Compliant control for efficient robot sensory-motor learning

Bojan Nemec, Viktor Stefanovski, Leon Žlajpah, and Aleš Ude

***Abstract*— The paper proposes a new strategy for learning of tasks where the robot interacts with the environment. The learning relies on the compliant control, which assures compliant behavior in the direction where the motion is constrained by the environment. The proposed approach learns first the directions, where the robot should be compliant. Then, it applies forces in the orthogonal directions and observes the resulting actions. This simple strategy is very efficient in the learning of tasks, where the motion is constrained by the environment. The proposed algorithm was verified both in the simulation and using real robot in challenging tasks such as learning of door or drawer opening.**

I. INTRODUCTION

For an autonomous robot acting in unstructured environment or when interacting with humans is necessary that the robots exhibit compliant behavior. For truly autonomous robot it is necessary also, that the robot is capable of autonomously finding new policies and to adapt previously known policies to the environment changes. Reinforcement learning is a widely used technique for such purposes. In robotics, especially for sensory-motor learning, model free direct statistical policy search algorithms like Pi2 and PoWER turned to be the most efficient one. They can scale to high dimensional learning problems usually encountered in robotics. Note that a humanoid robot can have 30 to 50 d.o.f. and that a typical policy is described with 30 to 100 parameters for each degree of freedom, which results in a huge search space. This is one of the reasons why robot learning approach is still inefficient compared to the capabilities of humans and animals. Techniques like learning in latent spaces, learning of meta parameters, which more efficiently describe the learning problem, or covariance matrix adaptation and statistical generalization techniques can dramatically reduce the search space in RL. However, they all require at least a partial knowledge of a model of the process, which can be either given apriori in an explicit form or inherited from previous experiments.

In the robotics community, tasks that involve interaction with environment are considered as extremely hard to learn due to the unknown and possibly changing environment. In this paper, we show that interacting with the environment can be advantageous in terms of the learning speed. In our opinion, learning of constraint tasks can be easier compared to the learning of tasks, where a robot can move freely in space. The reason is that the environment constrains the admissible robot movements. Consequently, the number of parameters, which have to be learned, can be reduced. Of course, it is necessary to ensure/allow the natural motion of the mechanism along the constraints imposed by the environment. A suitable framework for implementing such strategy is provided by the compliant robot control.

In this article, we present the application of the proposed methodology by two examples. The first example is the autonomous learning of opening a door with a robot. Beside learning the basic motion to open a door, the robot has to learn also the correct sequence of movements needed unlock the door, i.e. how to manipulate handles, bolts, etc. Second case is the drawer opening operation, where it is necessary to lift the drawer before open it. As before, the robot has to learn a correct sequence of movements to fulfill the task. In all this cases, the robot movements were constrained in all spatial directions except in two of them. Using the proposed approach the robot was able to learn the required policy in just a few learning cycles, despite of the fact, that virtually no previous information about the nature of the problem was given to the robot.

II. CONTROLLER BASED POLICY SEARCH

In this chapter we introduce a general strategy for the policy search. The essence of the proposed strategy is that instead of a random search in the parametrized space of policies, we apply a direct search in the action space [1]. Whenever a random action results in a movement, we try to continue this motion using a compliant controller, which maintains the speed in the current direction and allows motion along the constraints.

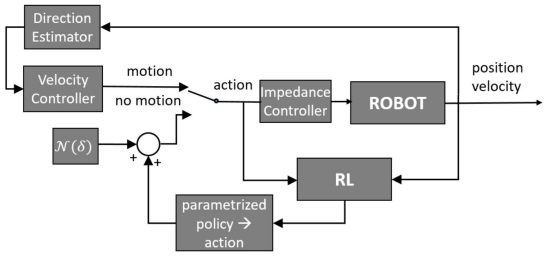

Fig. 1. General scheme for learning of policies which interact with the environment.

