2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Guiding Reinforcement Learning with Shared Control Templates

Abhishek Padalkar[1], Gabriel Quere[1], Franz Steinmetz[1], Antonin Raffin[1],
Matthias Nieuwenhuisen[2], João Silvério[1], Freek Stulp[1]

***Abstract*— Purposeful interaction with objects usually requires certain constraints to be respected, e.g. keeping a bottle upright to avoid spilling. In reinforcement learning, such constraints are typically encoded in the reward function. As a consequence, constraints can only be learned by violating them. This often precludes learning on the physical robot, as it may take many trials to learn the constraints, and the necessity to violate them during the trial-and-error learning may be unsafe. We have serendipitously discovered that constraint representations for shared control – in particular *Shared Control Templates (SCTs)* – are ideally suited for safely guiding RL. Representing constraints explicitly, rather than implicitly in the reward function, also simplifies the design of the reward function. The main advantage of the approach is safer, faster learning without constraint violations (even with sparse reward functions). We demonstrate this in a pouring task in simulation and on a real robot, where learning the task requires only 65 episodes in 16 minutes.**

I. Introduction

Reinforcement learning's super-human performance on Atari games [1] and Go [2] has led to a new wave of interest in this method. In robotics, reinforcement learning (RL) has made headlines by learning intricate and impressive motor skills such as juggling [3], ball-in-cup [4], in-hand manipulation [5], pick-and-place [6], or locomotion over highly variable terrain [7].

What sets human activities of daily living (e.g. pouring a drink, opening a door, heating food in a microwave) apart from the intricate motor skills above is that these activities commonly involve interactions with objects that have a specific purpose (e.g. bottles, doors, and microwaves). This purpose entails certain constraints, i.e. full bottles should be held upright, doors rotate around their hinge, etc.

In reinforcement learning, such constraints are typically encoded implicitly in the reward function, e.g. penalties are given for tilting a bottle. This approach implies that constraints need to be violated in order to learn them. In this paper, we take the stance that known, purposeful constraints should be modeled explicitly and enforced during learning and execution. This has the following advantages 1) Safer learning, as constraints are enforced during learning; 2) Faster learning, as constraints need not be learned; 3) Together, 1 and 2 enable direct learning on the real robot; 4) Simplified reward function design, as constraints need no longer be implicitly represented in the reward.

[1] German Aerospace Center (DLR), Robotics and Mechatronics Center (RMC), Münchner Str. 20, 82234 Weßling, Germany
[2] Fraunhofer Institute for Communication, Information Processing and Ergonomics FKIE, Fraunhoferstr. 20, 53343 Wachtberg

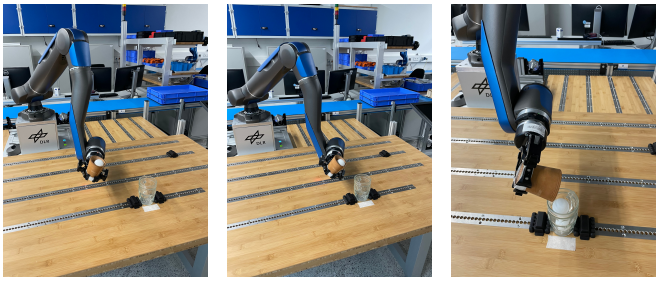

Fig. 1. We show that Shared Control Templates [8] – developed in our previous work for supporting users in executing tasks of daily living – are also effective in guiding RL. This leads to faster learning with fewer mistakes, as demonstrated on the pouring task with the SARA robot [9].

Of course, this stance raises the question of how such constraints should be represented and incorporated in the RL process. In our research on shared control – in particular *Shared Control Templates* [8] – 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 entails that exploration becomes safe.

