Door Opening by joining Reinforcement Learning and Intelligent Control *

Bojan Nemec, Leon Žlajpah, and Aleš Ude
Humanoid and Cognitive Robotics Lab
Jozef Stefan Institute, Ljubljana, Slovenia
bojan.nemec@ijs.si, leon.zlajpah@ijs.si, ales.ude@ijs.si

***Abstract*— In this paper we address a problem of how to open the doors with an articulated robot. We propose a novel algorithm, that combines widely used reinforcement learning approach with intelligent control algorithms. In order to speed up learning, we formed more structured search, which exploits physical constraints of the problem to be solved. The underlying controller, which acts as a policy search agent, generates movements along the admissible directions defined by physical constraints of the task. This way we can efficiently solve many practical problems such as door opening without almost any previous knowledge of the environment. The approach was verified in simulation as well as with real robot experiment.**

***Index Terms*— Reinforcement Learning, Intelligent Control, Autonomous Exploration**

I. Introduction

One of the most common operation of future generation of service robot is the door opening. Doors are used everywhere in human populated environments - to separate rooms, in wardrobes and cabinets, home appliances such as refrigerator, dishwasher. There are a lot of different parameters that affect the way doors can be opened: doors can be left or right handed, they can be opened by pushing or pulling, some doors in cabinets open vertically, doors can be sliding, etc. Many of previous work relied on detailed geometrical models of doors, where the required policy can be computed analytically [1]. Door opening can be also solved by intelligent control algorithms, which exploit natural constraints of the interactive mechanism to generate the corresponding movement [2], [3], [4]. Yet another approaches rely on learning and combining of motor primitives [5], [6], [7]. Despite of all previous work, door opening still remains a challenging task if the robot has to act autonomously. There will always be cases, for which solutions can not provided in advance. Moreover, to open the door it is usually necessary to unlatch them by moving the handle appropriately or release the latch. Such compound operations can not be solved using only the control approaches. Therefore, it is essential that a robot is capable of solving this task for any combination of doors types, handles and latches by autonomous exploration and learning.

*The research leading to these results has received partial funding from Horizon 2020 EU project AUTOWARE under grant agreement n^o 723909.

Learning of motor primitives has been heavily investigated in the past decades [8]. Currently, the majority of approaches rely on human demonstration and subsequent adaptation of the acquired policy to the actual robot dynamics while fulfilling given task and environment constraints. Reinforcement learning (RL) algorithms, which incrementally improve the initial policy, are used for model free autonomous adaptation. However, the main problem is a huge search space affected not only by the high degrees of freedom of modern humanoid and service robots, but also by the underlying parametric policy representation. In order to encode a typical policy with the duration of few seconds, it is required to learn 30 to 100 parameters for each joint, which makes the problem almost untraceable. Modern stochastic reinforcement learning algorithms [9] can deal also with such large scale problems. However, the learning is still slow. In order to overcome this problem, many approaches were proposed with the general aim to diminish the search space by diminishing the number of learning parameters [10], [11], [12]. The drawback of these approaches is that they are not general and that they require to manually design algorithms tailored for specific cases. Adaptation can be successfully used also by intelligent control algorithms such as ILCs [13], [14]. However, the latter approach is limited to some specific problems and requires to carefully tune adaptation parameters. In many cases it can adapt the policy only to some extent. Further improvements are possible with standard RL approaches applying random search algorithms [15].

This work proposes another approach, which aims to improve learning and adaptation speed by applying more focused search applied to RL. We propose that the search of the policy parameters is governed by intelligent control algorithms. Whenever this algorithms are not applicable or do not converge any more (or even diverge), we switch back to the classical random search. This paradigm was first used in bi-manual glass wiping policy adaptation, where adaptive ILC algorithm was applied for policy search [16]. In this work, we combine the RL and control policies to enhance the autonomous learning of door opening policies. We focus on learning of door opening skill only; we assume that we know the position of the handle/latch in advance and we also know how to grasp it. Everything beyond this, e.g how to manipulate the handle/bolt and how to manipulate the door, was solved by learning.

The paper consists of five sections. In Section II we review the existing control policies for door opening that inspired our approach and introduce an extension of force based control policy for orientations. Section III starts with presentation of the policy parametrization method used in our approach. Next, we introduce RL with control based policy parameter search algorithm. The proposed approach was verified in simulation, where we demonstrated the applicability of the proposed approach to diverse cases, as well as in real robot experiment. Results and implementation details are given in section of the experimental evaluation on two different platforms are given in Section IV, followed by final remarks and conclusion in Section V.

II. Control based approach for door opening

Many real world tasks, e.g. opening the doors, pushing the drawers or turning a crank, exhibit the workspace constraints which considerably limit the freedom-of-motion. The motion allowed by such constraints is actually the motion necessary to do the task. The challenge is to find this feasible path which accomplishes the task while considering all workspace constraints. For example, when opening a door with a robot, the procedure could be as follows. As soon as the robot grasps the handle, the kinematic chain is closed and the motion of the robot arm becomes very constraint. Next, to unlock the doors, turning the handle is necessary. After unlocking the door, the robot has to push or pull the doors to open them. During the motion, the robot has also to avoid any collisions or ill configurations. Therefore, it is preferred that the robots exhibits enough dexterity for such tasks.

