2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)
Beijing, China, November 6-9, 2018

Safe-To-Explore State Spaces: Ensuring Safe Exploration in Policy Search with Hierarchical Task Optimization

Jens Lundell*, Robert Krug‡, Erik Schaffernicht†, Todor Stoyanov†, Ville Kyrki*

Abstract— **Policy search reinforcement learning allows robots to acquire skills by themselves. However, the learning procedure is inherently unsafe as the robot has no a-priori way to predict the consequences of the exploratory actions it takes. Therefore, exploration can lead to collisions with the potential to harm the robot and/or the environment. In this work we address the safety aspect by constraining the exploration to happen in safe-to-explore state spaces. These are formed by decomposing target skills (e.g., grasping) into higher ranked sub-tasks (e.g., collision avoidance, joint limit avoidance) and lower ranked movement tasks (e.g., reaching). Sub-tasks are defined as concurrent controllers (policies) in different operational spaces together with associated Jacobians representing their joint-space mapping. Safety is ensured by only learning policies corresponding to lower ranked sub-tasks in the redundant null space of higher ranked ones. As a side benefit, learning in sub-manifolds of the state-space also facilitates sample efficiency. Reaching skills performed in simulation and grasping skills performed on a real robot validate the usefulness of the proposed approach.**

I. Introduction

Real-time robot motion planning and control is a key ability for autonomous robots operating in unstructured environments. In recent years, policy search has emerged as one of the most promising approaches to enable simultaneous motion planning and execution. It is a branch of reinforcement learning (RL) focusing on optimizing parameterized controllers. Policy search scales gracefully to high dimensions [1], but typically requires handcrafted low-dimensional policy parameterizations to learn in as few roll-outs as possible [2], [3], [4], [5]. In addition, policy search requires a robot to explore its environment, an inherently unsafe process as it can lead to dangerous situations, such as collisions with obstacles, which might cause damage to the robot and/or the environment. Also, in many practical applications the number of samples needed to ensure successful skill learning is prohibitive.

Classical approaches overcome such limitations by enforcing conservative policy updates between iterations [6], [7] or by discouraging entering parts of the state space by imposing penalties [8]. These approaches, however, only limit the risk of collision but cannot completely remove it. Instead, we propose a method to implicitly form *safe-to-explore state spaces* (STESS) that focuses exploratory actions in arbitrary operational spaces to a collision free subset of the original state space. This is accomplished by first decomposing a robotic skill (*e.g.*, reach-to-grasp movements) into prioritized elemental tasks defined in potentially different operational spaces of choice together with a map (*i.e.*, a Jacobian) from operational space to joint space. Then, a whole-body real-time inverse kinematics scheme [9], [10] is used to obtain the corresponding joint velocities while ensuring that higher ranked tasks (*e.g.*, obstacle/joint limit avoidance) are prioritized over lower ranked movement tasks. Therefore we can limit exploration to obey a pre-defined operational space hierarchy. The control laws corresponding to lower ranked movement tasks are learned via policy search.

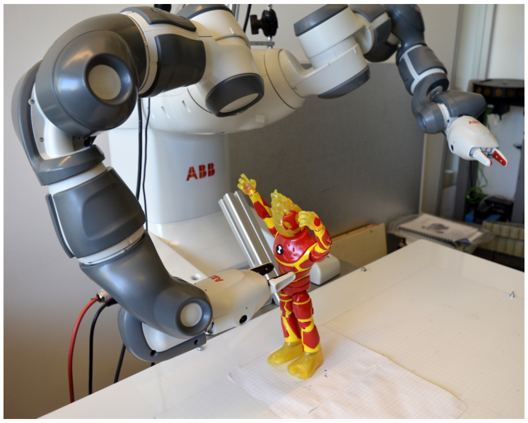


Fig. 1: System setup consisting of an ABB YuMi robot and a toy to be grasped.

The main contributions of this work are:

- a novel approach for ensuring safe exploration in policy search (Section II) consisting of a task-prioritized inverse kinematics framework (Section II-A) which facilitates safe learning of a time-invariant policy (Section II-B);
- an experimental evaluation (Section III) for a simulated reaching skill (Section III-A) and grasping skill on the platform shown in Fig. 1 (Section III-B) demonstrating that our method increases both safety, by removing the collision risk, and learning rate, by reducing the number of roll-outs before convergence.

Per se, our method is not tied to a specific policy search algorithm. In this work, we use a time-invariant version of the model-free Policy Improvements with Weighted Exploration (PoWER) algorithm [11].

*Intelligent Robotics Group, Department of Electrical Engineering and Automation, Aalto University, Finland. {firstname.lastname}@aalto.fi

†AASS Research Center, Örebro University, Sweden. {firstname.lastname}@oru.se

‡ Robotics, Learning and Perception lab, KTH Royal Institute of Technology, Sweden. rkrug@kth.se

This work was supported by Academy of Finland decision 314180 and by the Swedish Foundation for Strategic Research (SSF).

