IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 16, 2020, accepted April 1, 2020, date of publication April 9, 2020, date of current version April 29, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2986999

Accelerating Robot Trajectory Learning for Stochastic Tasks

JOSIP VIDAKOVIĆ, BOJAN JERBIĆ, BOJAN ŠEKORANJA, MARKO ŠVACO, AND FILIP ŠULIGOJ
Department of Robotics and Production System Automation, Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb, 10002 Zagreb, Croatia

Corresponding author: Josip Vidaković (josip.vidakovic@fsb.hr)

This work was supported in part by the Croatian Scientific Foundation through the Young Researchers' Career Development Project—training of new doctoral students, in part by the CRTA—Regional Centre of Excellence for Robotic Technologies Project, in part by the Centre of Excellence for Autonomous and Cooperative Robotic Systems—ACROSS, and in part by the NERO—Neurosurgical Robot Project.

ABSTRACT Learning from demonstration provides ways to transfer knowledge and skills from humans to robots. Models based solely on learning from demonstration often have very good generalization capabilities but are not completely accurate when adapting to new scenarios. This happens especially when learning stochastic tasks because of the correspondence problem and unmodeled physical properties of tasks. On the other hand, reinforcement learning (RL) methods such as policy search have the capability to refine an initial skill through exploration, where the learning process is often very dependent on the initialization strategy and is efficient in finding only local solutions. These two approaches are, therefore, frequently combined. In this paper, we present how the iterative learning of tasks can be accelerated by a learning from demonstration (LfD) method based on the extraction of via-points. The paper provides an evaluation of the approach on two different primitive motion tasks.

INDEX TERMS Learning from demonstration, policy search, robot task learning, robot trajectory.

I. INTRODUCTION

When trying to enable a robot to learn a task on the trajectory level, two main methods exist: Learning from Demonstration (LfD) methods and reinforcement learning (RL) methods. The LfD methods try to model human demonstrations into various statistical or dynamical models that aim at representing robot trajectories. On the other hand, RL methods use the robot exploration to find feasible policies for a given state space. The main challenge in LfD is to extract as much information from the demonstrations as possible and to encode generalizability into the model. In an ideal scenario, LfD can generalize a demonstrated task across the entire state-space. Task parametrization is a commonly used way to encode generalization capabilities into a trajectory model [1], [2], [3]. In the light of encoding generalizability, approaches that encode coupling terms into statistical models are shown in [4], [5] and [6]. Another group of approaches are those that automatically extract reward functions from demonstrations and optimize the trajectory with respect to them [7]. Learning from demonstration can be used with motion planning approaches in order to define the optimization criteria for motion planning algorithms [8], [9]. Researchers also try to provide complete solutions for robot learning, including the high level decomposition of robot tasks while also performing low level trajectory learning [10], [11]. We can include here end-to-end learning methods based on deep learning; these methods try to further minimize the required human engagement in the learning setup [12], [13].

Traditional reinforcement learning methods, on the other hand, rely on the state-value or action-value functions. As a model explores the whole state/action space, value/action functions are updated and a generalizable model is obtained [14]. The model can output an optimal policy for any given state. In robotics, however, state spaces are not deterministic. They are usually very large and complex; therefore, it is not often possible to explore them in such a way that a feasible function landscape can be provided. Tasks that involve continuous motion execution, as most robotic tasks do, require continuous action spaces. A robotic arm with six degrees of freedom has six action spaces, while the dimensionality of state spaces is dependent on a particular

The associate editor coordinating the review of this manuscript and approving it for publication was Huiyu Zhou.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

task. Thus, the popular curse of dimensionality phenomenon makes most of robotics-related problems impossible to learn by reinforcement learning in a generalizable way.

Another RL approach are policy search methods. These methods do not need to update the whole state-value or state-action space. Instead, only the current policy parameters are updated by performing rollouts [15]. This proved to be a good option for robot trajectory learning models because it does not require modeling of the system dynamics (transition probabilities). Some of the most famous policy search methods are gradient-based methods, such as REINFORCE [16], the path-integral PI^2 algorithm [17] and the Expectation-Maximization (EM)-based PoWER algorithm [18]. Policy search methods can be divided into black-box (BB) methods and white-box methods [22]. The difference between these two is that the black-box methods do not use any structure provided with the RL framework. This brings us to the main challenge of RL algorithms, i.e. the construction of a reward function. This process often re-quires a detailed analysis of the task and the design of a quantitative measure for important features. We distinguish two main strategies of reward functions, continuous (immediate) rewards and sparse rewards. The continuous reward functions, such as the REINFORCE and PI^2 algorithms, consider rewarding strategies that evaluate the policy during its execution time. The sparse reward approaches evaluate the policy only at the end, with a single reward value for the entire episode/rollout; this is characteristic of the BB policy search methods. In [19], it is shown that the BB optimization strategies can outperform the white-box, derivation-based strategies. An established BB optimization algorithm is CMA-ES which is used for policy search in many robotic applications.

A downside of the policy search methods is that they are suitable only for the local exploration of continuous action spaces. This prevents them from providing globally generalizable solutions on their own and makes the search process very dependent on the initial conditions. It is common practice to perform the initialization by human demonstrations. If a motion has to be learned, the learning is mostly initialized via kinesthetic teaching in order to minimize the correspondence problem. Examples of initializing RL algorithms with demonstrations can be found in [21] - a pancake flipping task, [22] - ball-in-cup task, [23] - table tennis movements, [24] – ironing and door opening, and [25] - dart and ball throwing. In most of these examples, generalization, if considered, is handled by the RL algorithm, e.g. by updating value functions.

In this paper, we propose a different approach, in which generalization is performed by an LfD method while the RL policy search is used to refine the movement for different situations. Our approach is presented as a framework for low level task-oriented trajectory learning. It combines the LfD method with the policy search algorithm; the former extracts information from demonstrations in order to provide the latter with a good initial trajectory. Using the black-box optimization strategy, we simplify the setup and make it applicable to a wide range of robotic tasks.

While the task-parameter approaches and the approaches that encode generalizability into statistical models ([4], [5], [6]) can encode information from demonstrations, they can suffer from locally optimal behaviors. The performance of the robot learner can be in the best case only as good as the performance of the teacher and cannot adopt a broader perspective on the solution space. On the other hand, the approach presented here is more similar to the reward-oriented approaches ([7]). It makes use of LfD to provide the robot with a good starting point for self-exploration in the policy search phase. The self-exploration process is only restricted by the initial learning rate and a simple task-oriented (result-oriented) cost function, but it is completely free in terms of exploring all parameters of the policy space. This leads to a more creative learning process, in which unorthodox solutions may be achieved.

II. METHODOLOGY

People initially transfer skills by demonstration: a teacher demonstrates a task to one or more apprentices and gives oral explanations in most cases. Such information serves as a good starting point for mastering a particular skill. After this initial phase, the learner is usually not able to accurately reproduce the skill autonomously. Therefore, he has to perform some trials on its own to fully grasp all parameters of the task during execution, where the number of trials depends on the circumstances such as the task complexity and the previous knowledge. Inspired by this, we have developed a framework for robot learning, which utilizes the LfD approach for the learning of an initial trajectory. This initial trajectory is then used as a starting point for the learning process refinement, where the robot tries to perform a task on its own. The proposed learning process will be described in the sections below.

A. DEMONSTRATION ANALYSIS AND TRAJECTORY GENERATION

Because of their local optimization behavior in robotics, policy search methods must be initialized by meaningful initial trajectories. When collaborative robot arms are considered, hand-guiding is a very popular way to do it. A human demonstrator holds the robot (or controls it indirectly via some teleoperation device) and guides it to perform a demonstration of the task. Such a movement is then used to calculate initial parameters of the movement policy.

The demonstration is usually done for only one specific situation. This way of initializing a trajectory is very good for individual cases but does not provide generalization capabilities. Tasks can have large state spaces, and only one demonstration can be insufficient to achieve the extraction of important features in the task space. This is the reason why we proposed to use a relative coordinate frame analysis presented

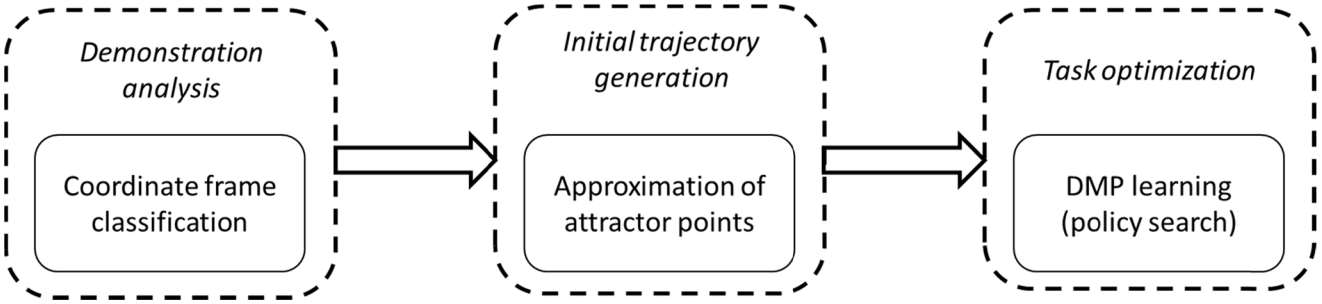

FIGURE 1. Flowchart of the proposed method. Learning is initialized by LfD.