Knowing the base position of the robot relative to the door, it is possible to calculate all the necessary paths and trajectories to turn the handle and to open the door. There have been many methods and strategies proposed to solve this path planning problem [17], [18], [19]. Common to all of them is that they are exploring the joint and/or task space to find trajectories satisfying the constraints and completing the task.

As an alternative to the above mentioned path planning strategies and methods, we want to present a concept how to execute tasks, where the path is defined by the task constraints. For that, we are using the compliancy. By observing how humans complete such tasks, we see that they make their body compliant in relevant directions. Then, by applying some forces or torques in the directions, which are not constraint by the object or environment, they move toward the position, or into the configuration which is necessary to complete the task. As the motion is "guided" by the constraints, no calculations of paths or trajectories necessary to accomplish the task are needed.

As only the concept of path following is the issue, we are assuming that the robot is equipped with some sensory system and a control system which allows the robot to move into the configuration where the kinematic chain can be closed, i.e. the robot can grasp the object. After the object is grasped, task constraints restrict the motion of the robot. Any attempt to move the robot in restricted directions results in high internal forces. We want to make use of these internal forces to reconfigure the robot into the configuration where internal forces are minimized. To keep the internal forces during the reconfiguration low, the robot should be as compliant as possible.

For a serial link manipulator consisting of n links with rotational or prismatic joints the configuration of the manipulator can be represented by the n-dimensional vector $\boldsymbol{q}$ of joint positions. As we want to define the compliance of the whole robot, we have used a cascade controller with the inner-loop controller in the form

$$\boldsymbol{\tau}_u = \mathbf{K}(\boldsymbol{q}_d - \boldsymbol{q}) + \mathbf{D}(\mathbf{d}) + \boldsymbol{\tau}_c + \boldsymbol{f}_{\text{dyn}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}), \qquad (1)$$

where $\boldsymbol{\tau}_u \in \mathbb{R}_n$ are commanded torques, $\mathbf{K}$ is a diagonal matrix defining the joint stiffness, $\mathbf{D}(\mathbf{d})$ is defining the joint space damping, and $\boldsymbol{f}_{\text{dynamics}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ is representing a known dynamic model of the robot. The term $\boldsymbol{\tau}_c$ represents superposed joint torques. To achieve a good behavior, very low joint stiffness $\mathbf{K}$ has been used and the desired joint position $\boldsymbol{q}_d$ have been equal to the actual joint positions $\boldsymbol{q}$,

$$\boldsymbol{q}_d = \boldsymbol{q}\,. \qquad (2)$$

Such a control allows to freely move the robot while applying external forces on the body of the robot. Note that with such a control the robot stands still in the current position when no external forces are applied to the robot.

The aim of the outer control loop is to generate the motion necessary to perform the task. As already mentioned, when the robot is in contact with the environment, it can move only in the direction which is not restricted by the environmental constraints. Assume for a moment that the robot is tightly coupled with the environment and that the robot is already moving. For above mentioned classes of tasks this motion is exactly the one needed to finish the task - any motion not contributing to the task competition is restricted by the environment. So, the goal of the outer control loop is to preserve the existing motion until the task is finished.

The idea is to apply a virtual force $\boldsymbol{F}_o$ to the robot end-effector which will push the robot in the moving direction until the task is completed. The original approach, proposed be Niemeier and Slotine [2], consist of two blocks; a) a velocity estimation, which robustly estimates the direction of the motion , and b) a controller, which applies the desired force and shape it in order to maintain the desired/admissible velocities. In an ideal case, the applied force is calculated as [2]

$$\boldsymbol{F} = \mathbf{K}_p \dot{\boldsymbol{p}}(v_d - \|\dot{\boldsymbol{p}}\|) \qquad (3)$$

where $\boldsymbol{F} \in \mathbb{R}^3$ is force vector, applied to the robot TCP, $\boldsymbol{p} \in \mathbb{R}^3$ are robot positions measured at the end-effector, $\mathbf{K}_p \in \mathbb{R}^{3\times3}$ is the diagonal positional controller gain matrix and v_d is the desired translational velocity. In the above equation the direction of the motion is determined by $\dot{\boldsymbol{p}}$, which might fail in noisy velocity estimates. In [2], authors proposed a spatial filtering to overcome this problem. A spatial filter smooths the direction of the motion estimates using a first order filter and assures, that the filtering does not affect the normalization. The resulting smoothed and normalized direction of linear motion $\boldsymbol{d}_p$ can be computed from

$$\dot{\boldsymbol{d}}_p = \lambda(1 - \boldsymbol{d}_p\boldsymbol{d}_p^T)\dot{\boldsymbol{p}}, \tag{4}$$

