Robotics and Computer–Integrated Manufacturing 86 (2024) 102657

Contents lists available at ScienceDirect

Robotics and Computer-Integrated Manufacturing

journal homepage: www.elsevier.com/locate/rcim

Full length article

Hierarchical learning of robotic contact policies

Mihael Simonič [a,b], Aleš Ude [a,b], Bojan Nemec [a,*]

[a] Dept. of Automatics, Biocybernetics, and Robotics, Jožef Stefan Institute, Jamova c. 39, Ljubljana, 1000, Slovenia
[b] Faculty of Electrical Engineering, University of Ljubljana, Tržaška c. 25, Ljubljana, 1000, Slovenia

ARTICLE INFO

Keywords:
Autonomous robot learning
Learning from experience
Compliance and impedance control
Perception-action coupling
Compliant assembly

ABSTRACT

The paper addresses the issue of learning tasks where a robot maintains permanent contact with the environment. We propose a new methodology based on a hierarchical learning scheme coupled with task representation through directed graphs. These graphs are constituted of nodes and branches that correspond to the states and robotic actions, respectively. The upper level of the hierarchy essentially operates as a decision-making algorithm. It leverages reinforcement learning (RL) techniques to facilitate optimal decision-making. The actions are generated by a constraint-space following (CSF) controller that autonomously identifies feasible directions for motion. The controller generates robot motion by adjusting its stiffness in the direction defined by the Frenet–Serret frame, which is aligned with the robot path. The proposed framework was experimentally verified through a series of challenging robotic tasks such as maze learning, door opening, learning to shift the manual car gear, and learning car license plate light assembly by disassembly.

1. Introduction

Many robot tasks require tight contact with the environment. Such tasks are common in industrial environments, e.g. in assembly operations and in domestic environments, where the robot has to perform operations like opening doors and drawers to access different rooms or objects. They are generally considered hard to learn, as the robot needs to learn a policy composed of poses and wrenches while interacting with an unknown and possibly changing environment. Imitation learning is a widely used paradigm to effectively specify tasks in contact with the environment [1]. However, involving a human teacher in the learning process is not always desirable. Especially for robots operating in unstructured environments, it is often beneficial if they can learn new contact skills by themselves. This requires lengthy task adaptation procedures, which are usually realized based on reinforcement learning (RL) [2] or iterative learning control (ILC) [3].

While user-friendly programming approaches and hardware reconfigurability capabilities have long been used to enhance the capabilities of industrial applications [4,5], autonomous learning is still considered too time-consuming for such settings. Lengthy policy learning and refinement processes hinder the practical application of autonomous learning algorithms and the deployment of robots in unstructured and complex industrial environments. For applications in flexible, small-scale production, characterized by a wide variety of assembly tasks, it is very important to reduce the programming effort and the required skill level of the operator. This problem is even more pronounced when introducing robots into inherently unstructured home environments, where we cannot expect the help of experienced operators.

The aim of this paper is to introduce a new methodology that enables a robot to quickly and autonomously acquire new contact skills, even without previous imitation learning. To this end, we propose a new approach to learning robotic tasks where physical contact with the environment contributes to faster learning. The proposed approach is based on the observation that learning physically constrained tasks, can be structured more efficiently than learning tasks where a robot moves freely in space. The reason for this is that the environment constrains the admissible movement directions, thereby limiting the search space. Consequently, the number of parameters that need to be learned is significantly reduced. To implement this type of learning, we need to allow the environmental constraints to determine the robot's motion. The concept of compliant robot control provides a suitable framework for implementing such a strategy. The early stage of this concept was applied in our previous work, where we demonstrated how robots could learn tasks with physical constraints, e.g., opening doors and drawers [6], and how to learn assembly tasks by disassembly [7].

The main contribution of this paper is a comprehensive framework for autonomous learning of complex skills where the robot maintains contact with the environment. The key components of this framework are:

- Graph-based task representation: We propose a graph-based representation that enables hierarchical decomposition of complex

* Corresponding author.
E-mail addresses: mihael.simonic@ijs.si (M. Simonič), ales.ude@ijs.si (A. Ude), bojan.nemec@ijs.si (B. Nemec).

https://doi.org/10.1016/j.rcim.2023.102657
Received 12 September 2022; Received in revised form 7 June 2023; Accepted 8 September 2023
Available online 18 September 2023

contact policies. This representation allows for efficient and fully autonomous learning of these skills.

- Constraint-space following (CSF) controller: The paper introduces a CSF controller that facilitates autonomous exploration within constrained spaces. This controller relies on robust estimation of the tangential direction of motion and utilizes the Frenet–Serret (FS) formulas to specify variable compliance along the robot's motion trajectory.
- Graph topology determination algorithm: The paper presents a novel algorithm that autonomously discovers the graph topology of the task representation. This algorithm surpasses the previous approach [7] in terms of execution speed, allowing for faster learning.
- Learning the optimal sequence of movements: The framework includes RL algorithm for learning the optimal sequence of movements within the graph representation of the task. This way, the robot can efficiently perform complex contact tasks.

