio.github.io.



Freek Stulp received the doctorate degree in computer science from the Technical University of Munich, Munich, Germany, in 2007. He is currently the Head of the Department of Cognitive Robotics, Institute of Robotics and Mechatronics, German Aerospace Center. Previously, he was an Assistant Professor with the École Nationale Supérieure de Techniques Avancées.