where λ is the filter bandwidth. A discrete time implementation of the above filter is

$$\boldsymbol{d}_p(k) = \boldsymbol{d}_p(k\text{-}1) + \lambda(1 - \boldsymbol{d}_p(k\text{-}1)\boldsymbol{d}_p^T(k\text{-}1))(\boldsymbol{p}(k) - \boldsymbol{p}(k\text{-}1)), \tag{5}$$

where k denotes the k-th time sample. In order to control the robot, forces

$$\boldsymbol{F}(k) = \mathbf{K}_p\boldsymbol{d}_p(k)(v_d(k) - \|\dot{\boldsymbol{p}}(k)\|) \tag{6}$$

are applied as command values to the robot controller. The original formulation neglects torques. In practice, it is often necessary to apply also torques, for, e.g. turning the door knobs. Therefore, it is necessary to extend Eq. 7 for torques. Straightforward extension yields

$$\boldsymbol{M} = \mathbf{K}_o\boldsymbol{\omega}(\omega_d - \|\boldsymbol{\omega}\|), \tag{7}$$

where $\boldsymbol{M} \in \mathbb{R}^3$ is torque vector, applied to the robot TCP, $\mathbf{K}_o \in \mathbb{R}^{3\times3}$ is the diagonal rotational controller gain matrix, $\boldsymbol{\omega} \in \mathbb{R}^3$ are robot rotation velocities and scalar ω_d is the desired rotation velocities. For the specification of the robot orientation, unit quaternions are usually used, as they provide convenient singularity free mathematical notation. We will denote them as $\mathcal{Q} = \{\eta, \boldsymbol{\epsilon}\} \in \mathbb{R}^4$, where η and $\boldsymbol{\epsilon}$ are the corresponding scalar and vector part of the quaternion, respectively. Angular velocities can be calculated from two subsequent quaternions as

$$\boldsymbol{\omega}(k) = 2\log(\mathcal{Q}(k) * \bar{\mathcal{Q}}(k-1)), \tag{8}$$

where $*$ denotes the quaternion multiplication and the quaternion logarithm is calculated as

$$\log(\mathcal{Q}) = \log(\eta, \boldsymbol{\epsilon}) = \begin{cases} \arccos(\eta)\dfrac{\boldsymbol{\epsilon}}{\|\boldsymbol{\epsilon}\|}, \ \eta \neq 0 \\ [0,0,0]^{\mathrm{T}}, \ \text{otherwise} \end{cases}, \tag{9}$$

The smoothed direction of angular motion $\boldsymbol{d}_o$ can be calculated as

$$\boldsymbol{d}_o(k) = \boldsymbol{d}_o(k-1) + T\lambda(1 - \boldsymbol{d}_o(k-1)\boldsymbol{d}_o^T(k-1))\boldsymbol{\omega}(k), \tag{10}$$

and used to calculate the commanded torques

$$\boldsymbol{M}(k) = \mathbf{K}\boldsymbol{d}_o(k)(\omega_d - \|\boldsymbol{\omega}\|). \tag{11}$$

T denotes the sampling frequency.

As we are not controlling directly the robot pose, some of the joints may move into the limits or the robot may move into an ill configuration. To avoid these, we have used the available redundant DOFs to optimize the pose of the robot, i.e. the robot should move to a predefined pose whenever possible. For that, we have used the self-motion of the robot.

To calculate the motor torques in each sampling interval k we have used the control (1) with superposed joint torques $\boldsymbol{\tau}_c$

$$\boldsymbol{\tau}_c(k) = \mathbf{J}^T \begin{bmatrix} \boldsymbol{F}(k) \\ \boldsymbol{M}(k) \end{bmatrix} + \mathbf{N}^T(k)\,\boldsymbol{\tau}_n(k) \tag{12}$$

where $\mathbf{J} \in \mathbb{R}^{6\times n}$ is the robot Jacobian, $\mathbf{N} \in \mathbb{R}^{n\times n}$ is a matrix representing the projection into the null space of $\mathbf{J}$, and $\boldsymbol{\tau}_n \in \mathbb{R}^n$ are the null-space torques used to maintain the desired pose.

Flowing this policy, a robot can perform many tasks from everyday life such as closing and opening doors, drawers, sliding doors, etc.. without any previous knowledge of the objects, which are involved in interaction. It applies forces solely in the direction of the movement, whereas it is intrinsically compliant in orthogonal directions. This property effectively minimizes internal wrenches, which can arise in position based policies due to the kinematics/dynamics model errors. The only think the robot has to know in advance is where to apply such forces, e.g. where is the door handle and how to grasp it. This problem was extensively studied in [5], where computer vision was applied to locate door handle.

However, this approach alone can not generate complex policies, which are composed of various primitives, such as pushing the door handle and opening the door. Another problem with this approach are backlashes, which are manifested as additional degrees of freedom. In order to overcome above mentioned problems, we will apply reinforcement learning.