II. Approach

Our approach consists of a hierarchical inverse kinematics (IK) whole body control framework and a policy search component. The aim of the framework is to produce joint-level commands by inverting a set of concurrent hierarchical controllers which operate in arbitrary operational spaces. In this work, we use policy search to learn a subset of lower ranked controllers responsible for movement generation. The policy is optimized with policy search where the goal is to learn policy parameters $\boldsymbol{\theta}$ that maximize the expected reward $E_{\pi_{\boldsymbol{\theta}}}\left[\sum_{t=1}^{T} r(\mathbf{u}_t, \mathbf{x}_t)\right]$ over trajectories $\tau = \{\mathbf{x}_1, \mathbf{u}_1, \dots, \mathbf{x}_T, \mathbf{u}_{T-1}\}$, where $\mathbf{x}$ are states, $\mathbf{u}$ are actions and expectations are taken with respect to the policy's trajectory distribution $p(\tau) = p(\boldsymbol{x}_1) \prod_{t=1}^{T} p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t) p(\boldsymbol{u_t}|\boldsymbol{x}_t)$ where $p(\mathbf{x}_{t+1}|\mathbf{u}_t, \mathbf{x}_t)$ represent system dynamics. In this work, actions are operational space velocities as the IK framework produces joint velocities, while states and rewards are application specific (Section III).

A. Task-prioritized inverse kinematics

The following task-prioritized real-time inverse kinematics scheme is described in depth in our previous work [12], which in turn builds upon the method developed in [9]. Here, we give a brief summary for completeness as it is an essential component of the presented approach.

As in prior works [12], [10], we consider kinematic manipulator control with the goal of computing joint velocity commands $\dot{\boldsymbol{q}}$ by inverting a set of hierarchical tasks defined in different operational spaces. The obtained joint velocities are then executed by a low-level tracking controller. We describe the joint configuration of a manipulator with the vector $\boldsymbol{q}$. Following [9], we define task Jacobians via the derivatives of user-defined task functions $\boldsymbol{e}(\boldsymbol{q})$. To give an example, consider a static plane described by its unit normal $\boldsymbol{n}$ and offset d. The task of moving an end-effector point $\boldsymbol{p}(\boldsymbol{q})$ onto this plane can be achieved by driving the residual of the function $\boldsymbol{e}(\boldsymbol{q}) = \boldsymbol{n}^T \boldsymbol{p}(\boldsymbol{q}) - d$ to zero. Here, the corresponding operational space is defined by the normal $\boldsymbol{n}$ and the Jacobian mapping from joint space to operational space is given by $\boldsymbol{J}(\boldsymbol{q}) = \boldsymbol{n}^T \frac{\partial \boldsymbol{p}(\boldsymbol{q})}{\partial \boldsymbol{q}}$. A desired task evolution can be imposed via control laws (policies) $\dot{\boldsymbol{e}}^*$ achieving, $e.\,g.$, exponential convergence by setting $\dot{\boldsymbol{e}}^* = -\lambda \boldsymbol{e}$ with $\lambda \in \mathbb{R}_+$. In the following, we drop dependencies on $\boldsymbol{q}$ for notational convenience. For a single equality task, finding joint space controls corresponds to solving the following least-squares problem

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

which can easily be done via a pseudoinverse of $\boldsymbol{J}$. In order to allow for inequality tasks and without loss of generality we henceforth use a general task formulation with upper bounds

$$\boldsymbol{J} \dot{\boldsymbol{q}} \leq \dot{\boldsymbol{e}}^*. \tag{2}$$

As shown in [9], this allows for lower bounds $\boldsymbol{J} \dot{\boldsymbol{q}} \geq \dot{\boldsymbol{e}}^*$, double bounds $\underline{\dot{e}}^* \leq \boldsymbol{J} \dot{\boldsymbol{q}} \leq \dot{\bar{\boldsymbol{e}}}^*$, and equalities $\boldsymbol{J} \dot{\boldsymbol{q}} = \dot{\boldsymbol{e}}^*$, by reformulating them, respectively, as $-\boldsymbol{J} \dot{\boldsymbol{q}} \leq -\dot{\boldsymbol{e}}^*$, $\begin{bmatrix} -\boldsymbol{J} \\ \boldsymbol{J} \end{bmatrix} \dot{\boldsymbol{q}} \leq \begin{bmatrix} -\underline{\dot{\boldsymbol{e}}}^* \\ \dot{\bar{\boldsymbol{e}}}^* \end{bmatrix}$, and $\begin{bmatrix} -\boldsymbol{J} \\ \boldsymbol{J} \end{bmatrix} \dot{\boldsymbol{q}} \leq \begin{bmatrix} -\dot{\boldsymbol{e}}^* \\ \dot{\boldsymbol{e}}^* \end{bmatrix}$. If the constraint in (2) is infeasible, a least squares solution for $\dot{\boldsymbol{q}}$ as in (1) can be found by including the slack variable $\boldsymbol{s}$ among the decision variables and solving the Quadratic Program (QP)

$$\begin{aligned} &\min_{\dot{\boldsymbol{q}}, \boldsymbol{s}} \ \frac{1}{2} \left\| \boldsymbol{s} \right\|^2 \\ &s.t. \ \boldsymbol{J} \dot{\boldsymbol{q}} \leq \dot{\boldsymbol{e}}^* + \boldsymbol{s}. \end{aligned} \tag{3}$$