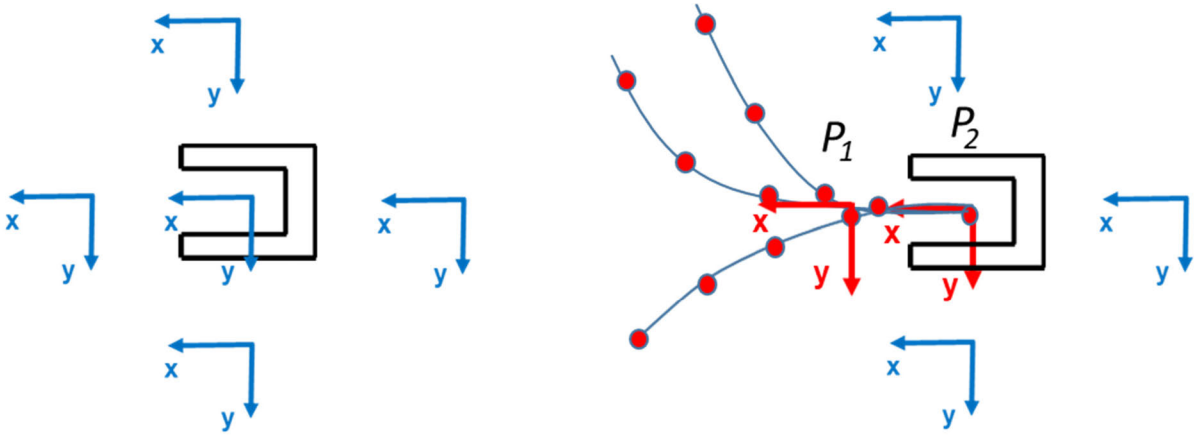

FIGURE 2. Classification of coordinate frames by using demonstrated trajectories. Left – an object with coordinate frames of interest. Right – coordinate frames of interest with respect to demonstrated trajectories.

in [26] to construct a generalized robot trajectory for every new configuration of the task space.

The method is explained in detail in [26], and we will describe it here only briefly. It uses a number of demonstrated trajectories (M) of a task and transforms them into coordinate frames of interest, where one object can have multiple frames N (Fig.2). The demonstrated trajectories are resampled to achieve the equal spacing of trajectory-position points (coordinate frames). The Euclidian distance from the frame of interest to every sampled demonstration point is calculated and an $M \times N$ distance matrix $\{\{d\}_{n=1}^{N}\}_{m=1}^{M}$ is constructed for every coordinate frame of interest $\boldsymbol{P_n}$. The distance matrix is then element-wise weighted by an exponential kernel function $\{\{d^w\}_{n=1}^{N}\}_{m=1}^{M} = x^d$, where $x \in \{0.1 \ldots 1\}$, that returns higher values for points closer to the coordinate frames of interest. Finally, the element sum of the weighted distance matrix gives a scalar value that describes the activity of a coordinate frame in the robot workspace given by the demonstrations.

Based on this scalar value and thresholding, coordinate frames can be classified into attractor frames, non-attractor frames and rejection frames. The attractor frames can then be used as via-points in order to construct a trajectory.

B. TASK OPTIMIZATION

In the same way as the knowledge obtained from observing a good teacher, a good initial trajectory should bring a robot close enough to the completion of a task. However, because of the correspondence problem, certain inaccuracies usually exist. Therefore, we perform an additional learning step based on the RL policy search.

In policy search, some of the most often used parametric policies for trajectories are via-points and splines, linear models, motor primitives, Gaussian mixture models (GMMs), and neural networks [22], [15]. In this study, we will focus on tasks that can be solved using primitive motions. These are movements with specified start and end points and with only one acceleration and one deceleration. Therefore, we will parametrize our policies using motor primitives, more specifically the Dynamic Movement Primitives (DMP).

The DMPs are considered as one of the best trajectory policies for learning [27]. They consist of a dynamical attractor-point system for every degree of freedom separately; they are then synchronized by an exponential decay function $\dot{t} = -\boldsymbol{\alpha_x} \cdot \boldsymbol{t}$ that serves as a central clock, where $\boldsymbol{\alpha_x}$ is the parameter that defines the falling rate of the decay function. Every dynamical system is influenced by a nonlinear function which is responsible for the shape of the trajectory. These

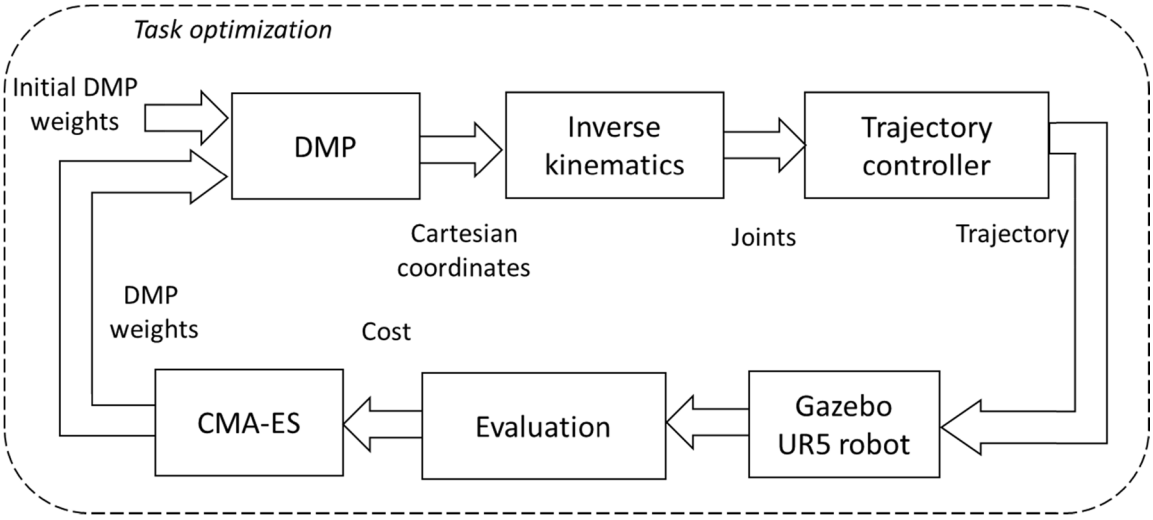

FIGURE 3. Block diagram of the proposed methodology.

nonlinear functions are parameterized by radial basis functions and their weights. Weights of the radial basis functions can be altered to achieve different shapes of the trajectory while maintaining its smoothness. This makes a DMP trajectory suitable for iterative learning in robotic applications. However, in order to generate a via-point trajectory, which is defined by the Cartesian states, we firstly use a slightly modified version of DMP, which is adapted from [28]. The acceleration of the system is calculated based on the sum of K proportional-derivative systems where K represents the number of states (or via-points), y and dy the current position and the velocity of the system, respectively and a_y and b_y are the gain parameters.

$$\hat{\dot{y}} = \sum_{i=1}^{K} h_i(t) \cdot a_y \left[b_y \left(\mu_i - y \right) - \dot{y} \right] \tag{1}$$

The time function is coupled with the dynamics of the system through a set of K Gaussians $N(t; \mu_i^t, \Sigma_i)$ with equally distributed centers over the period from zero to one where each of the Gaussians corresponds to a state in μ, where states are usually the Cartesian coordinates or joint positions. The time dependent probability that should be in a state is given by:

$$h_i(t) = \frac{\mathcal{N}(t; \mu_i^t, \Sigma_i)}{\sum_{k=1}^{K} \mathcal{N}(t; \mu_k^t, \Sigma_k)} \tag{2}$$

In order to transform this representation into a classical DMP as formulated in [27], weights of the radial basis functions are learned through a regression step, which is the most standard way of learning a DMP.

For task optimization, we conduct policy search using these weights as (initial) inputs. The search is based on the Covariance Matrix Adaptation Evolution Strategy (CMA-ES), which is an evolutionary black-box optimization algorithm suitable for optimizing non-linear stochastic functions with a large number of parameters [29]. The CMA-ES algorithm is used when derivative-based optimization methods fail due to discontinuities or unpredictable changes in the search landscape. This algorithm is adapted from [6]. CMA-ES is based on the principles of recombination and mutation combined with selection. In each iteration, only the best individuals become available as parents for the next iteration. The covariance matrix adaptation method is responsible for updating the search distribution, which is here represented by a covariance matrix. The algorithm has one open parameter and that is the initial step size.

III. SYSTEM SETUP

The system setup used for the validation of the presented methodology consists of two main parts, the demonstration and the task optimization setup. The experiments are performed with a UR5 robotic arm.

A. DEMONSTRATION SETUP

For the sampling of demonstrations, the positions of objects of interest are tracked using a Kinect 2.0 vision system and marker tracking. Trajectories of valid task executions are demonstrated by a human operator using kinesthetic teaching in the gravity compensation mode of the UR5. These trajectories are sampled at a fixed time rate (100 Hz) and logged on an external PC in the form of arrays of six-dimensional poses, together with the positions of the tracked objects.

B. TASK OPTIMIZATION SETUP

In this section a setup to test the proposed approach for iterative learning is presented (Fig. 3). The system is initialized by extracting the weights from the initial trajectory. The initial trajectory is then executed to evaluate its performance; in turn, it is used as a starting point for the search process. In every other iteration, the trajectory is refined by the search algorithms (CMA-ES) to obtain lower costs of the task execution.