In general we do not know in advance in which directions

Author are with Humanoid and Cognitive Robotics Lab, Jožef Stefan Institute, Ljubljana, Slovenia bojan.nemec@ijs.si, viktor.stefanovski@ijs.si, leon.zlajpah@ijs.si, ales.ude@ijs.si

are the natural constraints of the system. To find a feasible motion direction, we apply a random force in a random direction. If this force results in a movement, we use compliant control to continue the motion initiated by the random force. The control is realized in the tool coordinate frame and the control parameters make the robot compliant in all directions orthogonal to the direction of the motion. These directions can estimated by applying Frenet-Serret frames [2] to the resulting motion trajectory. The resulting motion is supervised by the RL, which, according to the given reward or cost, decides whether this movement has contributed to the problem solving or not. We assign intermediate and terminal reward/cost. The intermediate reward/cost is assigned whenever the robot finds an admissible motion. This way, we motivate RL algorithms to find policies that result in a motion. Terminal reward/cost is assigned when the roll-out is terminated and depends on how successful was the accomplishment of the task.

Whenever the initiated robot motion stops, we assume that this is due to the task constraints and we try to find a new feasible motion by applying again a random force in a random direction. Following this strategy, the robot eventually learns how to perform the task in a form of a parametrized policy. The general scheme of the proposed learning framework is presented in Fig. 1

III. DOOR OPENING

The door opening is one of the most common operations, which humanoid robots have to be capable to perform. Doors are used everywhere in a human populated environments — to separate rooms, in wardrobes and cabinets, or home appliances such as refrigerator, dishwasher etc. Doors can be very different: left or right handed, they can be opened by pushing or pulling, the opening motion can be horizontal, vertical or sliding, etc. Of course, the policy for the door opening can be computed analytically [3]. However, this requires detailed geometrical models of doors. Another possibility is to learn the corresponding policy from demonstrations combining different motor primitives [4], [5], [6]. The problem of the door opening can be also solved by control algorithms, which exploit natural constraints of the interactive mechanism to generate the corresponding movement [7], [8], [9]. Despite of all previously proposed approaches, the door opening remains a challenging task when the robot has to act autonomously and the solution cannot be provided in advance. In many cases it is necessary to unlatch doors, e.g. by moving the handle appropriately or by releasing the latch before opening the door. Such compound operations can not be solved using only the control approaches, the operation sequence has to be predefined or learned.

A. Controller for door opening

Our approach [10] is inspired by observing how humans complete such tasks. The basic principle is to apply some forces or torques in the directions, which are not constraint by the object or the environment and to make the system compliant in the orthogonal directions. In this way, the motion is "guided" by the constraints. Whenever the induced motion would force the system to move into a constraint, the compliant controller would align the motion to be orthogonal to the constraints. Consequently, no calculations of paths or trajectories are necessary to accomplish the task. The same strategy can be used for robots, where we apply a virtual force $\boldsymbol{F}_v$ to the robot end-effector, which pushes the robot in the moving direction until the task is completed. Our approach relies on the work of Niemeier and Slotine [7]. It consist of two blocks; estimation of the direction of the motion, and a controller, which shapes the force $\boldsymbol{F}_v$ in order to maintain the desired/admissible velocities. The applied force is calculated as

$$\boldsymbol{F}_v = \mathbf{K}_p \left(v_d - \|\dot{\boldsymbol{p}}\|\right)\boldsymbol{d}_p \tag{1}$$

where $\boldsymbol{F} \in \mathbb{R}^3$ is a force vector, applied to the robot end-effector, $\boldsymbol{p} \in \mathbb{R}^3$ are the robot end-effector positions, $\mathbf{K}_p \in \mathbb{R}^{3\times3}$ is a diagonal gain matrix, and scalar v_d is the desired translational end-effector velocity. The direction of motion $\boldsymbol{d}_p$ is the tangent vector of the Frenet-Serret frame. It can be estimated from $\dot{\boldsymbol{p}}$, which might fail in noisy velocity. A better estimate is obtained using a spatial filtering [7], which smooths the noisy estimates using a first order filter and assures, that the filtering does not affect the normalization. A discrete time implementation of the spatial filter is