The main contributions of this paper are: • Demonstrating that SCTs implement action- and transition-space shaping (Section IV-A) • Using SCTs to automatically generate functions for reward shaping (Section IV-B) • Demonstrating that applying SCTs to RL leads to more efficient learning in simulation and on a real robot, even with sparse rewards (Section VI) • Demonstrating that it is safer to adhere to explicit constraints, rather than to learn these constraints from their implicit encoding in shaped reward functions (also in Section VI). The key message of this paper, elaborated on in Section V, is that representing constraints implicitly in shaped reward function, can take as much human design effort as explicitly modeling these constraints, but that the latter leads to safer and faster learning.

Before following the structure interleaved above, we first discuss related work, and lay the foundations for our approach by describing different shaping approaches in RL, as well as Shared Control Templates (SCTs) in Section III.

II. Related Work

Considerable effort has been dedicated recently to constraining the search space in RL for robotics tasks. In [10], Cheng et al. proposed to use *Control Barrier Functions* (CBFs) [11] to avoid unsafe states for the robot while an RL agent explored the state-action space. Kanervisto et al. [12] conducted a thorough study on the effects of *action masking* which reduces action space by avoiding invalid actions in a particular state. Huang et al. [13] present an analysis on action masking concluding that it produces a valid policy gradient and scales better than punishing the invalid actions by a negative reward. Liu et al. [14] demonstrate that a constrained RL problem can be converted into an unconstrained one by performing RL on the tangent space of the constrained manifold. Results 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.

Another approach to reducing the dimensionality of the RL problem is to leverage frameworks for partial task specifications. Padalkar et al. [15], [16] proposed a partial task specification in the robot task space using the Task Frame Formalism [17], [18], 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.

SCTs have previously been used to automate tasks in [19] where policies are determined with local optimization; our aim here is to optimize the overall trajectory.

In summary, RL-SCT leverages pre-specified task knowledge to improve sample efficiency and simplify reward function design, facilitating direct learning on real robots and hence alleviating the need for simulation, a well-known problem in RL [20], [21].

III. Background

A. (Shaped) Markov Decision Processes (MDP)

A Markov Decision Process (MDP) is formalized with (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.

Reward Shaping (RS) [22] changes the reward function, for instance by changing the sparse terminal reward R into a dense, immediate reward R'. It does not change S, A, or T. Shaped rewards are more informative, and thus speed up learning.

Action Space Shaping (AS) [12] replaces the original action space with a reduced (state-dependent) action space. In *action masking* for instance [13], actions that are known to be invalid or suboptimal in certain states are excluded from the action space in those states, i.e. $A'_{s_i} \subset A$. Action space shaping does not modify S, T and R. As fewer actions need to be explored, such reduced action spaces speed up learning.

Consider a discrete MDP with $A = a_1, a_2$, and one wants to exclude the execution of a_2 in s_7. With Action Space Shaping this would be implemented as $A'_{s_7} = a_2$. Another approach to implementing this constraint would be to specify a reduced transition function T' so that $f_{T'}(s_7, a_2) = f_T(s_7, a_1)$, and have the agent learn with T' rather than T. That is, a_2 is not excluded from the action space, but executing a_2 in T' leads to the exact same effect as executing a_1 in T (and T'). This changes the transition function T, which maps actions to change in state, but leaves S, A and R unaffected. This approach, which we denote "Transition Function Shaping", is more general than Action Space Shaping. To the best of our knowledge, this has not been used in RL so far, also because explicitly excluding actions is more effective than remapping them. For this paper however the conceptual distinction is important, because it helps us make a clearer connection to two different components in the Shared Control Templates (Active Constraints and Input Mappings).

In summary, different approaches are available to *modify an MDP to speed up and facilitate learning*, i.e. Reward Shaping (modifies only R), Action Space Shaping (only A) or Transition Function Shaping (only T). One main hypothesis of this paper is that Action Space and Transition Function Shaping lead to faster learning than Reward Shaping, but, contrary to common belief, are often not more difficult to design.

B. 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") [23], which consists of a wheelchair with an articulated arm. EDAN typically takes a 3D input signal extracted from surface electromyography. It maps these inputs into 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 [8].