III. Learning of door opening

Reinforcement Learning (RL) is widely used in machine learning for solving problems where exact models are not available. Traditional RL approaches based on discrete states/actions need to encode each possible combination of the robot state and action as a discrete Markov decision process (MDP), which results in huge action-state space [20]. For this reason, classical RL algorithms like Q-learning and SARSA are rarely used in robotics. In the door opening problem, the robot can not visit every possible state, since the robot motion is restricted by the physical constraints of the door. Therefore, the course of dimensionality is not as crucial as for the robot motion in free space. However, states and actions are still discrete, which can result in non smooth policies. For this reason we will apply probabilistic policy improvement RL algorithms PI^2 [21], which can scale to complex learning systems and minimizes the number of tuning parameters.

PI^2 learns such policy parameters $\mathbf{W}$, which minimize the expected cost J of each learning cycle (refereed also as roll out)

$$J(\mathbf{W}) = c_t + \int_{t_o}^{T} (c_i(t) + \frac{1}{2}\boldsymbol{u}^T\mathbf{R}\boldsymbol{u})dt, \tag{13}$$

where c_t is the terminal cost received after the accomplished roll out, $c_i(t)$ is cost at each sampling interval (refereed also as intermediate cost), $\boldsymbol{u}$ is the control signal and $\mathbf{R}$ is the weighting matrix, which minimizes control cost [10]. Policy search is usually obtained by applying zero mean Gaussian noise $\mathcal{N}(0, \sigma^2)$ to the policy parameters $\mathbf{W}$, where noise variance σ^2 remains the only tunning parameter. After each roll out L, new optimized policy parameters $\mathbf{W}^*$ are calculated using all previous policy parameters, terminal and intermediate costs,

$$\mathbf{W}^* = \Upsilon(\mathbf{W}_l, \boldsymbol{c}_{i,l}, c_{t,l}, l = 1, \dots L). \tag{14}$$

Υ denotes the reinforcement learning algorithm PI^2. Detailed description of the of PI^2 is out of the scope of this paper. A good step by step instructions how to implement PI^2 can be found in [10]. In order to speed up learning and reject the unsuccessful attempts, the input data to (14) are reordered after each learning cycle using the importance sampling [9].

A. Policy representation

Modern reinforcement learning algorithms, that can generate continuous action/states policies, require appropriate policy representation. A choice of policy representation is not trivial, as it must fulfill a number of requirements, such as the smoothness of resulting policy, the scalability to high dimensional problems, the compactness in encoding a policy, the regularization ability, the invariance to different topological representations, the adaptability of the policy, etc. [22]. Among the most popular policiy representations suitable for RL are Gaussian Mixture Models (GMM) [23], Dynamic Motion Primitives (DMP) [24], Probabilistic Motion Primitives (PMP) [25] and Radial Basis Functions (RBF) [26]. Latter representation use a series of kernel functions, usually formed as Gaussian functions, and weights to encode an arbitrary time dependent function. By introducing a simple canonical system, the time dependency of the policy is removed, which enables to generalize to similar policies with different duration/velocity profiles.

Forces and torques, which are to be learned, are encoded as a weighted sum of m Gaussian kernels for each dimension

$$\begin{aligned} \boldsymbol{F}(s) &= \frac{\sum_{i=1}^{m} \boldsymbol{w}_{f,j,i}\Psi_i(s)}{\sum_{i=1}^{m} \Psi_i(s)} s, & (15) \\ \boldsymbol{M}(s) &= \frac{\sum_{i=1}^{m} \boldsymbol{w}_{m,j,i}\Psi_i(s)}{\sum_{i=1}^{m} \Psi_i(s)} s, & (16) \\ \Psi_i(s) &= \exp\left(-h_i\left(s - c_i\right)^2\right), & (17) \end{aligned}$$

where the free parameters $\boldsymbol{w}_{f,j,i}$ and $\boldsymbol{w}_{m,j,i}$ determine the shape of force and torque trajectories. c_i are the centers of RBFs, evenly distributed along the trajectory, with h_i their widths. A canonical systems

$$\tau\dot{s} = -\alpha_s s. \tag{18}$$

sets the phase variable s, which is initially set to 1 and decays to 0. α_s is the appropriately set decay factor.

To calculate $w_{f,j,i}$ from j-th component of the force vector $\boldsymbol{F}$, $\boldsymbol{F}_j(k), 1 \le k \le T$, we need to solve

$$\mathbf{B}\boldsymbol{w}_{f,j} = \boldsymbol{u}, \tag{19}$$

with

$$\boldsymbol{w}_{f,j} = \begin{bmatrix} w_{f,1,j} \\ \vdots \\ w_{f,M,j} \end{bmatrix}, \boldsymbol{u} = \begin{bmatrix} \boldsymbol{F}_j(1) \\ \vdots \\ \boldsymbol{F}_j(T) \end{bmatrix}, \tag{20}$$

and the system matrix $\mathbf{B} \in \mathbb{R}^{T \times M}$ defined as

$$\mathbf{B} = \begin{bmatrix} \frac{\psi_1(s(1))}{\sum_{j=1}^{M} \psi_j(s(1))} & \dots & \frac{\psi_M(s(1))}{\sum_{j=1}^{M} \psi_j(s(1))} \\ \vdots & \vdots & \vdots \\ \frac{\psi_1(s(T))}{\sum_{j=1}^{M} \psi_j(s(T))} & \dots & \frac{\psi_M(T\Delta t)}{\sum_{j=1}^{M} \psi_j(s(T))} \end{bmatrix}. \tag{21}$$

The weights $\boldsymbol{w}_{m,j}$ are calculated in the same way.