The agent that executes the trajectories in every iteration is a UR5 robot implemented in the physics simulation

environment of Gazebo, where it interacts with the environment in order to perform a task. At the end of every robot movement, the outcome is evaluated using a cost function, which is minimized using the policy search algorithm. In general, cost/reward functions that are defined in the same space as the searched policy are expected to give best results when compared to scenarios where there is a mismatch between these two spaces [20]. As the task cost-functions are intuitively defined in the Cartesian space, the search process for trajectory improvement is also conducted in that space.

The robot trajectory controller used for the UR5 robot in Gazebo operates only in the joint-space which is the reason why after determining a new DMP trajectory, an inverse kinematic solver is used to obtain a trajectory sequence in the joint space.

When searching in the Cartesian space (operational space), the joint space constraints have to be considered explicitly in order to obtain feasible solutions that can be performed on a real robot manipulator. In our approach, we conducted the search by considering the task-oriented cost in the Cartesian space and the existence of kinematic solutions for the robotic manipulator used. Trajectories that had poses with no inverse kinematic (IK) solutions were penalized by a high cost and therefore neglected. When solving the IK problem for the Cartesian trajectory, all configuration solutions are considered at the start of the trajectory, and the joint configuration with the lowest maximum joint acceleration throughout the trajectory is used for task simulation. Additionally, the threshold values of the maximum acceleration and velocity are used ($80\,^\circ/s^2, 60\,^\circ/s$), while trajectories exceeding these values are penalized.

IV. RESULTS

The methodology was tested on two robotic tasks. In these tasks, the policy search algorithm was tested on two initialization strategies. The first involves a linear initialization strategy and the second uses the LfD approach. The linear initialization is performed by simply constructing a trajectory that goes from the start position of the robot to the final position of the task.

Although the presented tasks seem to be three dimensional, the policy search algorithm optimized all six degrees of freedom (DOF) at once in all experiments. The DMPs for every DOF consist of 10 basis functions, each having its own weight parameter that serves as an input to the optimization algorithm. This gives a total number of 60 parameters.

To evaluate the learning processes, we will use two criteria. The first one is the best solution obtained until the current iteration C_{best}, according to the smallest cost for the task execution, where n is the current iteration number. This enables the evaluation of the learning processes in terms of time (iterations) needed to achieve the best solution and the quality of the best solution.

$$C_{best} = \min(Cost_{n=1}^{n_{current}}) \tag{3}$$

The second criterion is the mean task-cost value observed until the current iteration C_{mean}. This criterion proved to be useful when analyzing the whole process of learning. A bad initialization of the policy search algorithm and poorly set exploration rates will lead to higher mean cost values.

$$C_{mean} = \text{mean}(Cost_{n=1}^{n_{current}}) \tag{4}$$

To validate the methodology while maintaining relatively low simulation time (less than one hour), the learning time in all experiments was 300 iterations. This number of iterations proved to be sufficient for the learning of satisfactory solutions in both tasks. In more complex tasks, the number of iterations can be increased.

In order to ensure the validity of a trajectory and to avoid false-positive solutions, we use multiple rollouts (three) in every episode, so that the average of the cost can be calculated for every trajectory. This provides a more stable learning process as a whole.

The initial step size for the CMA-ES algorithm is essentially the standard deviation of a search distribution for every parameter in the search space. In our tests, it was tuned manually for both tasks. The benchmarks for this process were the tasks with the linear initialization strategy. We started with small step sizes and increased them until the CMA-ES algorithm was able to provide sufficient exploration, without giving kinematically useless trajectories. When a suitable amount of exploration was observed for the linear initialization strategy, we divided those learning rates by a factor of 4 for the LfD initialization case. This empirically based choice of learning parameters proved to be reasonable for the two test cases. For the peg-in-hole task, the initial standard deviation was set to 50 for all parameters, while for the sweeping task it was set to 1. This big difference is due to the fact that the linear trajectory of the peg-in-hole task was very far from the solution trajectory.

At the end of each task subsection we tested the learned trajectories on the real UR5 robot. During the optimization process of task learning, the robot kinematics in terms of reachability, acceleration and velocity in the robot joint space is considered in the same way as explained earlier.

A. PEG IN HOLE

In the first example, we tested our approach on a common robotic manipulation task. The goal of the peg-in-hole task is to place the peg into the hole without colliding with the box that contains the hole. The setup of the peg-in-hole task can be seen in Fig. 4.

The cost function has two goals. The first one is not to move the box during the insertion of the peg. The second goal is to place the peg into the hole as close as possible to the target point. Therefore, the cost function C^{pih} is defined as a combination of the moved Euclidean distance of the box and the Euclidean distance of the final robot tool center point pose from the target point of

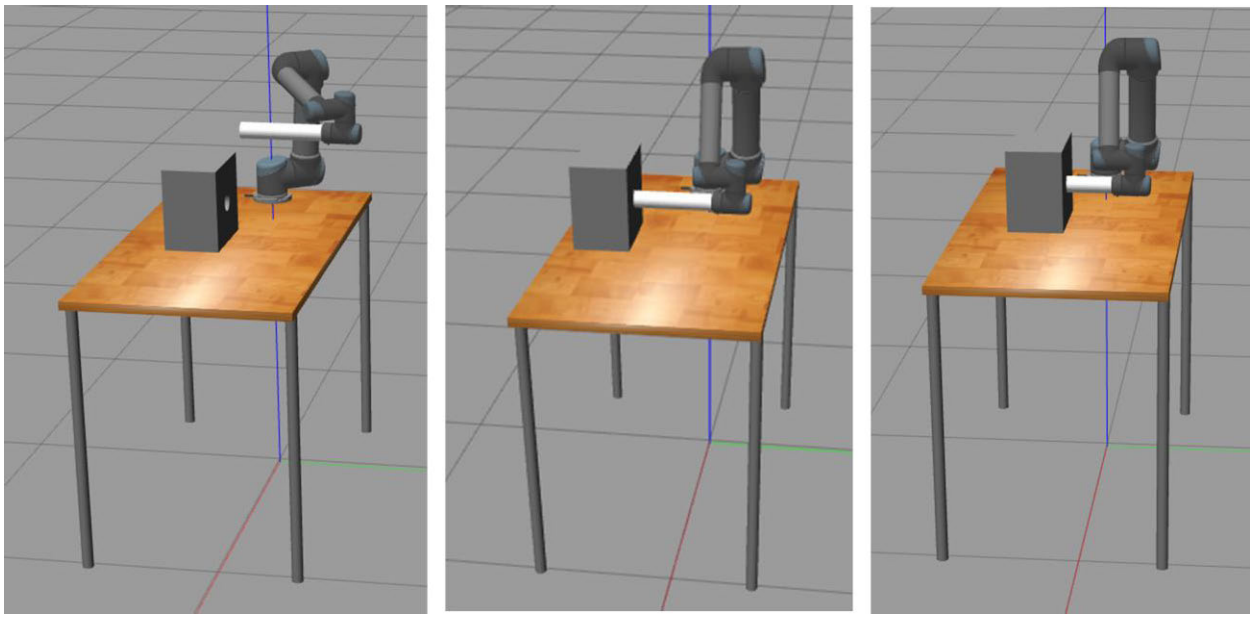

FIGURE 4. Peg-in-hole simulation setup and the sequence of the final learned trajectory.

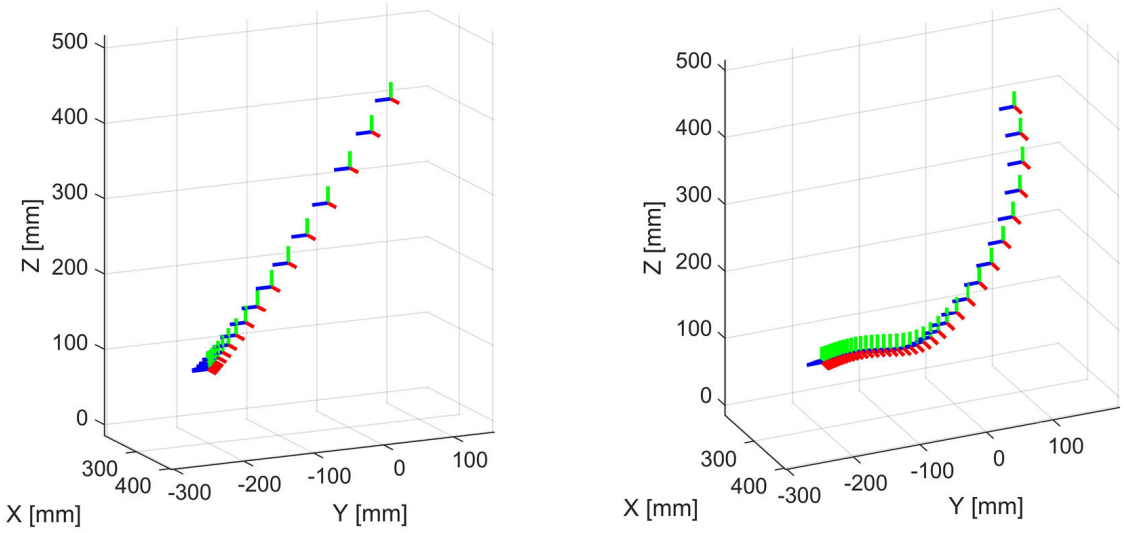

FIGURE 5. Initial trajectory (left) and the best learned trajectory (right) for the peg-in-hole task.

the hole.

$$C^{pih} = \|P_{box\,final} - P_{box\,initial}\| + \|P_{center\,of\,mass} - P_{TCP\,final}\| \tag{5}$$

In the policy search learning process, only the DMP weights and the goal position of the robot trajectory are used as learning parameters.