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 only 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) [8], which consist of several components. First, *Input Mappings* (IM), map the 3D input commands from EMG sensors to phase-dependent 6D end-effector motions. In the *Translational* and *Tilt* states in Fig. 2 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

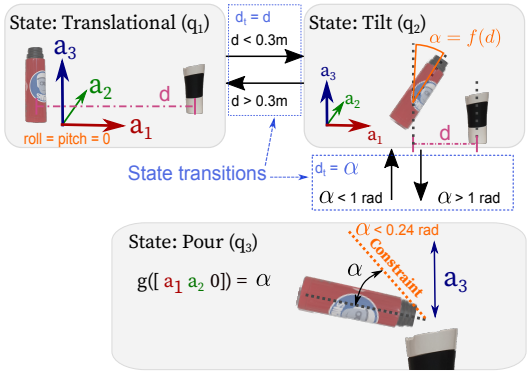

Fig. 2. The Shared Control Template (SCT) for pouring water. Different phases of a task are modeled as different SCT states (q_i) in a Finite State Machine ('Translational', 'Tilt', 'Pour'). In each state, an Input Mapping maps the 3D user input commands a_1, a_2, a_3 to 6D end-effector motions. Active Constraints (shown in orange text) limit the range of motion.

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.

Second, *Active Constraints* (AC) [24] constrain the end-effector pose that results by applying IM. As illustrated in Fig. 2 during the phase *Pour*, the tilt angle α of the bottle is constrained, to avoid excessive pouring. Another example of AC in *Tilt* state is the value of the tilting angle being a function of the distance from the target.

Formally, a SCT [8] is composed of a pre-defined number of states K. In each state q_i, $i = 1, ..., K$, an IM is a function which computes a desired displacement $\Delta\boldsymbol{H} \in \mathrm{SE}(3)$ in task space from an n-dimensional input $\boldsymbol{a}_t$ with $n <= 6$ at time step t (dropping the subscript i in q_i from now on):

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

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

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

After applying IM to obtain $\boldsymbol{H}^{\text{im}}_{t+1}$, geometric constraints can be enforced using AC. An AC [8] applies a projection of the form,

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

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 and volumetric constraints depending on the end-effector position when approaching an object.

Different states of a task require different IM and AC, as illustrated in Fig. 2. For this reason, they are organized in a finite-state machine, which monitors progress of the state in the form of scalars defined as transition distances d_t, and transitions to next states upon reaching a certain threshold of transition distances. In a SCT state q, the transition distance is a function of the current robot state $\boldsymbol{s}_t$:

$$d_t = f_q(\boldsymbol{s}_t). \tag{4}$$

Examples of transition distances are the distance to the target and the tilting angle as shown in blue rectangles in Fig. 2 along with the respective transition thresholds. For the former, Eq. (4) is written as $d_t = ||\boldsymbol{x}_{\text{th}} - \boldsymbol{x}_{\text{mug}}||$ where $\boldsymbol{x}_{\text{th}}$ and $\boldsymbol{x}_{\text{mug}}$ are the current thermos and mug tip positions respectively. For the latter, Eq. (4) takes the form $d_t = \alpha$, where α is the tilting angle towards the target.

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

IV. Shared Control Templates for Reinforcement Learning

The key insight in this paper 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 in this paper is to demonstrate this conceptually and empirically. In this section, we therefore link the different components of SCTs to the MDP shaping concepts described earlier.

A. Shared Control Templates for Action and Transition Function Shaping

In general a MDP (S, A, T, R) in the context of SCTs has the following components: S is a continuous state space consisting of the poses of relevant objects and the pose of the end-effector. A consists of continuous commands to the robot. T is the physics of the real world. The task to be completed is specified in the task-specific reward function R. The simplification of this MDP is done in three steps:

1) Using IM to map 3D user commands to 6D Cartesian commands, the action space A of the MDP is reduced to 3D. IM can accept inputs up to 6D, but in the tasks that we consider, see [8], more than 3 inputs are not needed. As IMs lead to different state transitions during different phases of the movement, we consider them an implementation of Transition Function Shaping.