$$\boldsymbol{d}_p(k) = \boldsymbol{d}_p(k\text{-}1) + \lambda(1 - \boldsymbol{d}_p(k\text{-}1)\boldsymbol{d}_p^T(k\text{-}1))(\boldsymbol{p}(k) - \boldsymbol{p}(k\text{-}1)), \tag{2}$$

where λ is the filter bandwidth and k denotes the k-th time sample. In order to control the robot, forces

$$\boldsymbol{F}_v(k) = \mathbf{K}_p(v_d(k) - \|\dot{\boldsymbol{p}}(k)\|)\boldsymbol{d}_p(k) \tag{3}$$

are applied as command values to the robot controller. The original formulation [7] neglects the torques. In practice, it is often necessary to apply also torques, e.g. when turning the door knobs. Therefore, it is necessary to extend Eq. 1 for torques. Straightforward extension yields

$$\boldsymbol{M} = \mathbf{K}_o(\omega_d - \|\boldsymbol{\omega}\|)\boldsymbol{\omega}\ , \tag{4}$$

where $\boldsymbol{M} \in \mathbb{R}^3$ is a torque vector, applied to the robot end-effector, $\mathbf{K}_o \in \mathbb{R}^{3\times3}$ is the diagonal rotational controller gain matrix, $\boldsymbol{\omega} \in \mathbb{R}^3$ is a vector of robot end-effector instantaneous velocities, and scalar ω_d is the desired rotation velocity. 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

$$\boldsymbol{\omega}(k) = 2\log(\mathcal{Q}(k) * \bar{\mathcal{Q}}(k-1)), \tag{5}$$

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

$$\log(\mathcal{Q}) = \log(\eta, \boldsymbol{\epsilon}) = \begin{cases} \arccos(\eta)\dfrac{\boldsymbol{\epsilon}}{\|\boldsymbol{\epsilon}\|}, & \eta \neq 0 \\ [0,0,0]^{\mathrm{T}}, & \text{otherwise} \end{cases}, \quad (6)$$

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

$$\boldsymbol{d}_o(k) = \boldsymbol{d}_o(k-1) + dT\lambda(1 - \boldsymbol{d}_o(k-1)\boldsymbol{d}_o^T(k-1))\boldsymbol{\omega}(k), \quad (7)$$

and is used to calculate the commanded torques

$$\boldsymbol{M}(k) = \mathbf{K}_o(\omega_d(k) - \|\boldsymbol{\omega}(k)\|)\boldsymbol{d}_o(k). \quad (8)$$

dT 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 prevent such situations, we have used the available redundant DOFs to optimize the pose of the robot, i.e. the robot should preserve a predefined pose whenever possible.

To calculate the motor torques in each sampling interval k we have used the following joint impedance control law control

$$\begin{aligned} \boldsymbol{\tau}_u(k) &= \mathbf{K}(\boldsymbol{q}_d(k) - \boldsymbol{q}(k)) + \mathbf{D}(\dot{\boldsymbol{q}}(k)) + \boldsymbol{\tau}_c(k) + \\ &\quad \boldsymbol{f}_{\text{dyn}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \\ \boldsymbol{\tau}_c(k) &= \mathbf{J}^T \begin{bmatrix} \boldsymbol{F}(k) \\ \boldsymbol{M}(k) \end{bmatrix} + \mathbf{N}^T(k)\, \boldsymbol{\tau}_n(k) \end{aligned} \quad (9)$$

where $\boldsymbol{\tau}_u \in \mathbb{R}_n$ are the commanded torques, $\mathbf{K} \in \mathbb{R}^{n\times n}$ is a diagonal joint stiffness matrix, $\mathbf{D}(\mathbf{d}) \in \mathbb{R}^n$ is the joint space damping vector, $\boldsymbol{f}_{\text{dyn}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \in \mathbb{R}^n$ is the vector that compensates for the robot non-linear dynamics, $\mathbf{J} \in \mathbb{R}^{6\times n}$ is the robot Jacobian, $\mathbf{N} \in \mathbb{R}^{n\times n}$ is a matrix representing the projection into the null space of $\mathbf{J}^T$, and $\boldsymbol{\tau}_n \in \mathbb{R}^n$ are the null-space torques used to optimize the robot motion. To achieve a compliant robot behavior, we select low gains in $\mathbf{K}$ and set $\boldsymbol{q}_d = \boldsymbol{q}$, which practically means that the robot in very compliant and it dose not violates the positional constraints at any time.

