Autonomous learning of assembly policy

Mihael Simonič
Jožef Stefan Institute
Ljubljana, Slovenia
mihael.simonic@ijs.si

Aleš Ude
Jožef Stefan Institute
Ljubljana, Slovenia
ales.ude@ijs.si

Bojan Nemec
Jožef Stefan Institute
Ljubljana, Slovenia
bojan.nemec@ijs.si

ABSTRACT

In the paper, we propose to learn an assembly task from the corresponding disassembly. Autonomous learning of disassembly can be easier than learning of the corresponding assembly task, because the admissible set of motions during disassembly is initially fully constrained by the environment. During the disassembly the robot exploits its compliance in order to detect admissible motions and takes appropriate decisions when multiple options exist. Learning of the disassembly was realized using hierarchical reinforcement learning. The disassembly policy is then used to derive the corresponding assembly policy. The proposed approach was experimentally validated on the case of light-bulb assembly.

KEYWORDS

reinforcement learning, robot learning, autonomous assembly

1 INTRODUCTION

Developing robust assembly skills is one of the main challenges in contemporary robotics. Assembly skills are needed not only in production plants, but will also be important for the future generation of home and service robots. For fast deployment of such tasks, new user-friendly tools for programming robot operations are needed. Ideally, a robot would be able to derive assembly policy autonomously.

Autonomous policy learning, is usually accomplished by utilizing reinforcement learning. Starting from an existing parameterized policy, a robot tries to adapt to a new situation by randomly changing task parameters and find out how to modify the policy to maximize the reward function [9, 12]. However, the main challenge is huge search space which characterizes an assembly policy. For that reason, there were very few successful attempts of completely autonomous learning of assembly tasks in robotics [4, 7]. Existing techniques for reducing the search space of reinforcement learning usually assume prior information about process, either in an explicit form or inherited from previous experiments and therefore still rely on skilled robot operators that guide the robot through the learning process [5].

In our previous research, we proposed an alternative approach to autonomous policy learning, which unifies compliant motion control and reinforcement learning. Tasks that involve interaction with the environment are traditionally considered as extremely hard to learn due to the unknown and possibly changing environment. On the other hand, interacting with the environment can be advantageous to accelerate the learning process. Namely, if appropriately addressed, learning of physically constrained tasks is more efficient than the learning of tasks, where a robot can move completely freely in space. The reason is that the environment limits the admissible movement directions. Consequently, the number of parameters that need to be learned can be greatly reduced. To implement this type of learning, we need to make use of the natural robot motion along with the constraints imposed by the environment. Compliant robot control provides a suitable framework for implementing such a strategy. This concept has been already successfully applied to the learning of tasks such as autonomous learning for doors and drawers opening [8].

In this paper, we present how the above-described methodology can be extended to autonomous learning of assembly operations. The main idea is that robot first learns the reverse action – disassembly of an object. In an assembled object, the set of possible motions is constrained, and typically only a single motion or operation is possible. During the disassembly, the motion becomes less and less constrained until the part is completely disassembled and the environment no more constrains motion of individual parts. The situation is opposite during the assembly. The initially virtually unlimited set of possible motions becomes more and more constrained as the assembly process advances. Given no previous knowledge about the task, learning of disassembly is therefore more straightforward than learning of the assembly task. Imagine generic peg-in-hole task: by removing a peg from a hole, we also learn the exact pose of the hole, whereas we would first have to guess where the hole is if we are to insert the peg into the hole without any prior knowledge.

Similarly, we transfer the knowledge obtained during disassembly to the corresponding assembly process. We assume that the initial assembly policy can be obtained by reverse execution of the learned disassembly policy. This is possible because in most cases assembly and disassembly are mutually reversible operations. Common assembly tasks such as putting/placing, peg-in-hole, or screwing are directly reversible [6]. Tasks that result in structural deformations or require external equipment (e.g. riveting pistol and rivets) are not directly reversible, but can be omitted for the purposes of disassembly learning and manually added to the final assembly policy.

This paper is structured as follows. We first introduce our algorithm for hierarchical reinforcement learning on the example of maze learning in Section 2. Then we present the underlying intelligent controller in Section 3. In section 4, we present our methodology to learn assembly policy from disassembly policy, along with experiential verification of the proposed framework in Section 5. We conclude with a short summary.