B. LINEAR INITIALIZATION

In the first test case, the LfD step was left out. Instead, the task optimization conducted using the policy search algorithm is initialized with a linear trajectory. This trajectory is constructed by assuming a linear movement from an arbitrary starting position to the known target position while keeping the orientation constant throughout the motion. Such a trajectory can be seen in Fig.5.

Because of the stochastic nature of the task (small changes in the trajectory parameters can result in unpredictable cost values), the task-oriented optimization process results in a stochastic learning process (Fig.6.). When evaluating the learning process through the current best solution, the best solution is found after approx. 125 iterations (Fig.6.). This solution results in a DMP trajectory that can perform the task with zero cost. The trajectory itself is shown in Fig. 5.

C. INITIALIZATION WITH LFD

The second test case involves the construction of the initial trajectory using the presented LfD approach. Demonstrations

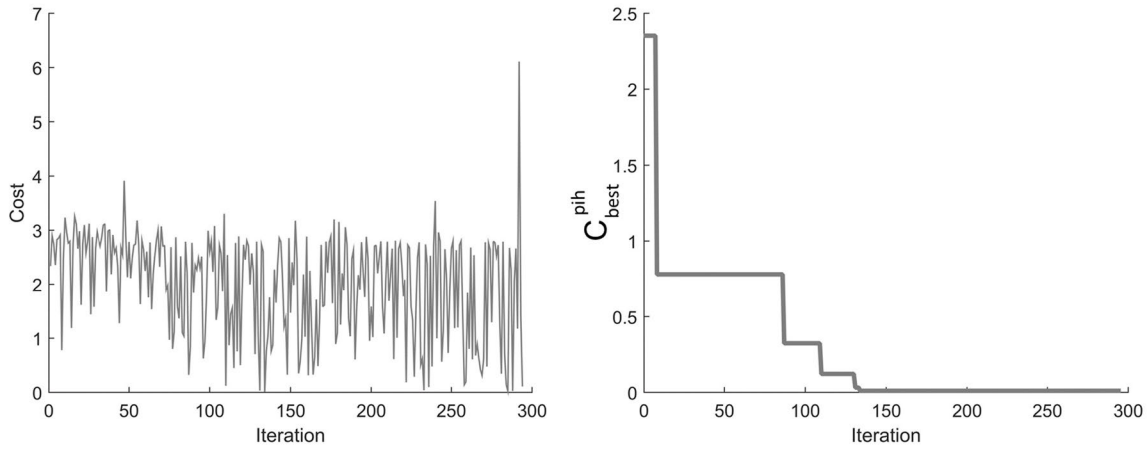

FIGURE 6. Raw cost values obtained in the learning process (left). Best current solution cost values obtained in the learning process (right) for the peg-in-hole task.

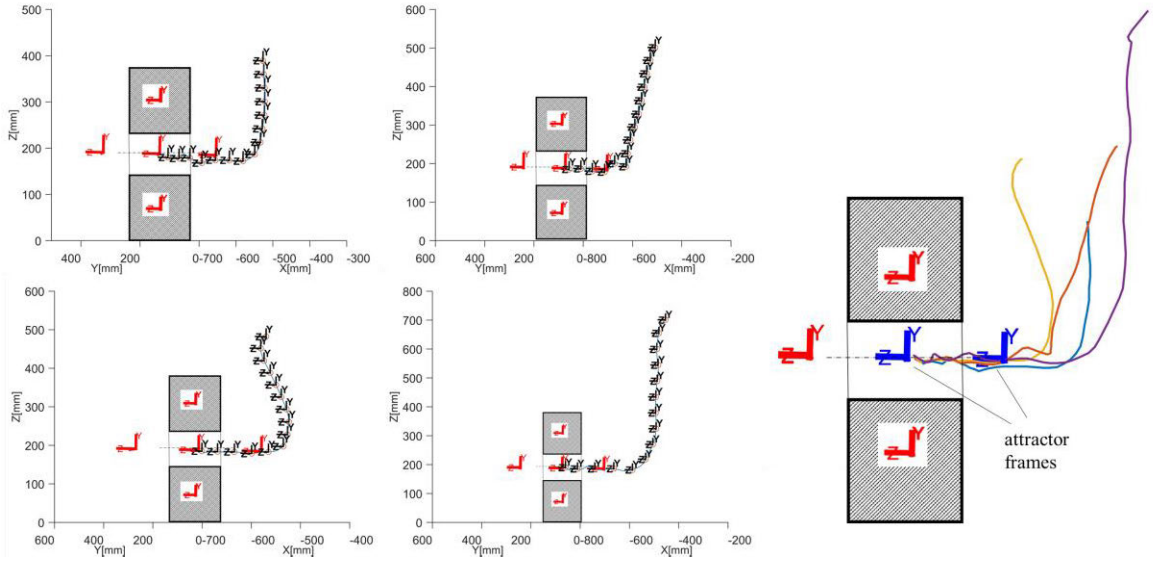

FIGURE 7. Demonstrations and classified attractor points for the peg- in-hole task. Left – individual demonstrations. Right – all the demonstrations in the same coordinate frame, together with the classified attractor frames (blue).

of the task for different task configurations were captured (Fig. 7.) and the analysis of the activity coordinate frames was performed according to the methodology described in section II.A.

Besides the default coordinate frame of the hole, an additional via-point coordinate frame was added to the trajectory based on the LfD coordinate frame analysis. By having the goal defined in the base of the hole and an additional approach via-point, the initial DMP trajectory can be constructed from any starting point. An example of the initial trajectory is shown in Fig. 8. – left.

The results of the policy search learning process initialized by such a trajectory are shown in Fig. 9. Looking at the best current solution metric, one can see that a valid solution for the task was found after approximately 60 iterations, which is significantly better than in the case of the linear initialization strategy. In terms of the quality of the solutions achieved with and without LfD, it is found that both solutions are approximately the same and both successfully solve the task.

To evaluate the stochastic learning processes obtained by the policy search algorithm, we use the proposed current mean cost criterion. When comparing the learning processes for the two initialization strategies by means of this metric, it can be observed that the LfD initialization provides better solutions, on average, during the whole learning process (Fig. 10.).

D. EVALUATION ON THE REAL ROBOT

We evaluated the simulation results on a real robot by performing the best learned trajectory in the joint space of the UR5 (Fig. 11.). The peg-in-hole task is dependent only on

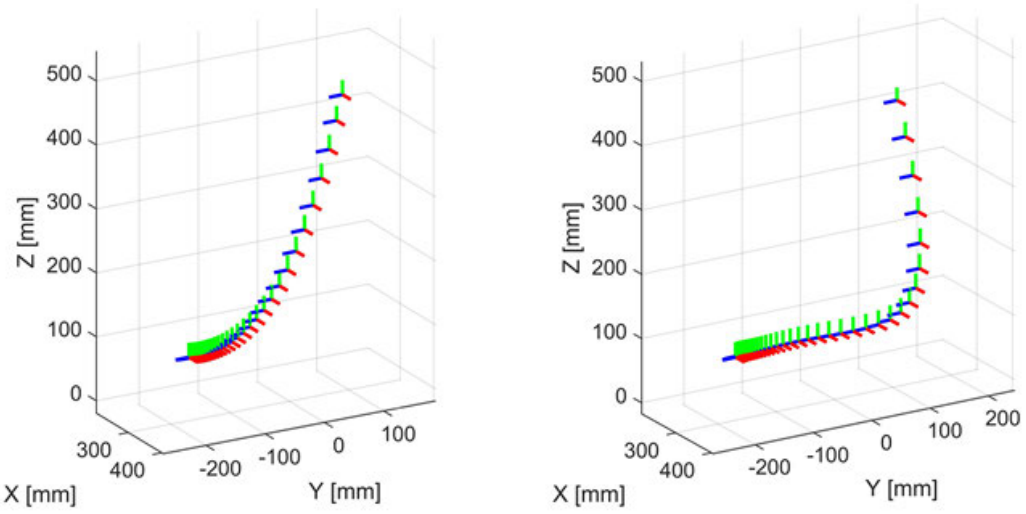

FIGURE 8. Initial trajectory (left) and final trajectory (right) for the peg-in-hole task.

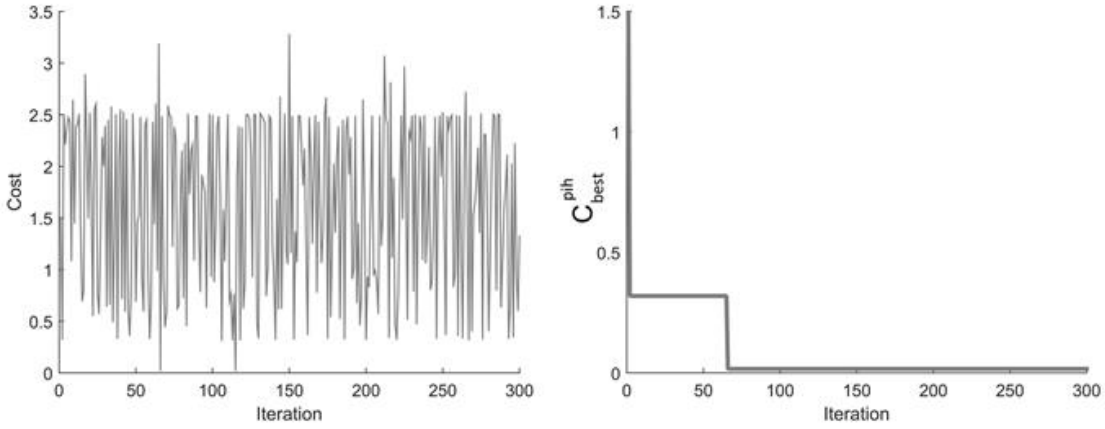

FIGURE 9. Raw cost values obtained in the learning process (left). Best current solution cost values obtained in the learning process (right).