To enforce a hierarchy of $p = 1, \dots, P$ priority levels, we stack all task Jacobians of equal priority p in a matrix $\boldsymbol{A}_p$. Also, all corresponding operational space controls $\dot{\boldsymbol{e}}^*$ are stacked in a vector $\boldsymbol{b}_p$ to form one constraint of the form $\boldsymbol{A}_p \leq \boldsymbol{b}_p$ per hierarchy level. The aim is to sequentially satisfy a constraint in the least-square sense while solving for the subsequent constraints of lower priority in the null space of the previous constraint, such that the previous solution is left unchanged. Therefore, the following QP, with the previous slack variable solutions $\boldsymbol{s}_i$ frozen between iterations, needs to be solved for $p = 1, \dots, P$

$$\begin{aligned} &\min_{\dot{\boldsymbol{q}}, \boldsymbol{s}_p} \ \frac{1}{2} (\left\| \boldsymbol{s} \right\|^2 + \lambda \left\| \dot{\boldsymbol{q}} \right\|^2) \\ &s.t. \ \boldsymbol{A}_i \dot{\boldsymbol{q}} \leq \boldsymbol{b}_i + \boldsymbol{s}_i^*, \ i = 1, \dots, p-1 \\ &\qquad \boldsymbol{A}_p \dot{\boldsymbol{q}} \leq \boldsymbol{b}_p + \boldsymbol{s}_p. \end{aligned} \tag{4}$$

Here, $\lambda \in \mathbb{R}_+$ is used to regularize the solution in order to avoid large velocities due to singularities. The control vector $\dot{\boldsymbol{q}}$ is obtained from the final (P-th) solution of (4).

The main underlying mechanism of the hierarchical framework outlined above is to solve lower ranked tasks as good as possible (in the regularized least-square sense) in the null space of higher ranked task. We exploit this property to incorporate prior knowledge for, $e.\,g.$, obstacle avoidance or desired end-effector alignments. These are posed as tasks on a higher priority level then movement tasks for which control policies are learned as described below. Therefore, no matter what exploration strategy is chosen for learning, by definition the resulting motion can not violate higher prioritized tasks for avoidance and thus a safe behavior is guaranteed. Also, as we will demonstrate in Section III, posing additional tasks encapsulating desired behaviors prunes the null space remaining for exploration on lower task levels and thus the policy search space. Therefore, learning in the remaining obstacle free low-dimensional space also facilitates learning rate.

B. Policy search in operational spaces

As discussed above, the tasks we consider in this work are described by Jacobian maps from joint space to operational spaces of choice, and corresponding operational space controllers. Here we describe how to learn the corresponding control laws (policies) $\dot{\boldsymbol{e}}^*$ in operational space. Following the work in [5], a natural choice to represent such a policy is a normalized Radial Basis Function (RBF) network

$$y(\boldsymbol{x};\boldsymbol{\theta}=[\Theta_1,\dots,\Theta_N]) = \frac{\sum_{i=1}^{N}\Theta_i\rho(\|\boldsymbol{x}-\boldsymbol{c}_i\|)}{\sum_{i=1}^{N}\rho(\|\boldsymbol{x}-\boldsymbol{c}_i\|)}, \tag{5}$$

where $\boldsymbol{x}$ is the state of the system (*e.g* location of the end-effector) and N is the number of basis functions, each of which is centered at individual positions $\boldsymbol{c}_i$ and weighted by Θ_i. The function $\rho(\|\boldsymbol{x}-\boldsymbol{c}_i\|) = \exp(-\frac{\|\boldsymbol{x}-\boldsymbol{c}_i\|^2}{2\sigma^2})$ denotes a Gaussian kernel with variance σ. In this work, σ is set to $\frac{\mathrm{d}_{\max}}{\sqrt{2N}}$, where $\mathrm{d}_{\max}$ is the maximum distance between kernel centers, as this finds a compromise between locality and smoothness [13].

A RBF network is essentially a single hidden layer neural network with ρ as the activation function, $\boldsymbol{x}$ as the input, and Θ_i as the linear output weights. It represents a mapping from (potentially high-dimensional) input data to a scalar value $y : \boldsymbol{x} \in \mathbb{R}^n \mapsto \mathbb{R}$. In this work, the RBF network constitutes a local, time-invariant policy parameterization that represents operational space policies

$$\dot{e}^*_\pi(\boldsymbol{x};\boldsymbol{\theta}) = y(\boldsymbol{x};\boldsymbol{\theta}), \tag{6}$$

where $\boldsymbol{x}$ is the input and $\boldsymbol{\theta}$ is the parameter vector. Exploration takes place directly in parameter space

$$\dot{e}^*_\pi(\boldsymbol{x};\boldsymbol{\theta}+\boldsymbol{\epsilon}_t) = y(\boldsymbol{x};\boldsymbol{\theta}+\boldsymbol{\epsilon}_t), \tag{7}$$

where $\boldsymbol{\epsilon}_t \in \mathcal{R}^N$ represent Gaussian noise with variance Σ.

The policy parameters $\boldsymbol{\theta}$ are iteratively updated using a time-invariant version of the PoWER algorithm [11]. In each iteration, stochastic roll-outs are executed by adding Gaussian noise to the parameter vector. The parameters are then updated as a convex combination of noise and their respective reward, where noise that resulted in higher rewards contributes more to the overall update. An in depth explanation of the algorithm is given in our previous work [14].