IS2019, October 7–11, 2019, Ljubljana, Slovenia, Europe

2 HIERARCHICAL REINFORCEMENT LEARNING

In the reinforcement learning (RL) literature, maze learning has been traditionally used as a benchmark for validating learning algorithms. Maze learning also bears a lot of similarities with disassembly process, where the robot should come from an initially fully contained state into the final unconstrained state. Within a maze, the agent mostly follows the corridors and only has to take decisions in the crossings.

Traditional approaches rely on discrete state-space with predefined set of actions as illustrated in Figure 1.

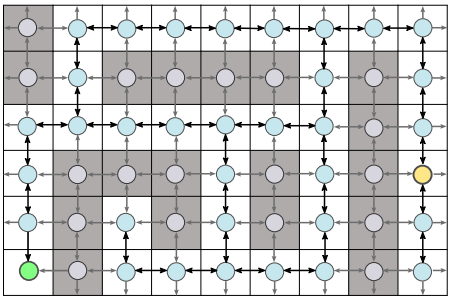

Figure 1: An example of maze with 9×11 cells. White cells represent corridors where the agent can move, whereas gray cells are walls. The state space for maze learning is represented with a graph. In each state (represented as node), the agent can choose from a fixed set of possible actions (represented with edges): relative left, right, up and down movement. The agent starts in the yellow node and should learn to exit maze (arrive at green node).

In contemporary robotics we need continuous policies. Within the traditional RL framework, an approximation of continuous policy can be achieved by increasing the number of states and actions, which substantively deceases the performance of the learning.

Considering the example in Figure 1, we can notice that in the discretization of the maze many of the states are redundant and the robot can not access them (wall cells). Following the corridor, the agent eventually arrives either at a crossing, in a dead-end or to the target. This suggests, that also the states between two crossings and between a crossing and a dead-end or the target can be left out.

Therefore, we propose to dynamically assign states rather than allocate them in advance. A suitable framework to achieve this is hierarchical reinforcement learning, where we combine RL with control algorithm as shown in Figure 2.

The upper hierarchical level is classical RL algorithm, where the states are discovered online by the lower hierarchical level. The later consists of an intelligent compliant controller, which autonomously moves within the environment constraints and detects where multiple movement options exist. The states for the upper level are only assigned when multiple options are possible.There are two main benefits of using such approach:

- The generated policies are inherently continuous.
- The number of states is greatly reduced.

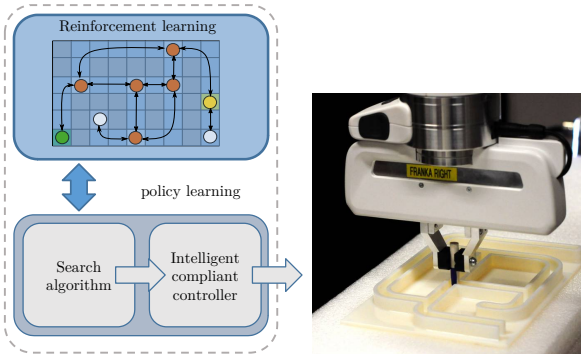

Figure 2: Block scheme of the proposed hierarchical policy learning algorithm. The upper level is RL of the policy, where the states and actions are represented with a directed graph. The lower level is an intelligent controller, consisting of a search algorithm and a Cartesian impedance controller.

The states and actions of hierarchical RL can be also represented with nodes and edges, respectively, of a directed graph as shown in the upper blue box in Figure 2.

3 INTELLIGENT COMPLIANT CONTROLLER

The lower level of the hierarchical learning utilizes a compliant control framework. As the robot moves along the boundaries, the controller searches for possible alternative movement directions.

In general, the physical constraints of the system are not known in advance. To find a feasible initial motion direction, the controller keeps applying force in random directions until this results in a movement. We then use operational space compliant controller to continue the motion in the initiated direction. The control parameters make the robot more compliant in directions orthogonal to the movement direction.