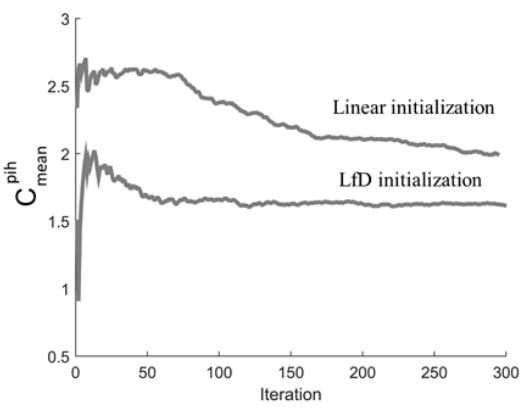

FIGURE 10. Comparison of the mean cost values for every iteration obtained during the learning process with linear initialization and the learning process with initialization performed with classified attractor frames as via-points.

the position of the hole and the robot trajectory, which makes it feasible for testing on the real robot. The learned trajectory performs the task successfully.

E. SWEEPING TASK

The second test case is the sweeping task. Here, the goal is to sweep a box lying in any of possible positions into a strictly defined position and orientation on the table. The robot is equipped with a simple rigid plate extension which can be utilized for pushing the box across the table. The cost function is here defined simply as the final distance of the final position of the swept box from the target position.

$$C^{sweeping} = \left\| P_{box\ target} - P_{box\ initial} \right\| \tag{6}$$

Because of the specific nature of the sweeping task, which involves environment factors like friction, we increased the search space. We added the starting pose and the target pose of the movement together with the movement velocity to the learning parameters. Together with the DMP weights, this leads to a total of 73 parameters.

F. LINEAR INITIALIZATION

Linear initialization is performed by creating a linear trajectory from the starting pose of the robot to the target pose

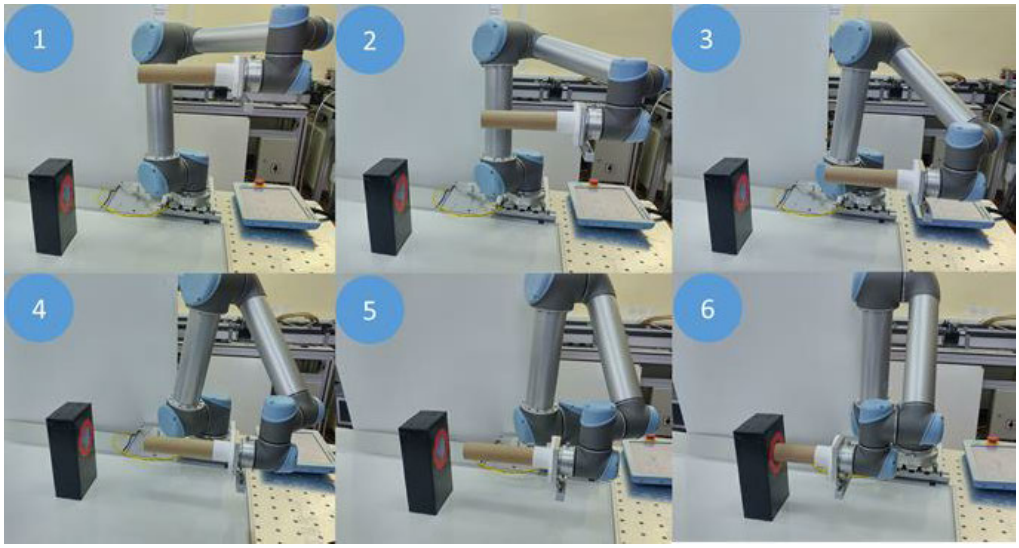

FIGURE 11. Sequence of motions for the peg-in-hole task performed by the UR5 robot.

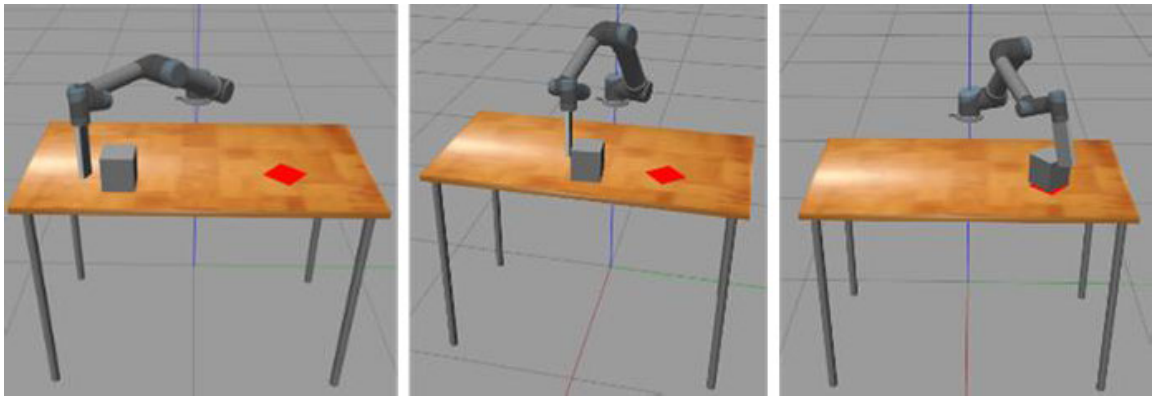

FIGURE 12. Final learned movement for the sweeping task. Left – box position and robot configuration at the start of the movement. Center – center of the movement. Right – end of the movement.

which is defined by the task-specified target destination of the sweeping movement, while maintaining a constant start orientation as shown in Fig. 13.

In the same way as in the peg-in-hole task, the sweeping task has proven to be very stochastic, which means that the margin for errors is very narrow. Starting from the linear trajectory, the policy search process is capable of finding a trajectory that solves the task after 130 iterations (Fig. 14.). The solution, however, relies on a trajectory with a relatively fast movement, where the robot does not follow the box throughout the entire path.

G. INITIALIZATION WITH LFD

Following the same methodology as in the peg-in-hole task, the sweeping task optimization policy search process is also initialized with a trajectory obtained from the LfD process. The sweeping task was demonstrated three times from various task configurations. The coordinate frame analysis was able to extract one additional via-point for both the target position and for the object position as the same approach pattern appeared in all demonstrations (Fig. 15.).

The initial trajectory constructed using two additional via-points for a new situation is shown in Fig. 16, together with the final learned trajectory. The policy search optimization is able to generate a good solution here after only 30 iterations (Fig. 17.).

By looking for the best current solution metric, convergence is achieved very quickly with a better quality of the obtained solution than that obtained with the linear initialization strategy.

The average solution metric also shows significantly better results of the learning process as a whole when the LfD initialization is used.

It can be observed that the final cost for both initialization strategies applied in this task does not converge to zero. This happens due to the fact that the sweeping task is not constrained in any direction of the movement (no guidance)

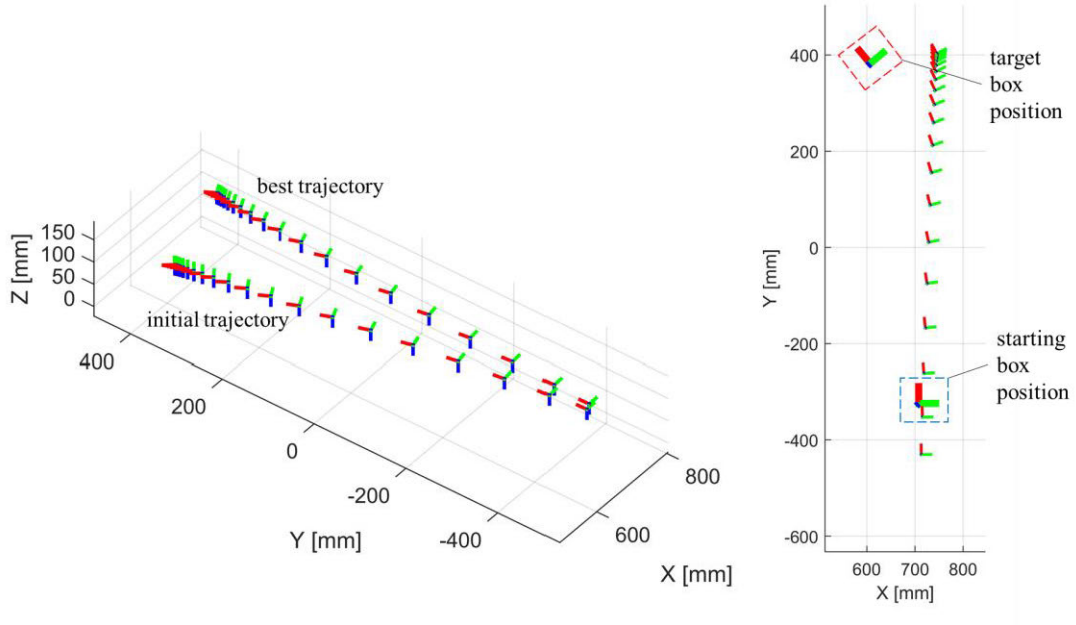

FIGURE 13. Linear initial trajectory and the best learned trajectory for the sweeping task (left). The best learned trajectory with respect to the initial box position and the target position (right).