B. Policy Learning

In Section II we introduced the controller that is able to autonomously perform simple actions such as door opening and pushing the door handle, if the initial direction of the motion is known in advance. As we aim to obtain autonomous robot behavior, that solves also situations which have never been encountered before, this information should be obtained by the autonomous exploration and learning. For this, we propose a novel approach, where the control algorithm acts as a parameter search process within the RL method. In this way, we can effectively solve also a jamming, collisions and other unexpected situations, that might arise during the learning.

The general strategy is again very simple. In each learning cycle the robot checks first if the forces and torques applied so far result in a motion. In this case the robot continues motion in the estimated direction. Otherwise, the robot applies forces and torques learned so far, perturbed by some random forces and torques, generated as uniformly distribute random number with specified variance. Robot collects intermediate costs, applied forces and torques. The roll out is ended after the reaching the goal (which is the door opening in our case) or after the maximal allowed time for each episode. At this time, the robot collects also the terminal cost and calculates new estimate of forces and torques for the next roll out using PI^2. This procedure is repeated until the robot learns the desired policy.

Intermediate and terminal cost were assigned as

$$c_i(k) = \begin{cases} 10|v_d - \|\dot{\boldsymbol{p}}(k)\||, & \text{doors opening} \\ 100\|\dot{\boldsymbol{p}}(k)\|, & \text{otherwise} \end{cases}$$
$$c_t = \begin{cases} t_o, & \varphi \geq \varphi_d \\ 10(\varphi_d - \varphi), & \text{otherwise} \end{cases}$$

where t_o is time needed to open the door and φ, φ_d denote the actual and the desired door opening angle, respectively.

In practice, the procedure is a little bit more complicated. For successful learning, all signals have to be of equal length and spatially aligned. Namely, we can not simply compare two action, that happen at the same time. Rather, we have to compare actions, that happen at the same place. To accomplish spatial alignment, we use the following algorithm.

First, we compute translational and rotation distances $\nu(k)$ in each sampling interval, $\nu(k) = \sum_{i=0}^{i=k} |v(i)| + \gamma|\omega(i)|$ and the corresponding phase signal $s(k)$ (18) for each roll-out and express the phase $s(\nu)$ as a function of distance. Scalar γ is used to weight the angular velocities, as they are summed with translational velocities. We save these functions in the importance sampler of length L_i [9]. During the next roll out, we compute $\nu(k)$ and phases from functions saved in the importance sampler, $s(k) = s_j(\nu(k)), j = 1 \dots L_i$. The auxiliary phase is then computed by reweighing the phase estimates according to the terminal cost $c_{t,j}$,

$$s_a(k) = \frac{\sum_{j=1}^{j=L_i} \frac{s_j(\nu(k))}{c_{t,j}}}{\sum_{j=1}^{j=L_i} \frac{1}{c_{t,j}}} \qquad (22)$$

With this auxiliary phase we estimate command force and torque signals using Eq. (15,16).

Next, we have to provide that all signals involved in the learning (forces, torques and intermediate costs) have equal lengths. This is accomplished by copying the final value of signals of all prematurely finished roll outs (in our case this is when the robot opens the door) until the end. Next, forces and torques obtained this way are encoded as RBF (19) and are passed to the RL algorithm, which computes next estimates of weights of RBF. This estimates are used in the next roll out. Note also, that the random search variance σ^2 was scaled by the factor $\alpha^l, \alpha < 1$. This choice assures smoother policy search in subsequent cycles. The entire learning procedure is summarized in Algorithm 1.