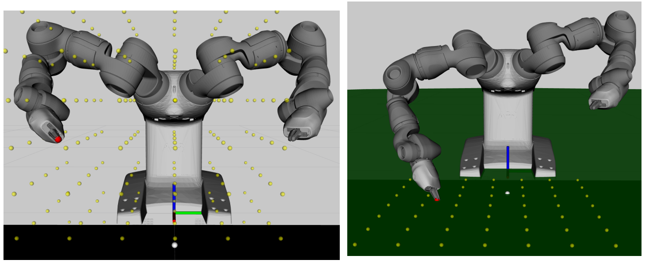
Noise is only sampled once per roll-out and added to the most active kernel every time step. To further control the magnitude of noise, we follow [14] and use a modified covariance matrix $\Sigma = \gamma\beta\mathbf{I}$. Here, $\mathbf{I}$ is the identity matrix, γ is a constant controlling the initial magnitude and β is given by

$$\beta = \frac{1}{\sum_{k=1}^{K} r_k^2}. \tag{8}$$

The sum in (8) is taken over the K best rewards $\{r_k\}_{k=1}^{K}$. Parameter β essentially lowers the amount of exploration required once a policy starts converging towards higher rewards.

III. Experimental Evaluation

We validate our approach by means of two illustrative examples: reaching (Section III-A) and grasping (Section III-B). To this end, we use one of the 7 DOF arms of the platform depicted in Fig. 1. The reaching experiment was simulated in Gazebo, while the grasping experiment was executed on the actual robot.



(a) Only the reaching task is specified.

(b) In addition to the reaching task a projection task forcing the red sphere onto the green plane is added.

Fig. 2: Reaching experiment: The yellow spheres represent the Gaussian kernels in (5), the red sphere the end-effector point and the white sphere the target position. The black surface indicates an obstacle. (Best viewed in color).

The RBF network kernels are evenly distributed over the whole search space. The total number of policy parameters depend on the number of kernels and on the number of control policies to learn, as each policy have their own set of policy parameters and the number of policies grows with the search space dimensionality ($e.g.$ one policy for a 1D search space, two policies for 2D etc.). Initially all policy parameters are set to zero.

A. Reaching

In this experiment, the main goal is to analyze the impact optional higher ranked tasks have on safety when learning movement policies on lower priority levels. As a side objective, we also want to study the effect on learning rate when adding additional higher ranked tasks that are not required for ensuring safety. To achieve this we chose a reaching experiment where the objective for the robot, illustrated in Fig. 2, is to colocate the red end-effector point $\boldsymbol{p}$ with the white target point $\boldsymbol{k}$ defined in the environment. Moreover, to actually validate the impact higher ranked tasks have on the learned policy, the experiment was split in three sub-experiments with varying prior knowledge.

In the first sub-experiment (Exp. 1), shown in Fig. 2a, no additional tasks were encoded besides the reaching task. Consequently, the STESS defaulted to 3D Cartesian space including the black obstacle. The second sub-experiment (Exp. 2) extends Exp. 1 by encoding a task that avoids obstacles on a higher priority level but maintains the same search space. In addition to the constraints in Exp. 2, the final sub-experiment (Exp. 3), shown in Fig. 2b, encodes a task that forces end-effector point $\boldsymbol{p}$ to also lie on the green plane, effectively reducing STESS from 3D to 2D.

Here, the operational space of the reaching task, posed at the lowest hierarchy level, corresponds to 3D Cartesian space. Thus, the corresponding task Jacobian in (2) is simply the manipulator Jacobian with respect to $\boldsymbol{p}$. For each Cartesian dimension, an operational space policy is learned where the state $\boldsymbol{x}$ in (7) corresponds to the end-effector position.

Each experiment ran for 10 trials with a maximum of 300 roll-outs (including 15 initial roll-outs before policy update)

	Exp. 1	Exp. 2	Exp. 3
Collision during training	**4/10 (40%)**	**0/10 (0%)**	0/10 (0%)
Converged trials	0/10 (0%)	0/10 (0%)	**10/10 (100%)**

TABLE I: Collision frequency and learning performance.

per trial. The importance sampler used the five best roll-outs for each policy update. If a collision occurred during training, the corresponding trial was deemed unsuccessful. The immediate reward function at time t is

$$r(\dot{\boldsymbol{q}}_t, \boldsymbol{p}_t) = \begin{cases} \exp\left(-\alpha_1 \|\dot{\boldsymbol{q}}_t\|_1 - \alpha_2 d^2\right), & \text{if } t = \mathrm{T} \\ \exp\left(-\alpha_1 \|\dot{\boldsymbol{q}}_t\|_1\right), & \text{otherwise} \end{cases} \qquad (9)$$

where $\dot{\boldsymbol{q}}$ are joint velocities and $\boldsymbol{p}_t$ is the 3D Cartesian position of the gripper. The term $d = \|\boldsymbol{p}_\mathrm{T} - \boldsymbol{k}\|_2$ is the distance between the gripper $\boldsymbol{p}$ and the target point $\boldsymbol{k}$ at the end of the roll-out. The parameters α_j are individual weighting factors chosen as $\alpha_1 = 0.001$ and $\alpha_2 = 10$.

In Exp. 1 and 2 we chose 249 kernels and, with a 3D search space, the total number of policy parameters were 747. In Exp. 3, on the other hand, the search was carried out in 2D and therefore we chose only 49 kernels, resulting in 98 policy parameters. For Exp. 1 and 2, the initial exploration parameter $\gamma = 4$, while for Exp. 3 $\gamma = 0.001$. The parameter β controlling the noise level was calculated using the reward from the 10 best roll-outs.