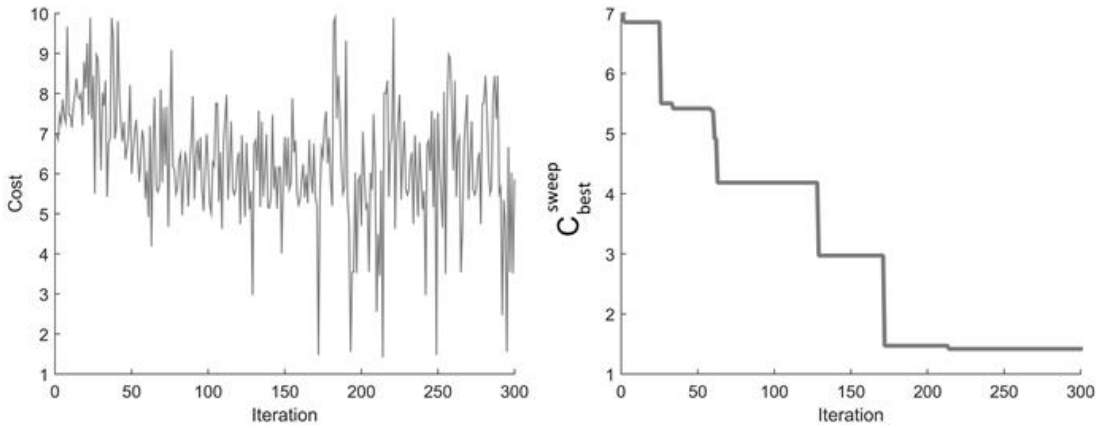

FIGURE 14. Raw cost values obtained in the learning process (left). Best current solution cost values obtained in the learning process (right).

and that it is very sensitive to small motion changes, which makes the task practically very difficult to perform with an absolute zero cost.

H. EVALUATION ON THE REAL ROBOT

Compared to the peg-in-hole task, the sweeping task is influenced by the environment to a higher degree. The result depends on the friction forces between the robot sweeping plate and the box, and between the box and the surface. This indicates a possible mismatch between the simulation environment and the real environment of the robot. However, in this case, the learned trajectory performs the task in the same way as in the simulation setup (Fig. 19.).

V. DISCUSSION AND FUTURE WORK

In the previous section we obtained results of learning for two robotic tasks. Because of the high sensitivity to parameter changes, it was not possible to obtain the classical raw-cost convergence of these tasks in any of the two cases despite the occurrence of qualitatively satisfying solutions in the search (learning) process. Therefore, we proposed the C_{best} and the C_{mean} evaluation metrics which enable a comparison of such stochastic cost/reward functions.

In the field of robotics, stochastic tasks are very challenging to learn. While modern model-based RL approaches are very data efficient [30], [31], [32], [33] for learning various robotic tasks, it still remains very difficult to learn a model for highly stochastic tasks. Despite their data

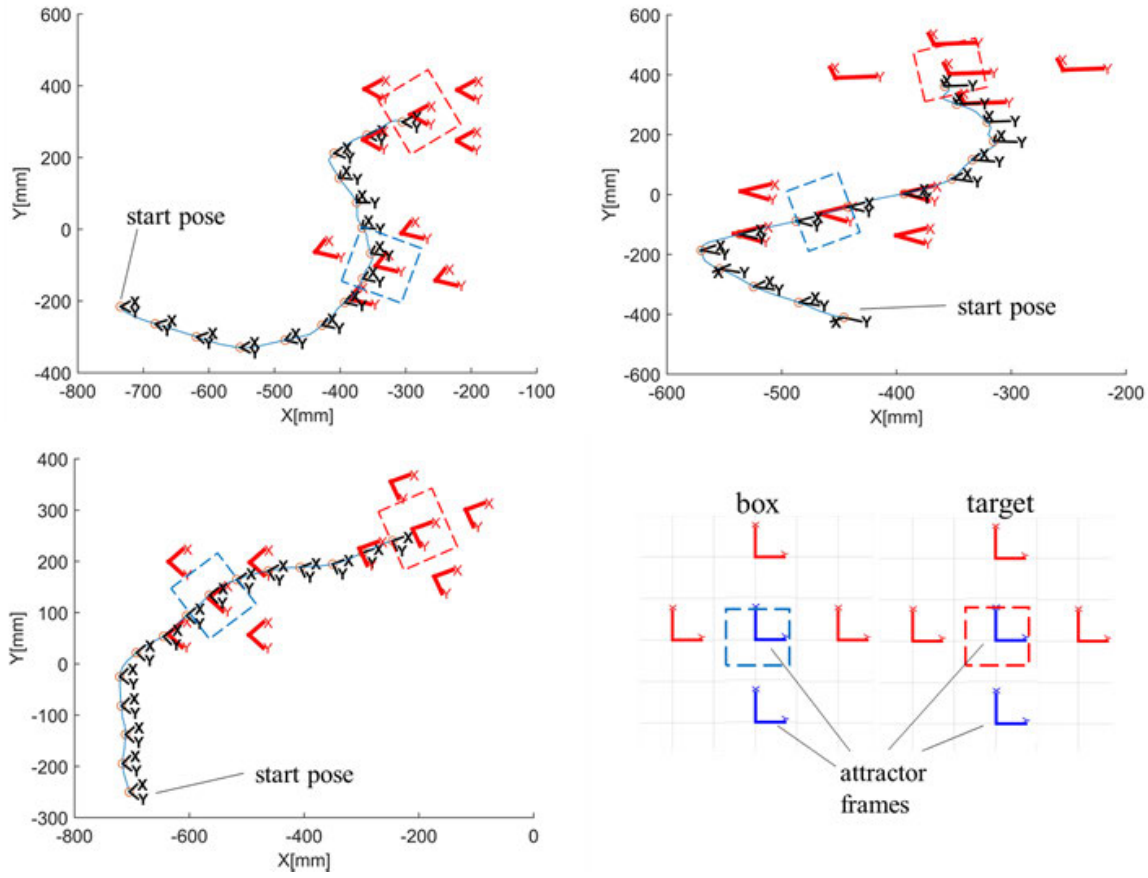

FIGURE 15. Demonstrations for the sweeping task and classified attractor points for the initial box position and the target.

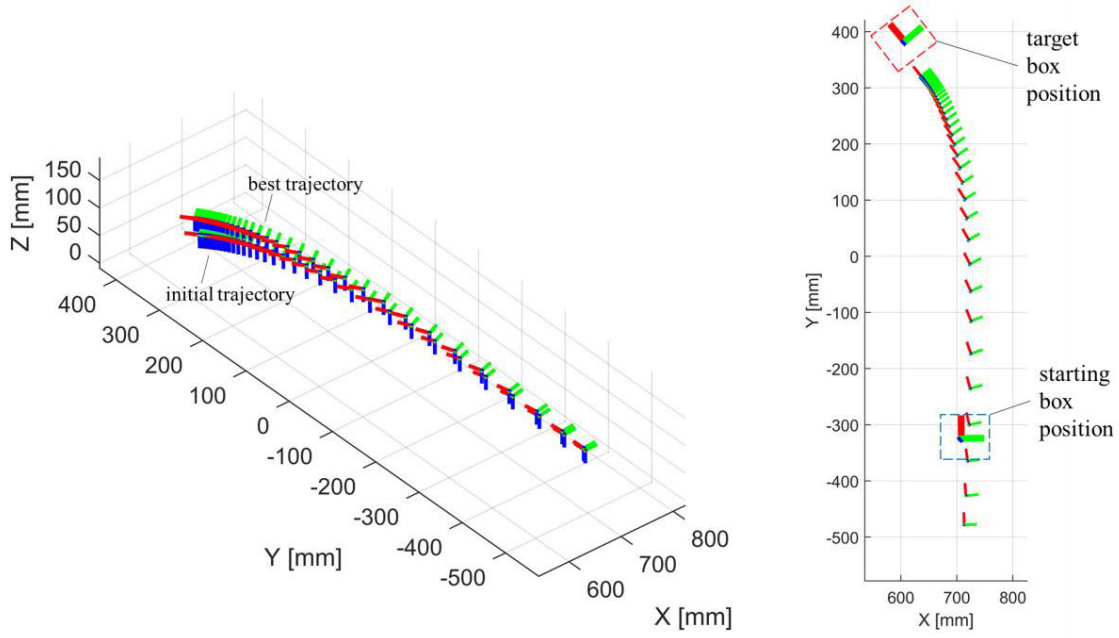

FIGURE 16. Trajectory initialized by using the attractor frames classified from the demonstration as via-points and the best trajectory (left). The best learned trajectory with respect to the initial box position and the target position (right).

inefficiency in terms of the number of rollouts required to learn a task, black-box policy search approaches are much simpler to implement and can cope with very stochastic learning environments. They are, therefore, used in many