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 or environment, which are involved in interaction. It applies forces solely in the direction of the movement, whereas it is intrinsically compliant in orthogonal directions. Such approach effectively minimizes internal wrenches, which can arise in position based policies due to the kinematics/dynamics model errors.

However, this approach alone can not generate motion necessary to perform tasks requiring complex policies, which are composed of a sequence of various primitives, such as pushing the door handle and opening the door. Another problem with this approach arises when there system has more non-constraint spatial directions then needed to perform the task. A typical practical example, which can occur is tasks like door opening, is a backlash, which manifests as an additional degrees of freedom. In order to overcome the above mentioned problem, we apply the reinforcement learning.

B. Learning of compound policy for door opening

In this section we present the implementation of the strategy described in the section II. The aim is to learn the appropriate force based policy, which will perform a compound sequence of operations such as first to unlatch the door with using the door handle, and then to open the door. The policy, defined as a sequence of the forces $\boldsymbol{F}^*$ and torques $\boldsymbol{M}^*$, is parametrized as a weighted sum of m Gaussian radial basis functions (RBFs) for each component j of the vector $\boldsymbol{F}^*$ and $\boldsymbol{M}^*$,

$$F_j^*(s) = \frac{\sum_{i=1}^m \boldsymbol{w}_{f,j,i}\Psi_i(s)}{\sum_{i=1}^m \Psi_i(s)} s, \quad (10)$$

$$M_j^*(s) = \frac{\sum_{i=1}^m \boldsymbol{w}_{m,j,i}\Psi_i(s)}{\sum_{i=1}^m \Psi_i(s)} s, \quad (11)$$

$$\Psi_i(s) = \exp\left(-h_i\,(s - c_i)^2\right), \quad (12)$$

where the free parameters $\boldsymbol{w}_{f,j,i}$ and $\boldsymbol{w}_{m,j,i}$ determine the shape of force and torque trajectories. Parameters c_i are the centers of RBFs, which are evenly distributed along the trajectory, h_i are the RBFs widths, and s denotes the phase variable, which will be defined latter.

For the policy learning, we have to implement a switching policy search strategy proposed in section II. In each learning cycle the robot first applies the forces and torques learned so far and checks if they result in a motion. If not, it perturbs them by some random forces and torques, generated as uniformly distributed random numbers with a specified variance,

$$F_j(s) = F_j^*(s) + \mathcal{N}(0, \sigma^2)$$
$$M_j(s) = M_j^*(s) + \mathcal{N}(0, \sigma^2),$$

where σ is the noise variation.

Whenever they result in a motion, it applies the control strategy given with (3) and (8). The algorithm collects intermediate costs, applied forces and torques. The roll-out is ended after the reaching the goal (which is when the door is opened in our case) or after the maximal allowed time for each episode has elapsed. At this time, the robot collects the terminal cost and calculates the new estimate of forces and torques for the next roll using the PI^2 RL algorithm [11]. This procedure is repeated until the robot learns the desired policy.