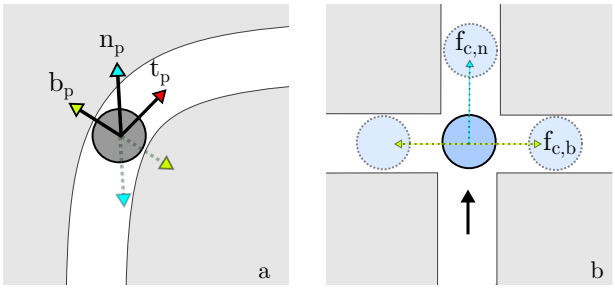

Figure 3: Searching path and possible states in restricted environment. The left part (a) shows Frenet-Serret frame attached to the end effector in the labyrinth. The right part (b) shows an instance, when the controller discovers a new state for reinforcement learning. Both parts show how search forces are applied in the normal and binormal direction.

We specify these directions using Frenet-Serret frames along the resulting motion trajectory [10] as illustrated in Figure 3 a. The Frenet-Serret frame can be expressed $\mathbf{R}_p = \begin{bmatrix} t_p & n_p & b_p \end{bmatrix}$ with

- the unit vector $\boldsymbol{t}_p = \frac{\dot{\boldsymbol{p}}}{\|\dot{\boldsymbol{p}}\|}$ tangent to the curve, pointing in the direction of motion,
- the normal unit vector $\boldsymbol{n}_p = \frac{\dot{\boldsymbol{p}} \times \ddot{\boldsymbol{p}}}{\|\dot{\boldsymbol{p}} \times \ddot{\boldsymbol{p}}\|} \times \boldsymbol{t}_p$, and
- the binormal unit vector $\boldsymbol{b}_p = \boldsymbol{n}_p \times \boldsymbol{t}_p$

where $\boldsymbol{p} \in \mathbb{R}^3$ are the measured positions of the robot end-effector.

In order to follow environmental constraints, we exploit robot's compliance. We modified a passivity-based variant of impedance control for manipulators with flexible joints [2] by allowing to set the compliance along the operational space trajectory expressed using Frenet-Serret frame rather than global frame. The task command input $\ddot{\mathcal{X}}_c = \left[\ddot{\boldsymbol{p}}_c^{\mathrm{T}}, \dot{\boldsymbol{\omega}}_c^{\mathrm{T}}\right]^{\mathrm{T}}$ is then given by:

$$\ddot{\boldsymbol{p}}_c = -\mathrm{R}_p \mathrm{D}_p \mathrm{R}_p^{\mathrm{T}} \dot{\boldsymbol{p}} + \mathrm{R}_p \mathrm{K}_p \mathrm{R}_p^{\mathrm{T}} \boldsymbol{e}_p, \quad (1)$$

$$\dot{\boldsymbol{\omega}}_c = -\mathrm{R}_o \mathrm{D}_o \mathrm{R}_o^{\mathrm{T}} \boldsymbol{\omega} + \mathrm{R}_o \mathrm{K}_o \mathrm{R}_o^{\mathrm{T}} \boldsymbol{e}_q, \quad (2)$$

where e_p and $\boldsymbol{e}_o$ are position and orientation tracking errors; $\mathbf{K}_p$ and $\mathbf{K}_o \in \mathbb{R}^{3\times3}$ are the diagonal matrices, which define the positional and rotational stiffness in the Frenet-Serret and global frames, respectively. Likewise, $\mathbf{D}_p$ and $\mathbf{D}_o \in \mathbb{R}^{3\times3}$ are diagonal damping matrices, which are set to $\mathbf{D} = 2\sqrt{\mathbf{K}}$ for critically damped system. For other parameters, please see [1].

By applying high positional gain in the direction of movement and low gains in the orthogonal direction, the robot can autonomously move along the environmental boundaries. However, following the constraints alone can not discover new states for the upper RL level. For this, small test forces are applied in the positive and negative directions of the normal and bi-normal (see Figure 3). All test forces are applied in each test position, which are placed in short intervals along the entire trajectory. If the robot moves above some predefined positional displacement threshold as a result of applying this forces in multiple directions in the same test position, the controller has found a new state (see Figure 3 b). In the new state each action corresponds to applying the specific force, which results in a movement in one of the admissible directions. The controller waits for the decision of RL algorithm, which action to take.

We assume that motion can be stopped only due to the task constraints. If the motion is interrupted, the controller searches for a new feasible motion by applying a random force in a random direction in the same manner as at the beginning. Following this strategy, the robot eventually generates a continuous policy.

4 ASSEMBLY LEARNING BY DISASSEMBLY

We can apply the same algorithm as for maze learning to disassembly operations. Key stages of disassembly and their analogies in the graph representation and hierarchical reinforcement learning are summarized in the Table 1.