Figure 3 shows the convergence rates of Exp. 1, 2, and 3. These, together with the data presented in Table I, indicate that obstacle avoidance eliminates the possibility for collisions during exploration, but does not improve the actual learning rate. However, encoding additional tasks that reduced the search space to 2D allowed the policy to converge in 100% of the trials after an average of 84.4 roll-outs.

The main objective in this experiment was to study the implication higher ranked tasks have on learning safety and, as the results indicate, our method is indeed capable of forming STESS that are collision free. However, based on the results from Exp. 1 and 2, it seems impossible to learn RBF network policies that successfully fulfill the task in high dimensional search space, probably originating from poor scaling of RBF networks to high dimensional state spaces [15]. In such cases special handcrafted policy representations are preferred [2], [3], [4], [5]. Nevertheless, as indicated in Exp. 3, adding additional domain knowledge projecting STESS to a lower dimensionality allowed general RBF network policies to converge in all trials. In terms of achieving higher learning rate it is possible to use other policy search algorithms as our method do not default to any particular one. Then again, based on the results in Exp. 3, the most significant learning boost is achieved by further limiting the STESS with additional constraints, something which is further demonstrated in the following grasping experiment.

B. Grasping

In this experiment, the goal is to demonstrate our method working on a real robot in the sense that it learns policies for grasping nontrivial objects after few roll-outs. To achieve this, the experiment is to grasp the toy and box displayed in Fig. 4 with the robot shown in Fig. 1. The results presented in the previous section indicate that prior data in form of additional tasks allows for both faster and safer policy search. Therefore, we attempt to maximize the amount of prior knowledge in order to reduce the search space. To this end, we use tasks forming a so-called grasping envelope as defined in our previous work [16]. These grasping envelopes are intended to capture grasping strategies observed in humans [17]. In [16] we used them to successfully produce robust grasps for a wide selection of objects including bottles, boxes, and plush toys.

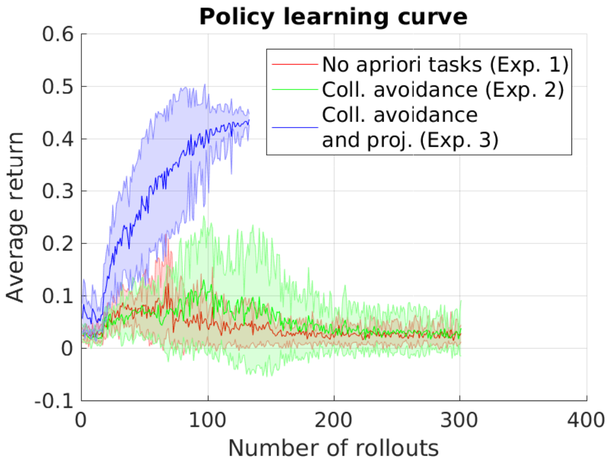


Fig. 3: The average return over the number of roll-outs for each experiment.

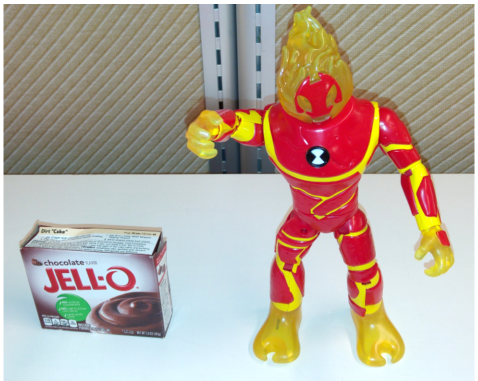


Fig. 4: Shows the two test objects used in the grasping experiment. The robot could only grasp the toy at specific heights from the smaller edges, whereas the box was graspable at all heights from shorter edges.

In this experiment orientation constraints of the gripper are consistent with the grasping envelope in [16], that is: the gripper's vertical axis z (see Fig. 5) is enforced to align with the cylinder axis z_0, while the gripper's approach axis x is to point towards the cylinder axis x_0. Moreover, during the roll-out the end-effector point $\boldsymbol{p}$ is constrained to lie on a larger blue cylinder as shown in Fig. 5b, while after the roll-out the same $\boldsymbol{p}$ is constrained to the smaller green cylinder displayed in Fig. 5c, thus allowing the gripper to enclose the object. Finally, when grasping the toy in Fig. 5a the gripper is constrained to lie between the upper and lower black planes, while for grasping the box in Fig. 5b it is constrained to lie on a single black plane. Together these constraints leave redundancy for the gripper to move both vertically and horizontally around the manifold when grasping the toy, *i.e.* a 2D search space, and only horizontally when grasping the box, effectively reducing the search space from 2D to 1D.