Intermediate and terminal cost were assigned as

$$c_i(k) = \begin{cases} 10|v_d - \|\dot{\boldsymbol{p}}(k)\||, & \dot{\varphi} > 0 \\ 10(0.1 - \|\boldsymbol{\omega}(k)\|), & \text{otherwise} \end{cases}$$

$$c_t = \begin{cases} t_o, & \varphi \geq \varphi_d \\ 10(\varphi_d - \varphi), & \text{otherwise} \end{cases}$$

where t_o is the time needed to open the door, and angles φ and φ_d denote the actual and the desired door opening angle, respectively. The terminal cost is thus lower if the robot opens the door in shorter time. If it can not open the door in the given time, the cost is proportional to the remaining angle needed to open the door. The intermediate cost is decreased if the robot rotates the handle in the first stage and if it opens the door with the right speed in the door opening stage. The set of updated policy parameters $\boldsymbol{w}_{f,j,i}$ and $\boldsymbol{w}_{m,j,i}$ are calculated with PI^2 after each learning cycle using all previous policy parameters, terminal and intermediate costs. A detailed description of the 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 [12]. In order to speed up the learning and to reject the unsuccessful attempts, the input data to PI^2 were reordered after each learning cycle using the importance sampling [13].

For successful learning, all signals, which are involved in learning, have to be of equal length and spatially aligned. Namely, we can not simply compare two actions, that happen at the same time. Rather, we have to compare actions, that happen at the same place. Spacial alignment is provided by the phase variable in the form

$$s = 1/s_0 \sum_{k=0}^{k=T} (\|\dot{\boldsymbol{p}}(k)\| + \zeta \|\boldsymbol{\omega}(k))\| dT, \qquad (13)$$

where ζ is a scaling factor to provide different scaling for positional and rotational movement and s_0 is an estimated constant, which assures that the phase remains within the interval $[0, 1]$.

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. In our implementation we scaled random search variance σ^2 by the factor $\alpha < 1$, after the completion of each roll-out. This choice assures smoother policy search in subsequent cycles.

The experimental evaluation of the proposed algorithm was implemented on a bi-manual robot composed of two KUKA LWR4 robot arms, equipped with a Barret hand and mounted on the torso with one rotational DOF. The task for the robot was to open a right handed door with the left robot arm while utilizing the torso rotation. The torso was used to avoid the joint limits of the robot arm, which was obtained by applying some motion on the torso in the vicinity of the joint limits (9) by using the self-motion of the complete robot system.

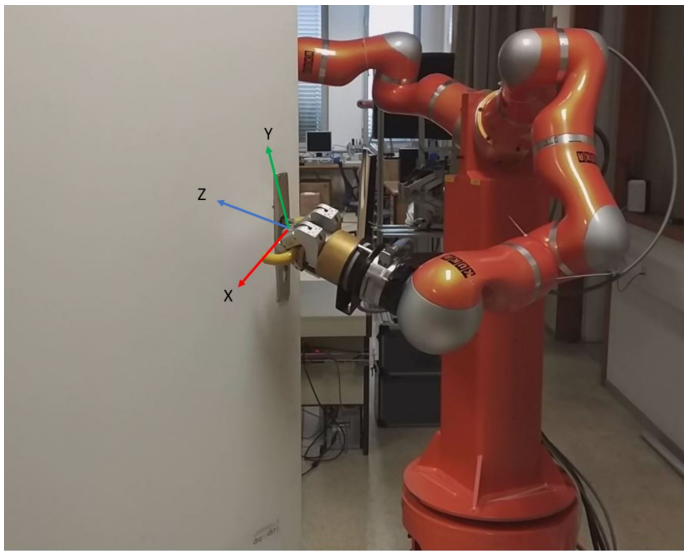

Fig. 2. Bi-Manual robot during door opening

The setup is shown in Fig. 2. The robot pose required to grasp the door handle was obtained with kinesthetic guiding in the zero gravity mode. Control and learning algorithms were implemented in MATLAB. The control (Eqs. (3) and (8)) was implemented at the sampling rate of 0.01 sec. On contrary, learning (Pi^2) was implemented at lower sampling rate of 0.4 sec. The control gains $\mathbf{K}_p$ and $\mathbf{K}_o$ were experimentally chosen as $50\ \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. They were chosen in such a way that the velocity controller had the desired tracking performance and that it exhibited stable operation in given environment conditions (e.g. door friction, door inertia, etc.) The initial value of the σ^2 was set to 50 (N) and α was 0.97. σ^2 is actually the only critical parameter, which affects both the speed of the learning and the learned policy. Too small values of σ^2 will result in slow learning, while excessive values can provoke erratic policies with excessive parameter variations. The length of the importance sampler was 3. 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 the learned forces/torques. The desired velocity v_d was set to 0.02 m/s and the desired opening angle φ_d was 70 deg. We performed 20 experiments with 20 roll-outs of learning. In average, the robot learned the required policy in 9 roll-outs. The learned force policy is shown in Fig. 3. Note that this plot shows learned forces in the tool coordinate system sampled at 0.4 sec interval. The actual forces sent to the robot controller are then smoothed and interpolated to 0.01 sec interval and mapped to the robot base frame. We can see that the robot has learned to manipulate the handle using force F_y and torque M_z, which both result in the handle rotation, providing that