2) The application of AC is conceptually equivalent with Action Space Shaping, in that not all actions defined by the IM have an effect[1].

3) Finally, R always has two components: 1) Primary costs, which may be shaped or sparse, terminal and task-dependent, and related to the task (Was water poured? Was the door traversed?) 2) Secondary costs, which are immediate, i.e. given at each time step, and task-independent, e.g. task execution duration or smoothness of motion. The next subsection explains reward shaping by SCT in detail.

The generic interface between SCT and RL components is illustrated in Fig. 3.

[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 [8].

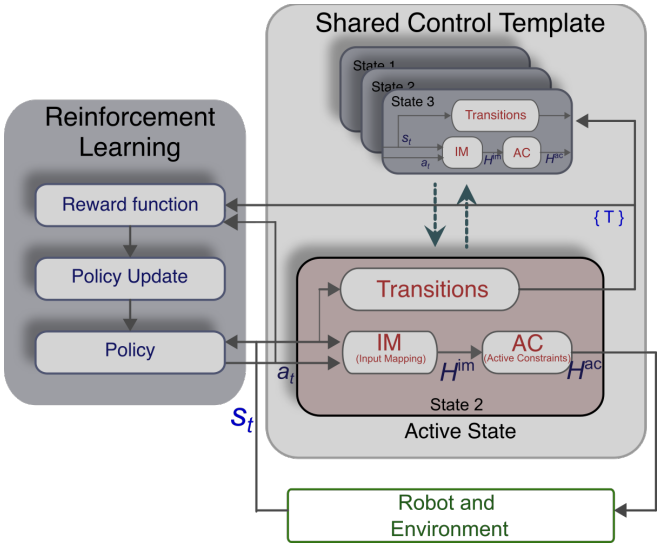

Fig. 3. Interface between a SCT and RL components. The input a_t to IM of SCT is computed by RL policy, s_t is the robot state (different from SCT states), $\{T\}$ is dictionary of transition distances to the neighboring SCT states.

B. Shared Control Templates for Reward Shaping

In the presence of a SCT, a reward function takes the form

$$r_t = k(d_{t-1} - d_t) - \boldsymbol{a}_t^T \boldsymbol{R} \boldsymbol{a}_t \tag{5}$$

where k is a constant, d is a transition distance to the next phase of task and $\boldsymbol{a}_t^T \boldsymbol{R} \boldsymbol{a}_t$ is an action cost with $\boldsymbol{a}_t$ an action, $\boldsymbol{R}$ a diagonal matrix with positive entries and r_T a termination reward given upon successful termination, otherwise null. The transition distance function in Eq. (4) is designed in such a way that it monotonically decreases as the robot moves towards the next progressive phase of the task. Hence, with appropriate values of k and $\boldsymbol{R}$, Eq. (5) generates positive rewards for moving towards task completion.

V. Explicit vs. Implicit Constraint Representation in RL

If constraints are implicitly encoded in the reward function, an agent will need to violate these constraints in order to learn them. However, in many robotic tasks, violating constraints can lead to physical damage. Moreover, reward functions also need to take into account the constraints arising from the geometry of the object and the robots' own manipulation capabilities. For a multiphase task, the reward function changes with each phase of the task, and hence its implementation is required to identify these phases.

SCTs address the above issues by: 1) precluding the robot from violating constraints with the Active Constraints, 2) identifying the phases and providing transition distances to the neighboring states as mentioned in Section III-B and shown in Fig. 2. These transition distances are the indicators of the progress of the current SCT state, and hence can be used for constructing shaped reward functions.