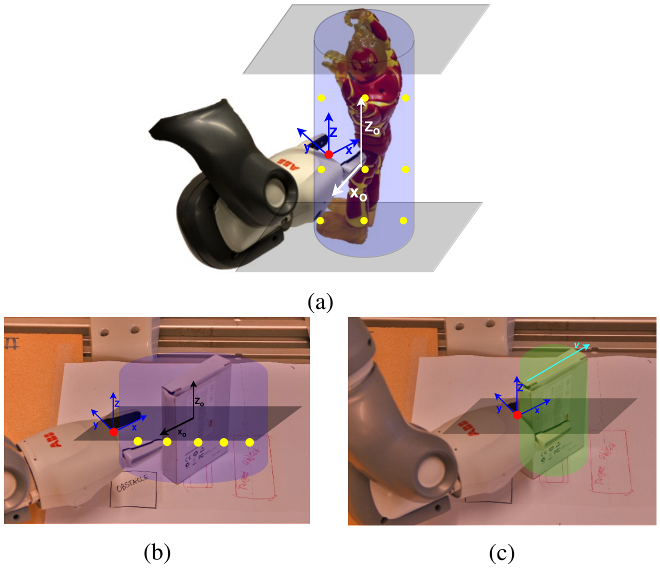
The movement tasks forcing the gripper to converge onto the grasp manifold were predefined using simple controllers of the form $\dot{\boldsymbol{e}}^* = -\lambda \boldsymbol{e}$. On the lowest priority level (and thus in the null space of all higher ranked task), we learned policies modulating the approach motion. In the case of grasping the toy, the aforementioned predefined reaching and alignment tasks leave the end-effector free to move in an operational space which is tangent to the cylinder shown in Fig. 5a. Therefore, to learn a policy modulating the movement in this redundant space, we define a task whose Jacobian maps to the cylinder's tangent space. In the case of grasping the box, the redundant space is further constrained to the pre-defined grasping plane illustrated in Fig. 5b. Thus, in this case we form an operational space mapping via a corresponding Jacobian which maps to the tangent space of the cylinder lying in the plane. When learning the modulation policy, exploration happens then only in this reduced redundant space. Based on this, the state $\boldsymbol{x}$ in (7) when grasping the toy is the (x, y, z) position of the end-effector while for the box reduces to (x, y) position.

Due to the limited opening size of the gripper, it could only grasp the toy and the box in Fig. 4 close to the shorter edges. Additionally the odd shape of the toy made it graspable only at specific heights, further complicating the situation. Hence, a suitable reward function for both objects needs to guide the learning towards either of the short sides. We devised such a reward function by reusing (9). Here however, the parameter d represents the shortest distance between end-effector point $\boldsymbol{p}$ and a vector $\boldsymbol{v}$ passing parallel to one of the principal components of the object (see Fig. 5c). The reward function for grasping is given by

$$r(d, I) = \exp\left(-\eta_1 d - \eta_2 I\right) \tag{10}$$

where I is a binary variable indicating a successful (0) or unsuccessful (1) grasp. The term d is calculated as $d = \|\boldsymbol{p}_T - \boldsymbol{v}\|$ where $\boldsymbol{p}_T$ is the 3D Cartesian position of the gripper at the end of the roll-out. The parameters η_1 and η_2 were, respectively, set to 700 and 1 when grasping the box, and 7 and 1.5 when grasping the toy. Grasp failure is detected if the gripper opening joint value crosses a predefined threshold indicating an empty grasp.

In this work we predefined a principal component as the preferred grasp direction $\boldsymbol{v}$, but it could also be determined using principal component analysis on a discrete representation of the target object's geometry. This, however, we considered not to be in the scope of the presented work and the literature offers many techniques to accomplish this task [18]. The reward function effectively guides the grasp towards one principal component of the object which, based on human grasping strategies, have been demonstrated to produce robust grasps [17]. Although the reward function is not the core contribution in this work, it is potentially useful for other learning tasks.



(a)

(b)

(c)

Fig. 5: Grasping experiment: (a) The grasp manifold consists of a larger blue cylinder and the black grasping planes. The red end-effector point $\boldsymbol{p}$ is forced to lie on the blue cylinder and between the black planes in (a) and on the black plane in (b). In (c) the red end-effector point is constrained to lie on the smaller green cylinder. The x and z axis of the gripper are aligned with the corresponding x_o and z_o axis of the cylinder. In (a) and (b), the yellow points indicate the RBF kernel distribution in (5). Furthermore, one principal component of the target object is indicated by the cyan vector $\boldsymbol{v}$ in (c).

For the grasping experiment the number of trials, maximum roll-outs, and number of roll-outs used by the importance sampler was the same as in the previous experiments. The number of initial roll-outs before policy iteration was set to 8. By trial and error, we found that 24 kernels were enough to learn a good policy for grasping the toy and 11 for grasping the box. As the search space was 2D for the toy and 1D for the box the number of policy parameters were, respectively, 48 and 11. The exploration parameter γ when grasping the toy was 0.0006 for kernels modulating vertical (z) movements and 0.00009 for modulating horizontal (x and y) movements. The same exploration parameters when only learning horizontal movements for grasping the box was set to 0.001. In both cases the seven best rewards were used to calculate β.

The convergence rate of the policy search is illustrated in Fig. 6. In both cases the policy converged in all 10 trials after 16.3 roll-outs on average when grasping the box and 39 when grasping the toy (including 8 initial roll-outs before policy optimization), after this the robot was able to consecutively grasp the objects. Fig. 7 shows an example executing the initial unlearned policy as well as the final learned policy.