Algorithm 1: door opening learning algorithm

```
Input: Initial position of the door handle
       initialize control gains $\mathbf{K}_p$, $\mathbf{K}_o$
       initialize $\boldsymbol{F}(k) = 0,\ \boldsymbol{M}(k) = 0, k = 1, \dots T$
Output: Learned policy ($\boldsymbol{F}(k)$, $\boldsymbol{M}(k)$)
1  for $l = 1, \dots,$ max learning cycles do
2      for $k = 1, \dots,$ $T$ do
3          if robot is moving then
4              calculate directions $\boldsymbol{d}_p$ and $\boldsymbol{d}_o$ (5,10)
5              calculate $\boldsymbol{F}(k)$ and $\boldsymbol{M}(k)$ (6,11)
           else
6              calculate auxilary phase variable $s_a$ (22)
7              calculate $\boldsymbol{F}(s)$ and $\boldsymbol{M}(s)$ learned so far
                (15,16)
8              add noise $\boldsymbol{F}(k) = \boldsymbol{F}(s) + \mathcal{N}(0,\ \sigma_p^2)$
9              add noise $\boldsymbol{M}(k) = \boldsymbol{M}(s) + \mathcal{N}(0,\ \sigma_o^2)$
10         calculate and apply joint torques to the motors
            (12)
11         collect intermediate cost $c_i(k)$
12     collect terminal cost $c_t$
13     calculate $\boldsymbol{w}_{f,l}$ and $\boldsymbol{w}_{m,l}$ from $\boldsymbol{F}(k), \boldsymbol{M}(k)$ (19)
14     save data in the importance sampler
15     estimate new optimized weights $\boldsymbol{w}_{f,l+1}$ and $\boldsymbol{w}_{m,l+1}$
        using PI$^2$ (14)
16     update search noise variance $\sigma_p^2 = \alpha\,\sigma_p^2;\ \ \sigma_o^2 = \alpha\,\sigma_o^2;$
```

IV. Results

The proposed algorithm was verified both in simulated environment and using a real robot. For this purpose, we applied state of the art MuJoCo HAPTIX simulation environment [27], which was used to simulate KUKA LWR 4 robot interacting with the environment. Control and learning algorithms were implemented in MATLAB. The control sampling rate of the outer control loop was 0.05 sec while the inner loop sampling rate was 0.002 sec.

The learning algorithm (Eqs. (6) and (11)) was implemented at slower sampling rate of 0.1 sec. The control gains $\mathbf{K}_p$ and $\mathbf{K}_o$ were chosen as 50 $\mathbf{I}$, where $\mathbf{I}$ is the identity matrix. The initial value of the σ^2 was set to 50 (N) and α was 0.97. In order to evaluate the success of the learning, we performed greedy cycle after each 5-th roll-out, where the robot was controlled only by learned forces.

In simulation, we evaluated how good is the performance of the proposed algorithm for the following cases: 1) Right hand door opening with normal handle (see Fig. 11); 2) Opening of cabinet sliding door, where the doors are unlatched by pushing latch up (see Fig. 12); 3) Left hand cabinet door opening with a handle, which has to be lifted to unlatch (see Fig. 13); 4) Opening a cabinet drawer, which has to be lifted to unlatch (see Fig. 14); 5) Opening a horizontal cabinet door (see Fig. 15). In all case we applied identical learning algorithm. The only difference was in assigning the terminal cost for drawer and sliding door opening, where we specified the desired distance instead of the desired angle. In all of this cases, the task was successfully accomplished by applying only forces. Note that also the tool orientation was changing although we didn't apply any torques. This property

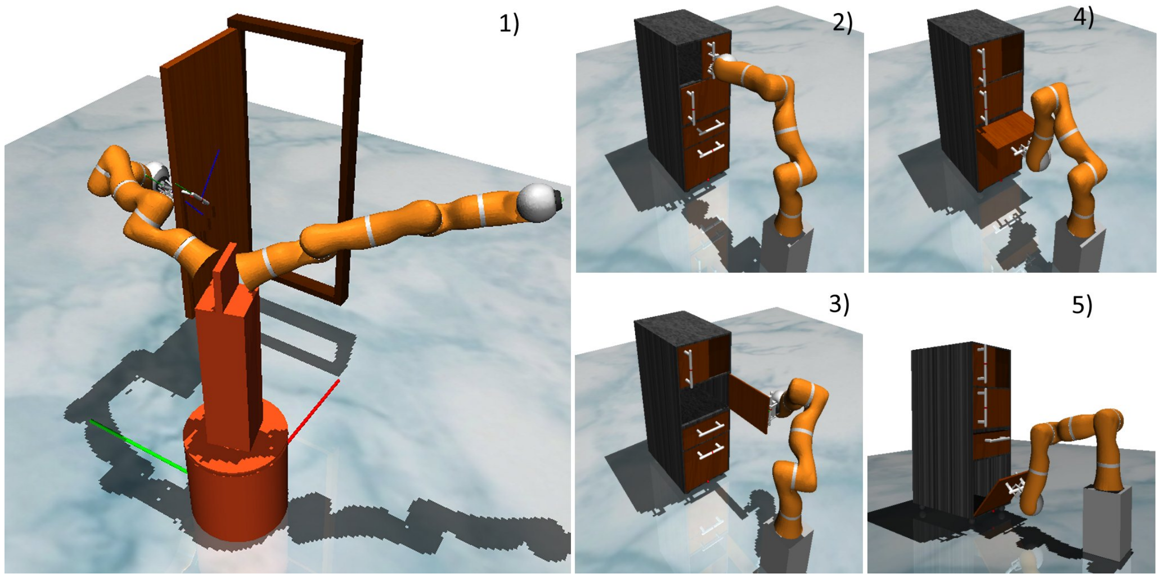

Fig. 1. Graphical output of MuJoCo simulation of: 1) Door opening; 2) Sliding cabinet door; 4) Drawer opening; 5) Vertical cabinet door;