This can be better understood with the help of the comparison provided in Table I. With the help of the pouring liquid task in Fig. 2, Table I compares how the same components

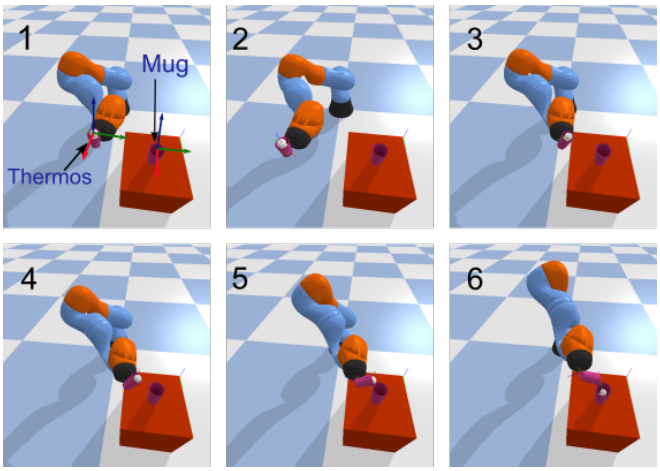

Fig. 4. Experimental setup for the pouring task with simulated KUKA IIWA holding the thermos and the target mug placed on the table with thermos tip and mug tip coordinate frames (x-axis in red, y-axis in green and z-axis in blue).

are modeled in either the reward function or in the IM, AC, and transition distances of a SCT and reward function of RL-SCT, during different states of the task. The reward components mentioned in Table I are used to construct the shaped reward function later presented in Eq. (6). Table I and Eq. (6) together highlight the importance of the intuition needed in choosing the robot state variables while constructing the reward function which affect the desired behavior is a task state (e.g. rotational motion in translation state results in spillage and hence should be penalized). SCTs can be constructed with the same intuition and can be even more systematic as the constraints are object-agnostic.

VI. Evaluation

We evaluated our proposed approach RL-SCT against RL without SCT on the pouring task with experiments in simulation using a KUKA IIWA (shown in Fig. 4), and further evaluated the performance of RL-SCT by learning the same task directly on the real 7-DoF SARA robot (shown in Fig. 1). 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. In the experiments, the liquid is replaced by two ping-pong balls with $4cm$ diameter. The task is considered successful only if both balls are successfully poured into the mug. 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.

A. Policy Representation and RL Algorithm

RL-SCT performs action and transition space shaping, independent of the underlying policy representation or RL algorithm used. Therefore, a variety of representations and algorithms can be used.

In our evaluation, we use a feed-forward neural network with 2 hidden layers for the RL policy representation $\boldsymbol{a}_t =$

TABLE I

COMPARISON OF KNOWLEDGE MODELED USING RL REWARD FUNCTION AND RL-SCT.

Task state	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 mug 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 motion in horizontal plane needed), AC: Stay above table and mug to avoid collision, Transition distance: Tilting angle

$\pi(\boldsymbol{s}_t)$. The state $\boldsymbol{s}_t$ is the 6D pose of the thermos tip expressed in the mug tip frame. In RL-SCT, as shown in Fig. 3, a policy generates a 3D action $\boldsymbol{a}_t$ which is the input to the step evaluation of SCT, which then computes end-effector pose from $\boldsymbol{a}_t$ as explained in Section III-B. In the case of RL without SCT, the policy generates the 6D end-effector velocity as action $\boldsymbol{a}_t$ which is used to compute the target end-effector pose. In both cases, the desired end-effector pose is fed to the Cartesian position controller running at 8KHz for the SARA robot and 100Hz for the simulated KUKA IIWA robot.

The weights of the neural network are initialized randomly. For learning with RL-SCT, the number of neurons in the hidden layers is 64 and for the RL without SCT it is 150. The underlying argument for using a bigger NN for RL without SCT is that the NN also needs to learn the task constraints and the policy for action space of higher DoF (6D in case of RL without SCT compared to 3D in case of RL-SCT). The above conclusion was reached empirically by performing multiple experiments, starting with 100 neurons and gradually increasing the size.