A positive reward is given only when the robot has disassembled the object, i.e the target state. Negative reward is assigned when the robot arrived in a state where the motion could not be continued.

When the robot explores state s_k, the action-value function $Q(s_k, a_k)$ is updated according to the SARSA algorithm [11]:

$$Q(s_k, a_k) \leftarrow Q(s_k, a_k) + \alpha(r_k + \gamma Q(s_{k+1}, a_{k+1}) - Q(s_k, a_k)), \quad (3)$$

where s_k is the label of the k-th state, a_k is the label of the action taken in s_k, r_k is the reward obtained in state s_k, $0 < \alpha < 1$ is the learning gain and $0 < \gamma < 1$ is the discount factor, which gives recent rewards higher importance. The optimal policy can be obtained by applying ϵ-greedy strategy in the form

$$\pi(s) = \begin{cases} \arg\max_a \ Q(s,a), & \text{with probability } 1-\epsilon, \\ \text{random action}, & \text{with probability } \epsilon, \end{cases} \quad (4)$$

where parameter ϵ is the ratio between the exploration and exploitation [12].

Using the hierarchical reinforcement learning, the robot not only learns the disassembly policy, but identifies all crucial stages for the corresponding assembly process.

We assume that assembly and disassembly are mutually reversible operations, therefore we obtain initial assembly policy by merely reversing the disassembly policy. However, even if the operation is reversible small deviations in part geometry, grasping, material, etc. can result in failure. To account for this, we have to apply appropriate control together with the exception strategies, which mimic human behavior during the assembly.

We set high gains in all spatial directions until the parts to be assembled are in contact. This assures precise path tracking during the approach motion in assembly. When the parts are in contact, we use the same compliance settings as during disassembly.

During the assembly, we measure contact forces and torques and compare them with the measured forces and torques during disassembly. Note that the forces/torques during assembly have the opposite sign in relation to those measured at disassembly. If the values are still notably different, we slow down the motion and if the forces/torques are still increasing, we carry out a trajectory in the

Table 1: Key stages of disassembly and their analogies in hierarchical reinforcement learning and graph representation

Observation	Lower level	Upper level	Graph
Fully assembled product.	Controller tries to move in different directions and thereby determines admissible directions.	Start state	Yellow node
Partially disassembled product.	Controller follows the environmental constraints and moves in the only admissible direction.	Action	Edge
Multiple options to continue disassembly.	Controller tries to move in different directions and thereby determines admissible directions.	State	Orange node
Disassembly cannot be continued in the same direction.	Goes in reverse direction.	Penalty state	White node
Fully disassembled product.	Controller can freely move.	Target state	Green node

opposite direction for some time and then try again, as suggested in [6].

For improving the obtained policies many different methods exists. We apply iterative learning control, which has proven useful for on-line adaptation of force profiles in manipulation tasks [1].

5 EXPERIMENTAL VERIFICATION

We experimentally verified the proposed disassembly learning on a Franka Emika Panda robot. The control algorithm was implemented as a ros_control plugin in C++ using libfranka[3], while the learning algorithm was implemented in Matlab as a ROS node.

We verified the proposed approach using a R5W car bulb and corresponding plastic casing, used to fix the bulb above the registration plates. The R5W bulb is mounted into the plastic casing using bayonet mechanism as shown in Figure 4.

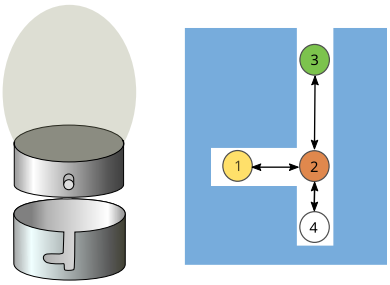

Figure 4: On the left illustration of a bayonet bulb with the corresponding casing is shown. Bayonet mechanism consist of radial pins, and a matching slot and spring to keep the two parts locked together. On the right, a projection of the slot in the casing to the plane is shown along with states than can be discovered by the controller. In disassembly task in order to release the lock, the robot first has to rotate the bulb across the horizontal part of the slot and then the pin slides into the vertical part of the slot. By lifting it upwards, the robot eventually learns to remove the bulb.