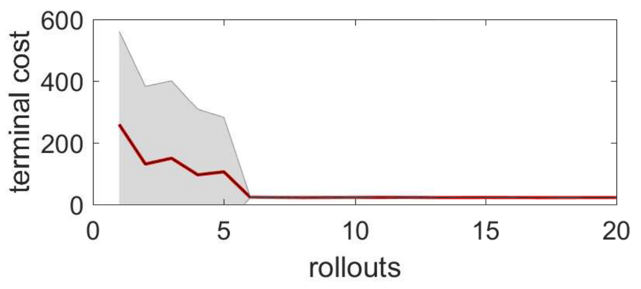

Fig. 2. Mean value (red line) and standard deviation (shaded region) of terminal cost during learning of door opening. Learning resulted in successful door closing at most after 6 roll outs in all cases.

was provided by the intrinsically compliant controller, which minimizes the interaction forces.

We performed 20 learning experiments consisting of 20 roll-outs. In simulation, we obtained 100% success for all tasks. In average, robot took 6 roll-outs to learn the appropriate policy except for the drawer opening, which took in average 10 roll outs. Namely, in this case the robot had to learn the force profile that is composed of vertical (z) force followed by the negative pulling force (x). Only after discovering this pattern, the controller can help to continue the motion in the unconstrained direction. Learning costs and standard deviation of door opening are given in Fig. 2.

Next, we evaluated the proposed learning procedure also experimentally. For this purpose we used bi manual robot composed of of two KUKA LWR4 robot arms, equipped with Barret hand and mounted on the torso with one rotational d.o.f. The task of the robot was to open right hand door with left robot arm. The setup is shown in Fig. 3. The robot pose required to grasp the door handle was obtained with kinesthetic guiding in zero gravity mode. We performed 20 roll out of learning with the same parameter settings as used in simulated environment. In this experiment, we didn't learn torques $\boldsymbol{M}$, since this was not necessary for this task. In average, the robot learned the required policy in 9 roll outs. Learned force policy is shown in Fig. 4.

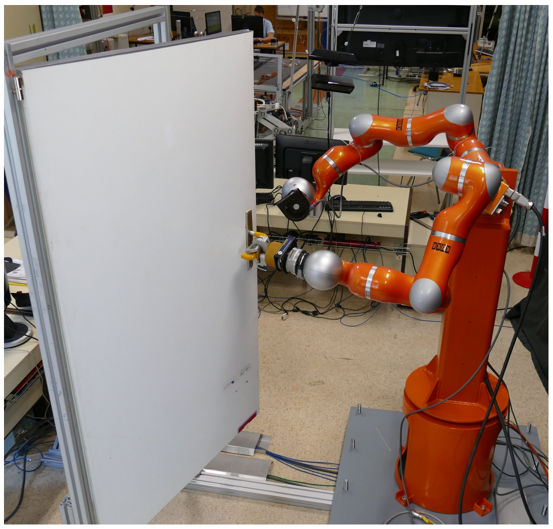

Fig. 3. Experimental setup for door opening.

V. Conclusions

In the paper we proposed a novel algorithm for learning of tasks, where the robot motion is constrained by the environment interaction (e.g. a mechanism). It joins the RL

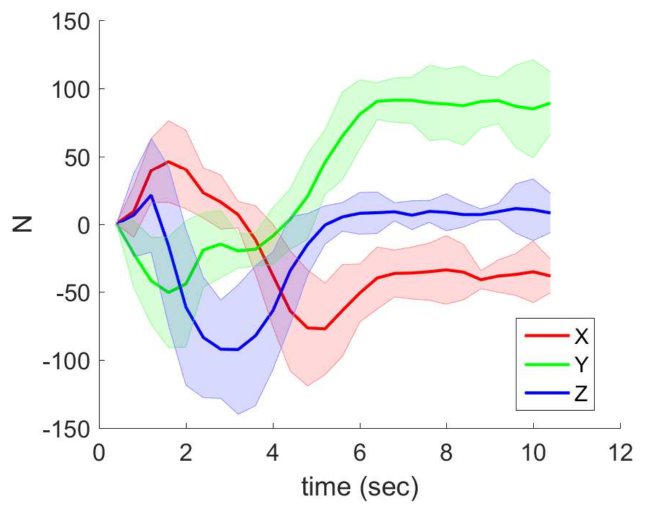

Fig. 4. Mean values and standard deviations (shaded regions) of learned force profiles

and intelligent control. The approach is inspired by previous work, where a controller applies forces to the unconstrained degrees of freedom of the robot. The task of the learning part of the algorithm is to determine the unconstrained degrees of freedom, that result in the given task accomplishment. The underlying controller is acting as exploration in the action space [8]. With this algorithm, the robot efficiently learns many tasks such as door and drawer opening. It can learn also complex motions, where it is necessary to manipulate the door handle or latch, without any presumption how to do this. In this work we assumed that the robot already knows how to grasp a door handle or latch. Algorithm was verified in simulated environment and with the real bimanual robot, composed of the two KUKA LWR 4 robot arms mounted on the torso with 1 d.o.f. Both simulations and real experiment demonstrated 100% success of learning. In future we will experimentally evaluate the proposed method on more complex cases and with different reward functions, which will minimize applied forces and assure smoother policies.