The results clearly indicate that forming STESS with our method enables a real robot to learn both safely and

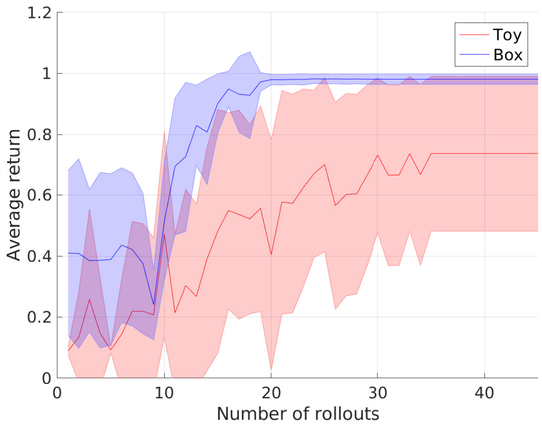


Fig. 6: The average return over the number of roll-outs for grasping the box and the toy. The reason the rewards for grasping the toy and box differs so much is because of different weighing parameters η_j in (10).

efficiently to grasp even very complex objects. As already mentioned in the reaching experiment adding additional tasks boost the learning significantly, a claim further strengthened by the 49% reduction in number of roll-outs before convergence when learning to grasp the box opposed to the toy. Although part of the fast learning rates presented in Fig. 6 originates from well tuned meta-parameters such as exploration rate γ and decay β, they are not crucial for the method to work, as poorly tuned parameters will not prohibit learning but instead just slow it down. In conclusion, the most important factor for learning is prior knowledge added to the system in the form of higher ranked tasks, and by adding tasks intended to mimic human grasping strategies not only reduced the search space to a collision free low-dimensional representation, but also led to successful grasping.

IV. Related Work

The core ingredients of our work are safe policy search, operational space learning and the normalized RBF network policy. Similar ideas for safe exploration as presented here was recently studied in [19] where they force output actions from a neural network to safe spaces by taking closest actions that satisfy specific constraints. In that work, however, they only constrained a single QP at each iteration while we impose hierarchical constraints allowing greater flexibility. Other methods trying to ensure safe exploration enforce conservative policy updates between iterations [6], [7], [20], [21], [11]. These methods are typically combined with low-dimensional policy representations that are initialized on prior data, $e.g.$, human demonstrations or trajectories optimized using an initial model [4], [5], [3]. Limiting policy updates between iterations lowers the probability of straying into unexplored state space, but does not provide any guarantees [22]. The associated risk is also evidenced by the results reported in Section III: if the search space includes obstacles, collisions can and, in general, will occur between the robot and obstacles. Another approach to safe behavior is to directly discourage entering part of the state space by inferring penalties [8]. These penalties are incorporated as a term in the reward function that punishes the robot if it comes in proximity to obstacles. These method bears resemblance to classical potential field methods to obstacle avoidance [23]. Similar to potential fields, while significantly reducing the collision rate, state space penalization does not provide guarantees while introducing additional tuning parameters. In comparison, our method can completely remove the collision risk during exploration without additional tuning parameters by posing appropriate avoidance tasks.

Regarding the policy representation used in this paper, previous work in value-based RL uses RBF networks to approximate the value function [24], while in policy search it is a vital part of the popular Dynamic Movement Primitives [3]. Only recently was it used as a standalone policy representation optimized to learn a limit-cycle walking gait in simulation [5]. This, in addition to our work, indicates that RBF networks are applicable policy representations for learning a variety of tasks.

In terms of operational space learning, we can also consider prior methods that learn the maps from joint space to operational space ($i.e.$, Jacobians) together with the corresponding control laws [25]. In that work, the controller is represented as a locally linear policy which is optimized with reward weighted regression. If no prior information regarding a skill is known, learning the Jacobians and potentially the dynamics in operational space can be valuable. It would be interesting to extend our framework to account for these possibilities.

V. Conclusion and Future Work

We presented a method for safe and sample-efficient learning of policies in arbitrary operational spaces. The key concept is a method to implicitly form *safe-to-explore state spaces* for policy search by decomposing a skill into elemental sub-tasks focusing exploratory actions to a collision free subspace of the original search space. By decomposing a skill of interest into elemental sub-tasks, our method allows to encode prior knowledge in form of task constraints. Movement policies are learned in the null space of higher ranked tasks which enforce, *e.g.*, obstacle avoidance or desired end-effector alignments. Thus, by construction, our approach ensures safe exploration. Also, the size of the search space available for learning task policies can be controlled by posing higher ranked tasks which implicitly prune the redundant space remaining for exploration. We provide an experimental evaluation by means of a simulated reaching task demonstrating that our method allows safe, collision-free movement policy learning while simultaneously leading to an increased learning rate. Furthermore, we evaluated our approach via grasping tasks executed on a real robot, achieving fast policy convergence even for complex objects.

To speed up learning, we predefine some parameters (kernel placement and width) of the chosen policy representation. This limits the applicability of the method, but also opens up interesting future research avenues. One option is to incorporate visual feedback of the scene [16] in order