The policy learning algorithm used in our experiments is a state-of-the-art off-policy algorithm called Soft Actor-Critic [25] with parameters: 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. We use the Stable-Baseline3 [26] open-source implementation of SAC[2].

B. Reward functions

The baseline is a designed reward function, which is either shaped (RL-Shaped) or sparse (RL-Sparse). 'Shaped' means that the immediate reward contains information about the primary task to be solved, i.e. both balls ending up in the mug. 'Sparse' implies that only the terminal reward contains information about this task. Our proposed method based on SCTs is also evaluated with a shaped (RL-SCT-Shaped) and sparse (RL-SCT-Sparse) reward function.

Reward function for RL-Shaped. Reward shaping means that information about the ultimate task is encoded in the immediate reward at each time step. This requires careful design, and leads to a more complex design process and resulting reward function, as highlighted by the complexity of Eq. (6)-(9). First, the pouring task is divided in two phases, 1) transport thermos near the mug without spilling the liquid, and 2) pour the balls 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 mug avoiding any translation. These phases need to be identified correctly, and give the reward for not tilting the thermos in first phase and reward for tilting around the correct axis and not around the other axes.

The implementation of this reward function for RL-Shaped is given by Eq. (6)-(9), where j_t is the distance of the thermos tip from mug tip, and a_t^x, a_t^y and a_t^z are the angular positions, expressed as Euler angles (roll, pitch and yaw) of thermos tip in the mug tip frame (orientations of the tip frames are depicted in Fig. 4), and r_T is the terminal reward.

$$r_t = r_t^{\text{dist}} + r_t^{\text{tilt}}/4, \tag{6}$$

$$r_t^{\text{dist}} = \begin{cases} 15(j_{t-1} - j_t), & \text{if } j_t > 0.04 \\ 15(j_{t-1} - j_t) + 0.2, & \text{otherwise} \end{cases} \tag{7}$$

$$r_t^{\text{tilt}} = \begin{cases} -|\delta a_t^x| - |\delta a_t^y| - |\delta a_t^z|, & \text{if } j_t > 0.04 \\ 20\delta a_t^x - |\delta a_t^y| - |\delta a_t^z|, & \text{otherwise} \end{cases} \tag{8}$$

$$r_T = \begin{cases} 200, & \text{on successful termination} \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

Reward functions for RL-SCT-Shaped, RL-SCT-Sparse and RL-Sparse. The shaped reward function is automatically derived from the SCTs with Eq. (5). For the pouring liquid problem under consideration, this corresponds to :

$$r_t = 20(d_{t-1} - d_t) - \boldsymbol{a}_t^T \boldsymbol{a}_t, \tag{10}$$

$$r_T = \begin{cases} 20 & \text{on successful termination} \\ -20 & \text{on termination due to collision} \end{cases} \tag{11}$$

where d_t is the transition distance given by the SCT, see Fig. 2 and r_T is the terminal reward. The reward function for RL-Sparse and RL-SCT-Sparse is the same, except that the reward shaping term $20(d_{t-1} - d_t)$ is removed from Eq. (10).

C. Experiments in Simulation

Results. Fig. 5 shows the rate of success, spillage and collision with the environment during the learning process.

[2]https://stable-baselines3.readthedocs.io/en/master/modules/sac.html

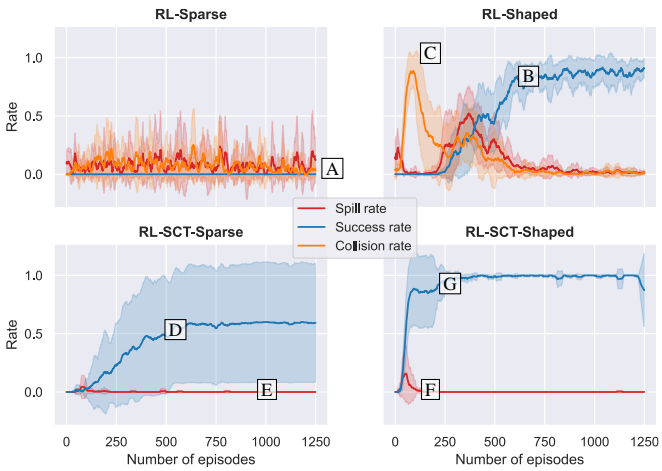

Fig. 5. Success, spill and collision rates vs number of episodes, in different experimental settings in simulation. Each point shows the average of 10 trials, together with one standard deviation.