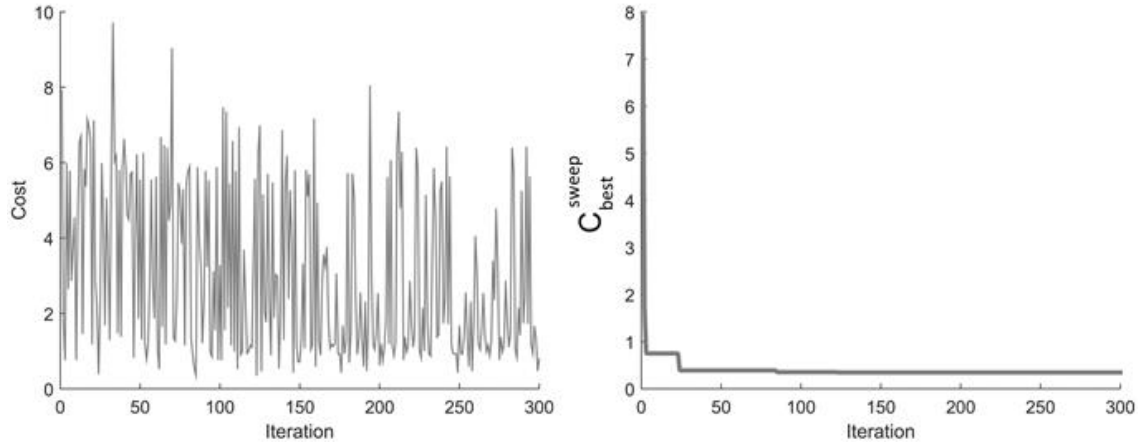

FIGURE 17. Raw cost values obtained in the learning process (left). Best current solution cost values obtained in the learning process (right).

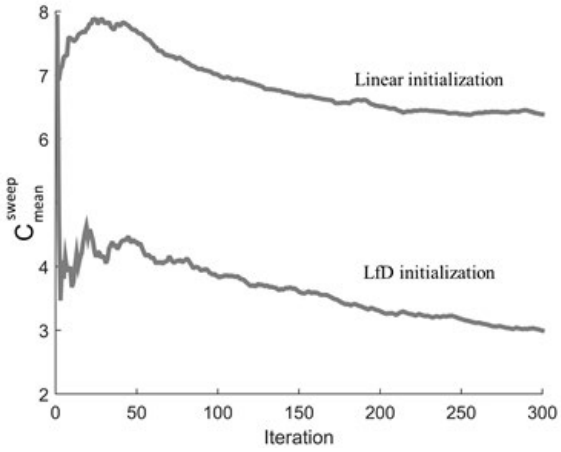

FIGURE 18. Comparison of the mean cost values for every iteration obtained during the learning process with linear initialization and that with initialization performed with classified attractor frames as via-points.

examples [34], [20], [35], [36]. Also, black-box policy search approaches do not require complex cost/reward functions, but can operate with sparse evaluation, which also greatly simplifies the learning setup and applicability. We show that the use of LfD can contribute to a much more efficient use of learning data when using a black-box stochastic optimizer like the CMA-ES algorithm for policy search. We initialized the algorithm using two strategies, the first one involving an LfD method and the second one without it. Results obtained from the two-task test scenarios show that initial trajectories that already encode some information are crucial to efficient learning of a task in robotics using the BB policy search.

Learning algorithms usually have some open parameters which need to be set manually. For the CMA-ES algorithm, this is the initial step size which can be also considered as an exploration rate. In order to make the policy search efficient, the exploration rate for policy search algorithms needs to be set according to the quality of the initial solution; if the initial trajectory is far from the final trajectory, higher exploration rates have to be set, and vice-versa. Too high exploration rates can lead to very slow convergence, while too low rates involve the risk of never converging at the right solution. In this research, the initialization of the policy search algorithm with trajectories obtained from the LfD step, proved to allow setting lower exploration rates without impairing the quality of the final solution.

Even though the presented approach was tested on two tasks, it is scalable to other robotic tasks; however, two constraints are imposed. The constraint of the LfD approach is that the tasks need to be demonstrated on the trajectory level with respect to objects of known poses in order to extract additional relevant via-points. The constraint of the policy search step is that it needs a reward function. However, it can be simply constructed for most tasks as it is a sparse cost function which evaluates only the end-result of the robot behavior. In terms of learning environments, we performed kinesthetic teaching on a real robot while performing the policy search task optimization step in simulation. To evaluate the final learned trajectory, the real robot is used. This setup exploited all the benefits of simulating robot learning with respect to the real-world learning, such as accelerated real-time clock for faster interactions (which shortens the learning time drastically) and no environment safety concerns which pose a big problem in the real-world robot learning. We have showed that this is a valid learning setup for the tasks that are not heavily influenced by environmental conditions. For tasks that are heavily influenced by the environment dynamics, the mismatch between the simulation and the real world would be, predictably, more emphasized.

Future work will be focused on finding more efficient algorithms for policy search with sparse evaluation. The other

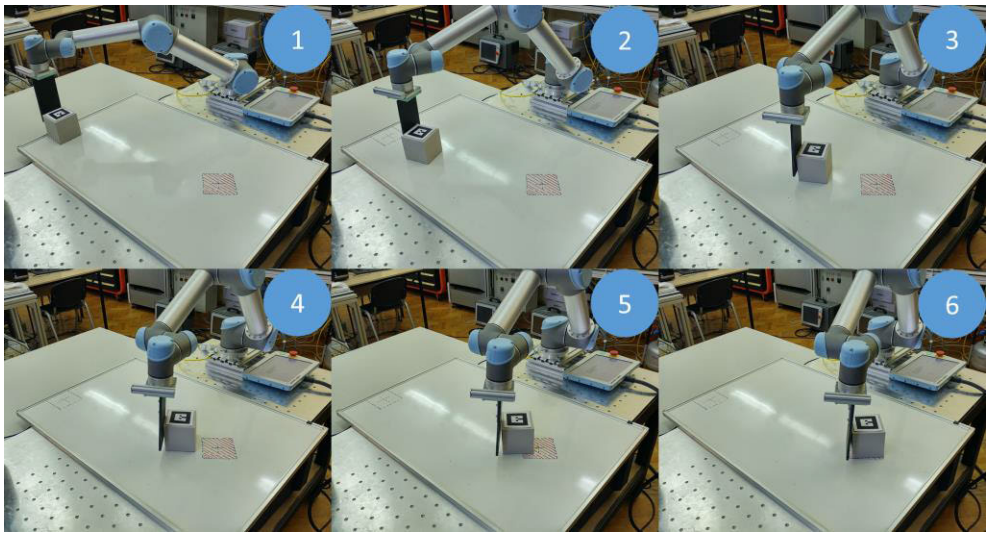

FIGURE 19. Sequence of motions for the sweeping task performed by the UR5 robot. The target position of the task is marked on the horizontal surface in red.

direction will be the automatic estimation of exploration rates at the beginning of the learning process and their potential online tuning, as shown in [37]. Also, the automatic extraction of end-result-oriented cost/reward functions will be a topic of interest.

VI. CONCLUSION

In this paper, we propose a methodology for performing task-oriented learning for continuous motion tasks. Learning from demonstration based on the classification of coordinate frames is used in order to initialize a task-oriented black-box policy search algorithm. The evolutionary CMA-ES algorithm is used in order to perform the task optimization, using DMPs as a policy representation. A simulation setup for task optimization is also presented.

We tested the approach in two robotic test scenarios: a peg-in-hole task and a sweeping task. Two initialization strategies were compared in terms of the learning speed and the current mean cost. It is shown that by initializing the policy search with LfD, we are able to accelerate the learning process with respect to the initialization strategy with no knowledge obtained from demonstrations.

REFERENCES

[1] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intell. Service Robot.*, vol. 9, no. 1, pp. 1–29, Jan. 2016, doi: 10.1007/s11370-015-0187-9.

[2] A. Pervez and D. Lee, "Learning task-parameterized dynamic movement primitives using mixture of GMMs," *Intell. Service Robot.*, vol. 11, no. 1, pp. 61–78, Jan. 2018, doi: 10.1007/s11370-017-0235-8.

[3] A. L. P. Ureche, K. Umezawa, Y. Nakamura, and A. Billard, "Task parameterization using continuous constraints extracted from human demonstrations," *IEEE Trans. Robot.*, vol. 31, no. 6, pp. 1458–1471, Dec. 2015, doi: 10.1109/TRO.2015.2495003.

[4] A. Rai, G. Sutanto, S. Schaal, and F. Meier, "Learning feedback terms for reactive planning and control," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2017, pp. 2184–2191, doi: 10.1109/ICRA.2017.7989252.

[5] A. Gams, M. Denisa, and A. Ude, "Learning of parametric coupling terms for robot-environment interaction," in *Proc. IEEE-RAS 15th Int. Conf. Humanoid Robots (Humanoids)*, Seoul, South Korea, Nov. 2015, pp. 304–309, doi: 10.1109/HUMANOIDS.2015.7363559.

[6] T. Alizadeh, M. Malekzadeh, and S. Barzegari, "Learning from demonstration with partially observable task parameters using dynamic movement primitives and Gaussian process regression," in *Proc. IEEE Int. Conf. Adv. Intell. Mechatronics (AIM)*, Banff, AB, Canada, Jul. 2016, pp. 889–894, doi: 10.1109/AIM.2016.7576881.