the robot wrist is compliant. In average, the robot needs 6 sec to open the door after the learning.

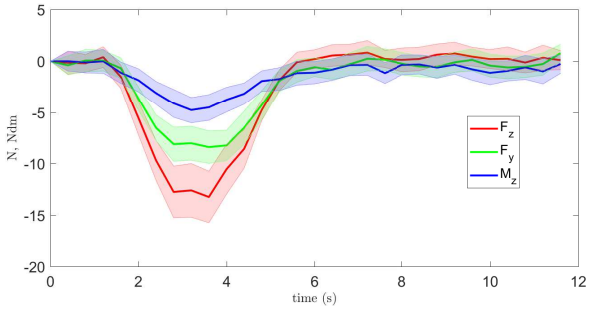

Fig. 3. Mean values and standard deviations (shaded regions) of learned force and torque profiles

IV. DRAWER OPENING

The second task was drawer opening. This task was implemented and tested in simulated environment. For this purpose, we applied state of the art MuJoCo HAPTIX simulation environment [14], which was used to simulate KUKA LWR 4 robot interacting with the environment. The environment model was a cabinet that comprised drawer, sliding doors, vertical doors and swing doors (see Fig. 4). In order to open the drawer, one has to lift first the drawer for approx. 1cm in vertical direction (Z) and then to pull it in the horizontal direction ($-X$). As no rotational movements are involved in this task, we were learning only forces in X, Y, and Z direction.

We applied identical control and learning algorithms with identical settings as for door opening experiments. The only difference was the definition of the cost function. The cost functions were defined as

$$c_i(k) = 10|v_d - \|\dot{\boldsymbol{p}}(k)\|\|$$

$$c_t = \begin{cases} t_o, & \boldsymbol{p}_x \geq p_l \\ 10(\boldsymbol{p}_l - \boldsymbol{p}_x), & \text{otherwise} \end{cases},$$

where p_l denotes the desired opening distance of the drawer.

Also in this cases we performed 20 learning experiments with 12 roll-outs. The robot learned the drawer opening policy in 6 roll-outs in average. The learned force policy is shown in Fig. 5. Note that here the forces are plotted in the robot base coordinate system. The blue line shows the vertical force F_z needed to unlatch the drawer. After the initial lifting the robot keeps this force since no additional cost was given for this action. The force F_x in the direction for the drawer opening actually opens the drawer. Note that the drawer could not be pulled out of the cabinet due to limiters. Therefore, the robot kept the learned forces also after the full opening of the drawer. The orthogonal force F_y, which does not affect the motion, was close to 0 all the time. The average terminal cost and the standard deviation of 12 learning cycles for this learning are shown in Fig. 6.