The curves show the mean and standard deviation obtained from 10 different trials. 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. RL-Sparse is not able to learn the task in 1250 (maximum) episodes [A]. RL-Shaped converges within 600 episodes [B], but also shows very high spill rate and collision rate due to unconstrained exploration [C]. RL-SCT-Sparse converges within 600 episodes [D], shows very low spill rate and no collisions [E] due to the constraints, but the learning is unstable with some trials not converging. RL-SCT-Shaped converges within 250 episodes [G] showing the best performance overall, with small initial spill rate [F] and no collisions.

D. Real Robot Experiments

To validate the approach on a real robot, the pouring task was learned with the DLR SARA robot as shown in Fig. 1. The aim of the experiment is to learn the task using RL in the presence of SCT on the real robot. The task description, reward function and policy representation are the same as in the simulation. 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. A human observing the task provided the feedback about the success of the task.

Results. Fig. 6 shows the mean and standard deviation of learning progress of 5 independent trials on the real robot. The learning agent achieves a success rate of 1 within 65 episodes, on average, taking 16 minutes without considering the time taken for resetting the environment.

E. Discussion

Compared to the common approach of learning the task from scratch with a sparse reward function, our method is not only safer, but also learns the task faster. Learning with RL-SCT-Sparse is possible but, compared to RL-SCT-Shaped, it takes 600 episodes to learn the task in simulation against 250 episodes of RL-SCT-Shaped.

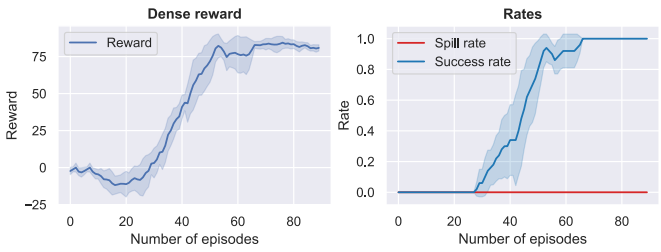

Fig. 6. Success rate, spill rate and reward for RL-SCT-Shaped on the real robot. Each point shows the mean and one standard deviation over 5 trials.

RL-SCT-Sparse also results in larger deviation in success rate over multiple trials with some trials not converging. Nevertheless, these results show that, with SCTs, the reward function can be further simplified to a sparse reward function without sacrificing the safe exploration ability. As the RL-SCT utilizes action space with reduced DoF and does not need to learn many environmental constraints, it can learn the policy with a neural network with a much smaller size ($<$0.5 times) than RL without SCT constraints.

VII. Conclusion

In this paper, we propose to guide reinforcement learning with constraints that are represented as Shared Control Templates. Indeed, 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 advantageous for robot reinforcement learning also. Our experiments show that the explicit representation of constraints leads to faster learning without the need to design a complicated reward function.

Our approach is tailored to the learning of tasks involving the use of objects with known constraints, which includes all the activities of daily living considered in our work on shared control [23]. It is less suited for learning intricate motor skills, such as those required for juggling, ball-in-cup, or locomotion.

Our work aims at providing users with SCTs that allow for effective and intuitive control of the robot, and where the resulting motion is efficient and looks natural. We expect that the RL-optimized SCTs will perform better in this respect than the non-optimized ones. Our future work aims to validate this expectation. Considering the importance of safety during in-contact tasks, our future work will also focus on tasks that involve contact dynamics. Finally, this paper demonstrated that learning can be done from scratch on real robots. We will use demonstrations to initialize the policy which should speed up learning further.