[7] A. M. E. Ghalamzan, C. Paxton, G. D. Hager, and L. Bascetta, "An incremental approach to learning generalizable robot tasks from human demonstration," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, Seattle, WA, USA, May 2015, pp. 5616–5621, doi: 10.1109/ICRA.2015.7139985.

[8] M. A. Rana, M. Mukadam, S. R. Ahmadzadeh, S. Chernova, and B. Boots, "Skill generalization via inference-based planning," in *Proc. RSS Workshop Math. Models, Algorithms, Hum. Robot Interact.*, 2017, pp. 1–3.

[9] M. A. Rana, M. Mukadam, S. R. Ahmadzadeh, S. Chernova, and B. Boots, "Towards robust skill generalization: Unifying learning from demonstration and motion planning," in *Proc. 1st Annu. Conf. Robot Learn. (PMLR)*, vol. 78, 2017, pp. 109–118.

[10] D. A. Duque, F. A. Prieto, and J. G. Hoyos, "Trajectory generation for robotic assembly operations using learning by demonstration," *Robot. Comput.-Integr. Manuf.*, vol. 57, pp. 292–302, Jun. 2019, doi: 10.1016/j.rcim.2018.12.007.

[11] N. Figueroa, A. L. P. Ureche, and A. Billard, "Learning complex sequential tasks from demonstration: A pizza dough rolling case study," in *Proc. 11th ACM/IEEE Int. Conf. Hum.-Robot Interact. (HRI)*, Mar. 2016, pp. 611–612.

[12] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1334–1373, 2016.

[13] R. Rahmatizadeh, P. Abolghasemi, L. Boloni, and S. Levine, "Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, Brisbane, QLD, Australia, May 2018, pp. 3758–3765, doi: 10.1109/ICRA.2018.8461076.

[14] M. Švaco, B. Jerbić, M. Polančec, and F. Šuligoj, "A reinforcement learning based algorithm for robot action planning," in *Proc. 27th Int. Conf. Robot. Alpe-Adria-Danube Region (RAAD)*, Patras, Greece. Berlin, Germany: Springer, 2018, pp. 493–503.

[15] M. P. Deisenroth, "A survey on policy search for robotics," *Found. Trends Robot.*, vol. 2, nos. 1–2, pp. 1–142, 2011, doi: 10.1561/2300000021.

[16] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, pp. 229–256, May 1992.

[17] E. Theodorou, J. Buchli, and S. Schaal, "Learning policy improvements with path integrals," *J. Mach. Learn. Res.*, vol. 9, pp. 828–835, Jan. 2010.
[18] J. Kober and J. Peters, "Policy search for motor primitives in robotics," *Mach. Learn.*, vol. 84, nos. 1–2, pp. 171–203, Jul. 2011, doi: 10.1007/s10994-010-5223-6.
[19] F. Stulp and O. Sigaud, "Policy improvement methods: Between black-box optimization and episodic reinforcement learning," *J. Francophones Planification, Décision, Apprentissage Pour la Conduite Syst.*, pp. 1–15, Jul. 2013.
[20] A. Fabisch, "A comparison of policy search in joint space and Cartesian space for refinement of skills," in *Proc. Int. Conf. Robot. Alpe-Adria Danube Region*, vol. 980, 2020, pp. 301–309, doi: 10.1007/978-3-030-19648-6_35.
[21] P. Kormushev, S. Calinon, and D. G. Caldwell, "Robot motor skill coordination with EM-based reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2010, pp. 3232–3237.
[22] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1238–1274, Sep. 2013, doi: 10.1177/0278364913495721.
[23] K. Mülling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *Int. J. Robot. Res.*, vol. 32, no. 3, pp. 263–279, Mar. 2013, doi: 10.1177/0278364912472380.
[24] P. Kormushev, S. Calinon, and D. G. Caldwell, "Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input," *Adv. Robot.*, vol. 25, no. 5, pp. 581–603, Jan. 2011, doi: 10.1163/016918611X558261.
[25] J. Kober, A. Wilhelm, E. Oztop, and J. Peters, "Reinforcement learning to adjust parametrized motor primitives to new situations," *Auton. Robots*, vol. 33, no. 4, pp. 361–379, Nov. 2012, doi: 10.1007/s10514-012-9290-3.
[26] J. Vidaković, B. Jerbić, B. Šekoranja, M. Švaco, and F. Šuligoj, "Learning from demonstration based on a classification of task parameters and trajectory optimization," *J. Intell. Robot. Syst.*, pp. 1–15, Dec. 2019, doi: 10.1007/s10846-019-01101-2.
[27] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Comput.*, vol. 25, no. 2, pp. 328–373, Feb. 2013, doi: 10.1162/NECO_a_00393.
[28] S. Calinon, I. Sardellitti, and D. G. Caldwell, "Learning-based control strategy for safe human-robot interaction exploiting task and robot redundancies," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2010, pp. 249–254.
[29] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evol. Comput.*, vol. 9, no. 2, pp. 159–195, Jun. 2001, doi: 10.1162/106365601750190398.
[30] A. F. de Broissia and O. Sigaud, "Actor-critic versus direct policy search: A comparison based on sample complexity," *Proc. Journees Francaises Planification Decis. Apprentissage*, pp. 1–9, Aug. 2016.
[31] M. Zhang, S. Vikram, L. Smith, P. Abbeel, M. J. Johnson, and S. Levine, "SOLAR: Deep structured representations for model-based reinforcement learning," 2018, *arXiv:1808.09105*. [Online]. Available: http://arxiv.org/abs/1808.09105
[32] O. Sigaud and F. Stulp, "Policy search in continuous action domains: An overview," *Neural Netw. J.*, vol. 113, pp. 28–40, May 2019.
[33] M. P. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *Proc. 28th Int. Conf. Mach. Learn. (ICML)*, 2011, pp. 465–472.
[34] X. Li, Z. Liang, and H. Feng, "Kicking motion planning of nao robots based on CMA-ES," in *Proc. 27th Chin. Control Decis. Conf. (CCDC)*, Qingdao, China, May 2015, pp. 6158–6161, doi: 10.1109/CCDC.2015.7161918.
[35] F. Stulp and O. Sigaud, "Robot skill learning: From reinforcement learning to evolution strategies," *J. Paladyn Behav. Robot.*, vol. 4, no. 1, Jan. 2013, doi: 10.2478/pjbr-2013-0003.
[36] A. Abdolmaleki, B. Price, N. Lau, L. P. Reis, and G. Neumann, "Deriving and improving CMA-ES with information geometric trust regions," in *Proc. Genet. Evol. Comput. Conf. (GECCO)*, Berlin, Germany, 2017, pp. 657–664, doi: 10.1145/3071178.3071252.
[37] S. Li, C.-M. Chew, and V. Subramaniam, "Smooth and efficient policy exploration for robot trajectory learning," in *Proc. 27th IEEE Int. Symp. Robot Hum. Interact. Commun. (RO-MAN)*, Nanjing, China, Aug. 2018, pp. 1087–1092, doi: 10.1109/ROMAN.2018.8525631.

JOSIP VIDAKOVIĆ received the Diploma degree in mechanical engineering from the Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb (UNIZAG), in 2014, where he is currently pursuing the Ph.D. degree. He is currently a Research Assistant with the Faculty of Mechanical Engineering and Naval Architecture, UNIZAG. His current research interests include the fields of industrial and surgical robotics, artificial intelligence methods in robotics, and motion planning and learning from demonstration.

BOJAN JERBIĆ received the B.Sc., M.Sc., and Ph.D. degrees from the Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb, in 1983, 1987, and 1993, respectively. He is currently a Full Professor and the Head of the Chair of Manufacturing and Assembly System Planning. He is a member of the Scientific Board for Technology Development with the Croatian Academy of Sciences and Arts, the Croatian Robotics Society, and the Department of Systems and Cybernetics, Croatian Academy of Engineering. His main research interests include cognitive robotics, artificial intelligence, multiagent systems, new robotic applications, and medical robotics.

BOJAN ŠEKORANJA received the Diploma and Ph.D. degrees in mechanical engineering from the Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb (UNIZAG), in 2009 and 2015, respectively. He is currently a Postdoctoral Researcher and Teaching Assistant with the Faculty of Mechanical Engineering and Naval Architecture, UNIZAG. His current research interests include industrial and surgical robotics, and human–robot interaction.

MARKO ŠVACO received the B.Sc. degree in mechanical engineering, the M.Sc. degree in computer-aided engineering with a specialization in intelligent assembly systems, and the Ph.D. degree in robotics and automation from the Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb (UNIZAG), Croatia, in 2008, 2009, and 2015, respectively. Since 2012, he has been an Associate with the Department of Neurosurgery, University Hospital Dubrava, Zagreb, Croatia. He is currently an assistant professor at the Faculty of Mechanical Engineering and Naval Architecture, UNIZAG. His current research interests include the fields of industrial and medical robotics, artificial intelligence methods in robotics, and computer vision.

FILIP ŠULIGOJ received the B.Sc. and M.Sc. degrees in mechanical engineering and the Ph.D. degree in robotics and automation from the Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb (UNIZAG), in 2008, 2009, and 2018, respectively. He currently works as a Postdoctoral Researcher and a Senior Teaching Assistant with the Faculty of Mechanical Engineering and Naval Architecture, UNIZAG. His current research activities are focused on the development of imaging methods and the control of multiagent robot systems with applications in surgical procedures. His research interests include the fields of industrial and medical robotics, machine vision, and artificial intelligence methods.