This example shows why disassembly can be easier than the assembly. In disassembly, the robot starts in state 1, and the only decision it has to make is in the state 2 to arrive in the state 3. In assembly, however, it has first to learn the proper pose of the state 3 and then search for the state 2.

The robot learns to remove the bulb from the casing as shown in Figure 5.

Applying the procedure described in Section 4, the robot successfully learns the assembly operation - bulb insertion.

6 CONCLUSIONS

Physical constrains can be used to structure and reduce search space for reinforcement learning. During the disassembly the motion of object parts is more constrained. As a consequence, learning of disassembly can be easier than learning of assembly.

Hierarchical reinforcement learning, consisting of high level decision making and intelligent compliant controller, has proven to be an efficient framework for learning in the constrained environments, such as disassembly processes. The controller exploits its

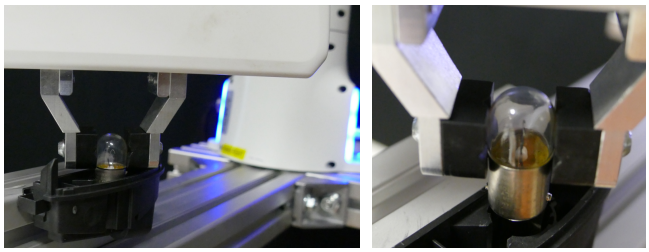

Figure 5: On the left the bulb is mounted in the casing. On the right the bulb is removed from the casing revealing its two radial pins.

compliance in order to detect admissible motions. When motion in multiple directions is possible, decisions are taken at the upper hierarchical level.

The proposed approach was experimentally validated on the case of light-bulb insertion. During the disassembly (bulb removal from the casing), all crucial stages for the corresponding assembly process (bulb insertion) can be learned autonomously and simplify the assembly learning.

Our future research will focus on evaluation of the proposed method for objects, composed of multiple parts.

ACKNOWLEDGMENTS

The research leading to these results has received funding by the EU Horizon 2020 Research and Innovation Programme under grant agreement No 820767, project CoLLaboratE.

REFERENCES

[1] Fares J. Abu-Dakka, Bojan Nemec, Jimmy A. Jørgensen, Thiusius R. Savarimuthu, Norbert Krüger, and Aleš Ude. 2015. Adaptation of manipulation skills in physical contact with the environment to reference force profiles. *Autonomous Robots* 39, 2 (2015), 199–217.

[2] A. Albu-Schaffer, C. Ott, and G. Hirzinger. 2007. A Unified Passivity-based Control Framework for Position, Torque and Impedance Control of Flexible Joint Robots. *The International Journal of Robotics Research* 26, 1 (2007), 23–39.

[3] Franka Emika. 2019. libfranka: C++ library for Franka Emika research robots. https://github.com/frankaemika/libfranka.

[4] T. Inoue, G. De Magistris, A. Munawar, T. Yokoya, and R. Tachibana. 2017. Deep reinforcement learning for high precision assembly tasks. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 819–825.

[5] J. Kober, J. a. Bagnell, and J. Peters. 2013. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research* 32, 11 (aug 2013), 1238–1274.

[6] J. S. Laursen, L.-P. Ellekilde, and U. P. Schultz. 2018. Modelling reversible execution of robotic assembly. *Robotica* 36, 5 (2018), 625–654.

[7] Sergey Levine, Nolan Wagener, and Pieter Abbeel. 2015. Learning Contact-Rich Manipulation Skills with Guided Policy Search. *International Conference on Robotics and Automation* (2015), 156–163. arXiv:1501.05611

[8] B. Nemec, L. Zlajpah, and A. Ude. 2017. Door opening by joining reinforcement learning and intelligent control. In *18th International Conference on Advanced Robotics (ICAR)*. 222–228.

[9] Jan Peters. 2010. *Policy Search for Motor Primitives in Robotics.* Technical Report.

[10] R. Ravani and A. Meghdari. 2006. Velocity distribution profile for robot arm motion using rational Frenet-Serret curves. *Informatica* 17, 1 (2006), 69–84.

[11] Gavin A Rummery and Mahesan Niranjan. 1994. *On-line Q-learning using connectionist systems.* Vol. 37. University of Cambridge, Department of Engineering Cambridge, England.

[12] Richard S. Sutton and Andrew G. Barto. 2015. *Reinforcement Learning: An Introduction, Second edition.* The MIT Press, Cambridge, London.