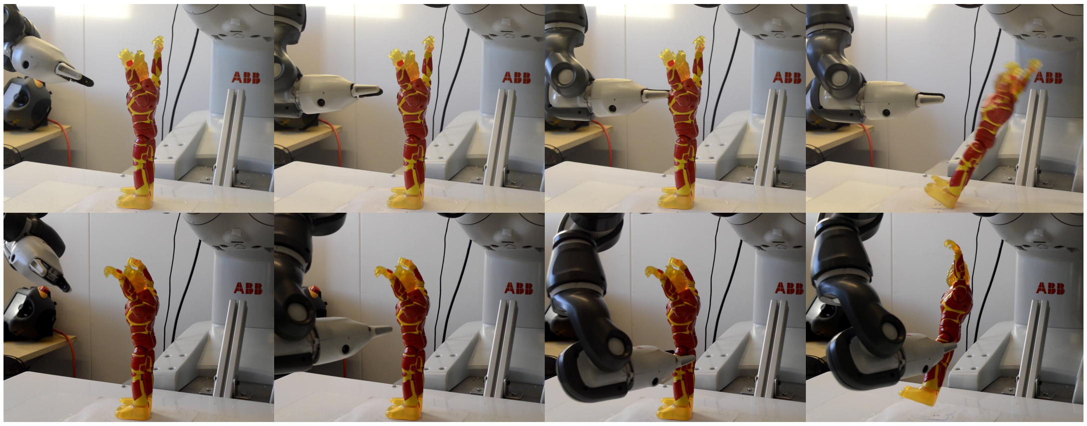


Fig. 7: Sequence of images from execution of the initial unlearned policy (top figure) and the final learned policy (bottom figure) for graping the toy. It is clearly visible that grasping the toy requires precise vertical and horizontal placement of the manipulator.

to decide on the number of kernels and their individual placement. Another option is to treat kernel centers and widths as additional policy parameters and to learn them using model-free policy search. Also, while our framework is not tied to a specific policy representation, the policies learned in this work are local. Therefore, they will not generalize well to new unseen situations, such as different object rotations. To allow generalization, one option is to learn global policy representations such as (deep) neural networks. Also, it would be interesting to train several local policies in varying settings and to combine them into a global model allowing inter- and extrapolation [14].

References

[1] S. Schaal, "Dynamic movement primitives-a framework for motor control in humans and humanoid robotics," in *Adaptive motion of animals and machines*. Springer, 2006, pp. 261–280.

[2] J. Peters and S. Schaal, "Reinforcement learning by reward-weighted regression for operational space control," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 745–750.

[3] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 2. IEEE, 2002, pp. 1398–1403.

[4] M. Kalakrishnan, L. Righetti, P. Pastor, and S. Schaal, "Learning force control policies for compliant manipulation," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 4639–4644.

[5] D. S. Feirstein, I. Koryakovskiy, J. Kober, and H. Vallery, "Reinforcement learning of potential fields to achieve limit-cycle walking," *IFAC-PapersOnLine*, vol. 49, no. 14, pp. 113–118, 2016.

[6] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural networks*, vol. 21, no. 4, pp. 682–697, 2008.

[7] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015, pp. 1889–1897.

[8] M. P. Deisenroth, C. E. Rasmussen, and D. Fox, "Learning to control a low-cost manipulator using data-efficient reinforcement learning," *Robotics: Science and Systems*, 2011.

[9] O. Kanoun, F. Lamiraux, and P.-B. Wieber, "Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 785–792, 2011.

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

[11] J. Kober and J. R. Peters, "Policy search for motor primitives in robotics," in *Advances in neural information processing systems*, 2009, pp. 849–856.

[12] R. Krug, T. Stoyanov, V. Tincani, H. Andreasson, R. Mosberger, G. Fantoni, and A. J. Lilienthal, "The next step in robot commissioning: Autonomous picking and palletizing," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 546–553, 2016.

[13] N. Benoudjit, C. Archambeau, A. Lendasse, J. A. Lee, M. Verleysen *et al.*, "Width optimization of the gaussian kernels in radial basis function networks." in *ESANN*, vol. 2, 2002, pp. 425–432.

[14] J. Lundell, M. Hazara, and V. Kyrki, "Generalizing movement primitives to new situations," in *Conference Towards Autonomous Robotic Systems*. Springer, 2017, pp. 16–31.

[15] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics New York, 2001, vol. 1.

[16] T. Stoyanov, R. Krug, R. Muthusamy, and V. Kyrki, "Grasp envelopes: Extracting constraints on gripper postures from online reconstructed 3d models," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 885–892.

[17] R. Balasubramanian, L. Xu, P. D. Brook, J. R. Smith, and Y. Matsuoka, "Physical human interactive guidance: Identifying grasping principles from human-planned grasps," *IEEE Transactions on Robotics*, vol. 28, no. 4, pp. 899–910, 2012.

[18] Y.-S. Liu and K. Ramani, "Robust principal axes determination for point-based shapes using least median of squares," *Computer-Aided Design*, vol. 41, no. 4, pp. 293–305, 2009.

[19] T.-H. Pham, G. De Magistris, and R. Tachibana, "Optlayer-practical constrained optimization for deep reinforcement learning in the real world," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6236–6243.

[20] H. B. Ammar, R. Tutunov, and E. Eaton, "Safe policy search for lifelong reinforcement learning with sublinear regret," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015, pp. 2361–2369.

[21] J. Peters, K. Mülling, and Y. Altun, "Relative entropy policy search," in *AAAI*. Atlanta, 2010, pp. 1607–1612.

[22] J. Garcia and F. Fernández, "Safe exploration of state and action spaces in reinforcement learning," *Journal of Artificial Intelligence Research*, vol. 45, pp. 515–564, 2012.

[23] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.

[24] R. M. Kretchmar and C. W. Anderson, "Comparison of cmacs and radial basis functions for local function approximators in reinforcement learning," in *Neural Networks, 1997., International Conference on*, vol. 2. IEEE, 1997, pp. 834–837.

[25] J. Peters and S. Schaal, "Learning to control in operational space," *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 197–212, 2008.