Acknowledgment

This work was partially funded by the European Union's Horizon 2020 Research and Innovation Programme under Grant Number 951992 (project VeriDream).

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.

REFERENCES

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[3] K. Ploeger, M. Lutter, and J. Peters, "High acceleration reinforcement learning for real-world juggling with binary rewards," *arXiv preprint arXiv:2010.13483*, 2020.

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

[5] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[6] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International journal of robotics research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[7] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[8] G. Quere, A. Hagengruber, M. Iskandar, S. Bustamante, D. Leidner, F. Stulp, and J. Vogel, "Shared control templates for assistive robotics," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 1956–1962.

[9] M. Iskandar, O. Eiberger, A. Albu-Schäffer, A. D. Luca, and A. Dietrich, "Collision detection, identification, and localization on the dlr sara robot with sensing redundancy," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021. [Online]. Available: https://elib.dlr.de/144160/

[10] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 3387–3395.

[11] A. Taylor, A. Singletary, Y. Yue, and A. Ames, "Learning for safety-critical control with control barrier functions," in *Learning for Dynamics and Control*. PMLR, 2020, pp. 708–717.

[12] A. Kanervisto, C. Scheller, and V. Hautamäki, "Action space shaping in deep reinforcement learning," *CoRR*, vol. abs/2004.00980, 2020. [Online]. Available: https://arxiv.org/abs/2004.00980

[13] S. Huang and S. Ontañón, "A closer look at invalid action masking in policy gradient algorithms," *CoRR*, vol. abs/2006.14171, 2020. [Online]. Available: https://arxiv.org/abs/2006.14171

[14] P. Liu, D. Tateo, H. B. Ammar, and J. Peters, "Robot reinforcement learning on the constraint manifold," in *Conference on Robot Learning*. PMLR, 2022, pp. 1357–1366.

[15] A. Padalkar, M. Nieuwenhuisen, S. Schneider, and D. Schulz, "Learning to close the gap: Combining task frame formalism and reinforcement learning for compliant vegetable cutting," 2020.

[16] A. Padalkar, M. Nieuwenhuisen, D. Schulz, and F. Stulp, "Closing the gap: Combining task specification and reinforcement learning for compliant vegetable cutting," in *International Conference on Informatics in Control, Automation and Robotics*. Springer, 2020, pp. 187–206.

[17] M. T. Mason, "Compliance and force control for computer controlled manipulators," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 11, no. 6, pp. 418–432, 1981.

[18] H. Bruyninckx and J. De Schutter, "Specification of force-controlled actions in the" task frame formalism"-a synthesis," *IEEE transactions on robotics and automation*, vol. 12, no. 4, pp. 581–589, 1996.

[19] S. Bustamante, G. Quere, K. Hagmann, X. Wu, P. Schmaus, J. Vogel, F. Stulp, and D. Leidner, "Toward seamless transitions between shared control and supervised autonomy in robotic assistance," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3833–3840, 2021.

[20] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of real-world reinforcement learning," *arXiv preprint arXiv:1904.12901*, 2019.

[21] E. Salvato, G. Fenu, E. Medvet, and F. A. Pellegrino, "Crossing the reality gap: a survey on sim-to-real transferability of robot controllers in reinforcement learning," *IEEE Access*, 2021.

[22] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *In Proceedings of the Sixteenth International Conference on Machine Learning*. Morgan Kaufmann, 1999, pp. 278–287.

[23] J. Vogel, A. Hagengruber, M. Iskandar, G. Quere, U. Leipscher, S. Bustamante Gomez, A. Dietrich, H. Höppner, D. Leidner, and A. O. Albu-Schäffer, "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*, 2020.

[24] S. A. Bowyer, B. L. Davies, and F. R. y Baena, "Active constraints/virtual fixtures: A survey," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 138–157, 2013.

[25] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[26] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, 2021. [Online]. Available: https://elib.dlr.de/146386/