References

[1] K. Nagatani and S. Yuta, "An experiment on opening-door-behavior by an autonomous mobile robot with a manipulator," *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems* (IROS), pp. 45–50, 1995.

[2] G. Niemeyer and J.-j. E. Slotine, "A simple strategy for opening an unknown door," *Proceedings of the 1997 IEEE International Conference on Control Applications*, pp. 1448–1453, 1997.

[3] Y. Karayiannidis, C. Smith, P. Ögren, and D. Kragic, "Adaptive force/velocity control for opening unknown doors," *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 753–758, 2012.

[4] M. Levihn and M. Stilman, "Using environment objects as tools: Unconventional door opening," *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 2502–2508, 2014.

[5] E. Klingbeil, A. Saxena, and a. Y. Ng, "Learning to open new doors," *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* , pp. 2751–2757, 2010.

[6] F. Endres, J. Trinkle, and W. Burgard, "Learning the dynamics of doors for robotic manipulation," *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 3543–3549, 2013.

[7] S. Otte, J. Kulick, M. Toussaint, and O. Brock, "Entropy-based strategies for physical exploration of the environment's degrees of freedom," *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 615–622, 2014.

[8] M. P. Deisenroth, "A Survey on Policy Search for Robotics," *Foundations and Trends in Robotics*, vol. 2, no. 1, pp. 1–142, 2011.

[9] J. Kober, J. a. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[10] F. Stulp and O. Sigaud, "Path Integral Policy Improvement with Covariance Matrix Adaptation," *arXiv preprint arXiv:1206.4621*, 2012.

[11] J. Kober, A. Wilhelm, E. Oztop, and J. Peters, "Reinforcement learning to adjust parametrized motor primitives to new situations," *Autonomous Robots*, vol. 33, no. 4, pp. 361–379, 2012.

[12] B. Nemec, R. Vuga, and A. Ude, "Efficient sensorimotor learning from multiple demonstrations," *Advanced Robotics*, vol. 27, no. 13, pp. 1023–1031, 2013.

[13] F. J. Abu-Dakka, B. Nemec, J. A. Jørgensen, T. R. Savarimuthu, N. Krüger, and A. Ude, "Adaptation of manipulation skills in physical contact with the environment to reference force profiles," *Autonomous Robots*, vol. 39, no. 2, pp. 199–217, 2015.

[14] B. Nemec, T. Petrič, and A. Ude, "Force adaptation with recursive regression Iterative Learning Controller," *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 2835–2841, 2015.

[15] R. Vuga, B. Nemec, and A. Ude, "Enhanced Policy Adaptation Through Directed Explorative Learning," *International Journal of Humanoid Robotics*, vol. 12, no. 03, 2015.

[16] B. Nemec, M. Simonic, N. Likar, and A. Ude, "Enhancing the performance of adaptive iterative learning control with reinforcement learning," in *Submitted to 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[17] Z. Yao and K. Gupta, "Path planning with general end-effector constraints," *Robotics and Autonomous Systems*, vol. 55, no. 4, pp. 316–327, 2007.

[18] S. LaValle, J. Yakey, and L. Kavraki, "A probabilistic roadmap approach for systems with closed kinematic\nchains," *Proceedings 1999 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1671–1676, 1999.

[19] M. Stilman, "Global manipulation planning in robot joint space with task constraints," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 576–584, 2010.

[20] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in Cognitive Sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[21] E. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," *The Journal of Machine Learning Research*, vol. 11, pp. 3137–3181, 2010.

[22] P. Kormushev, S. Calinon, D. G. Caldwell, and B. Ugurlu, "Challenges for the policy representation when applying reinforcement learning in robotics," *Proceedings of the International Joint Conference on Neural Networks*, pp. 10–15, 2012.

[23] S. Calinon, "Robot Learning with Task-Parameterized Generative Models," *Proc. Intl Symp. on Robotics Research*, pp. 1–16, 2015.

[24] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors." *Neural computation*, vol. 25, no. 2, pp. 328–73, 2013.

[25] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Probabilistic Movement Primitives," *Neural Information Processing Systems*, pp. 1–9, 2013.

[26] D. Buhmann, *Radial Basis Functions: Theory and Implementations*. Cambridge Univ. Press, 2003.

[27] E. Todorov. Mujoco advanced physics simulation. [Online]. Available: http://www.mujoco.org/