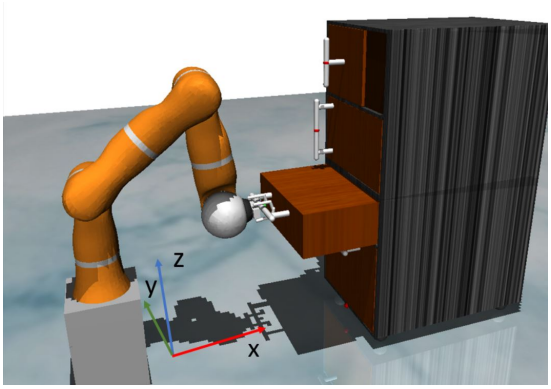

Fig. 4. Graphical output of MuJoCo simulation of drawer opening

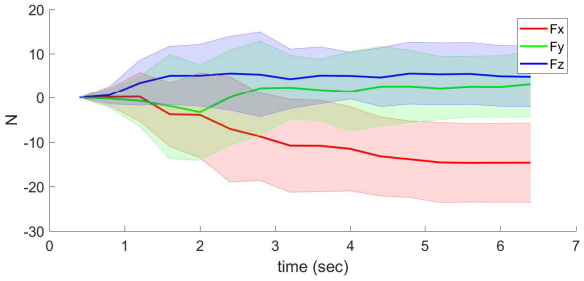

Fig. 5. Mean values and standard deviations (shaded regions) of learned force profiles

V. CONCLUSION

In the paper, we proposed a framework for learning of tasks, where the robot motion is constrained by the environment. The framework joins the benefits of two approaches: the ability of RL of model-free learning and the efficiency of the intelligent control. It is especially efficient in cases, where the motion of the robot is constrained by the environment and the robot can freely move in just some spatial directions. The task of the learning part of the algorithm is to determine the unconstrained degrees of freedom, which are then used for the task accomplishment. The underlying controller is acting as exploration in the action space [1]. With this algorithm, the robot efficiently learns many useful tasks such as door and drawer opening, closing a valve, pulling a lever, etc. It can learn also complex motions, e.g. where it is necessary to manipulate the door handle or latch, without any presumption how to do this. Note that in this study we learn force based policies, which are according to our opinion more appropriate for such tasks and easier to generalize to a small or moderate environment changes.

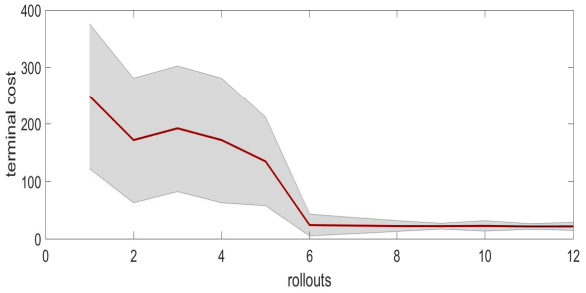

Fig. 6. Mean value (red line) and standard deviation (shaded region) of the terminal cost during learning of drawer opening in the simulated environment. Learning resulted in successful drawer opening at most after 6 roll outs in all cases.

The proposed approach was verified both in a simulated environment as well as with a real bi-manual robot, composed of the two KUKA LWR 4 robot arms mounted on the torso with 1 DOF. Both simulation and real experiment demonstrated 100% success of learning.

In this study, we neglected many important issues, such as how to grab the handle and how the learned forces affect the whole body motion of a humanoid robot or a robot on a mobile platform. Our future work, therefore, involves testing the algorithm in a more complex environment, minimization of interaction forces during the learning and application of the proposed framework to the humanoid robot Talos.

REFERENCES

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

[2] R. Ravani and A. Meghdari, "Velocity distribution profile for robot arm motion using rational Frenet-Serret curves," *Informatica*, vol. 17, no. 1, pp. 69–84, 2006.

[3] 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*, pp. 45–50, 1995.

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

[5] 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.

[6] 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.

[7] 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.

[8] Y. Karayiannidis, C. Smith, P. Ögren, and D. Kragic, ""Open Sesame!"Adaptive force/velocity control for opening unknown doors," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, p. 40404047, 2012.

[9] 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.

[10] B. Nemec, L. Zlajpah, and A. Ude, "Door opening by joining reinforcement learning and intelligent control," in *18th International Conference on Advanced Robotics (ICAR)*, 2017.

[11] 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.

[12] F. Stulp and O. Sigaud, "Path integral policy improvement with covariance matrix adaptation," in *Proceedings of the 10th European Workshop on Reinforcement Learning (EWRL 2012)*, vol. 1, 06 2012.

[13] 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, aug 2013.

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