In summary, the paper contributes to the field of autonomous robotics by presenting a comprehensive framework that combines a graph-based task representation, an algorithm for graph topology discovery, optimal control policy learning, and a CSF controller. To the best of our knowledge, the proposed methodology is the first framework capable of entirely autonomous learning of tasks where the environment constrains the robot's motion. Moreover, the learning speed is comparable to that of humans. The validity of the proposed approach has been demonstrated by learning four challenging tasks taken from everyday life and industrial environments.

This paper consists of five sections. In Section 2, we briefly review existing approaches to learning contact policies. Our main contribution is a new scheme for learning such policies based on a Hierarchical Reinforcement Learning (HRL), which is presented in Section 3. The experimental evaluation presented in Section 4 considers four challenging tasks: maze learning, door opening, learning to shift a manual car gearbox, and learning to assemble a car license plate by disassembly. The discussion and final conclusions are provided in Section 5.

2. Related work

A lot of research on learning tasks that involve contact-rich manipulation in unstructured environments has been performed in the past. A comprehensive survey paper on robots performing manipulation tasks that require varying contact with the environment has been published recently [8]. This section focuses on learning-based approaches [9].

The most basic and straightforward approaches are based on the Iterative Learning Control paradigm (ILC), which has been used to adapt the trajectory of tasks involving contact with the environment [10,11]. These approaches minimize force tracking errors but cannot improve the desired force–torque contact profile. More general are the approaches based on reinforcement learning (RL), which has been successfully applied for learning tasks such as door opening and picking a pen from the table [12]. However, this type of learning requires approximately a hundred trials to learn the policy and is slow compared to humans. Learning of contact-rich tasks is closely related to the learning of impedance parameters, as proven in [13], where the authors addressed deep learning in action space. Imitation can increase learning speed, as demonstrated in the wood planing experiment [14], where RL enhanced the skill transfer from humans to robots. In [15], the authors proposed Guided Policy Search to handle contact-rich tasks. Initialized by demonstration and optimized by PI^2 RL algorithm, the local policies were used to generate a global policy using deep neural networks (DNN) and robot joint torques directly from the visual input. A DNN was also applied for learning a peg-in-hole (PiH) task without prior demonstration [16], where a robot learned the desired search and insertion policy using the deep Q-learning algorithm. After 100–200 trials, the robot learned a robust search and insertion policy. To increase the data efficiency and learning speed, a hierarchical RL scheme was proposed and applied to a dual PiH task [17]. PiH was also addressed in [18], where the authors applied movement primitives encoded as a neural network and a deep deterministic policy gradient algorithm for learning neural network parameters. The learning efficiency was also pursued in [19], where the authors proposed learning meta-parameters to encode a specific skill in combination with an adaptive impedance controller. Another recent approach to autonomous learning of contact tasks is based on observing time-reversed visual cues, enabling one to learn the policy without previous demonstration or exploration [20]. Very similar objectives were also pursued in the recent study [21], where the aim was to learn a shape descriptor that establishes geometric correspondences between object surfaces and their target locations directly from the visual stream.

In this paper, we propose a new approach to learning contact-rich tasks based on a hierarchical learning scheme. In this respect, our approach is related to the approach described in [17], except that in our case, the highest hierarchical level is decision learning, and the lowest level includes a variable compliance controller to move along the physical constraints of the environment. It is also partly related to the approaches proposed in [20,21], as we also learn the reverse policies of assembly tasks [7]. There are also some similarities between our research and [19], where they also used a directed graph to represent the learned policy. However, there is a significant difference between the two approaches in the purpose and construction of the graph.

One of the main advantages of our approach is its high learning speed, which exceeds the performance of reinforcement and deep reinforcement learning methods that do not exploit the constraints of contact tasks. The high learning speed is achieved by a meaningful decomposition of the task into the determination of environmental constraints and the decision level, which is implemented by a hierarchical learning scheme. In addition, the proposed graph-based task representation provides a better insight into the nature of the problem. The proposed algorithm does not require prior knowledge of this graph but builds it autonomously through exploration.

3. Learning of contact policies with hierarchical learning

Our approach to learning contact tasks exploits the configurations that allow only partial freedom of motion, both in terms of position and orientation. The boundary between the region where the robot's motion is constrained by the environment and the region where the motion is free is called a *C-surface* [22]. Motion is possible along the tangential directions of the C-surface and constrained along the orthogonal directions. The dimension of the C-surface determines the number of degrees of freedom of a physically feasible robot motion. Our research considers tasks with one dimensional C-surface, which is typical for assembly tasks. We further assume that the task consists of different motion primitives that can be combined to form a longer action sequence. Each motion primitive is associated with a one-dimensional C-surface. The sequence of motion primitives can be such that we have more choices on how to proceed with the task. For example, consider the shifting of car gears, as discussed in Section 4.3. When moving the gear lever from neutral to the left, we have two options: turn up to the first gear or turn down to the second gear. Such a sequence with branches can be represented with a graph, where nodes represent various key states of the task and edges represent motion primitives constrained by the environment [7].

An example of such a graph is shown in Fig. 1. The nodes represent the key states and the edges represent the robot movements between the key states of the task. The start node (colored yellow) can be anywhere in the graph and denotes the initial state of the task. The node is characterized by how many branches start from each node, called the degree of the node. In a node of degree 2 or more (colored orange), the robot must decide how to proceed. In a node of degree 1 (colored blue), we do not have multiple options of how to continue the task, but

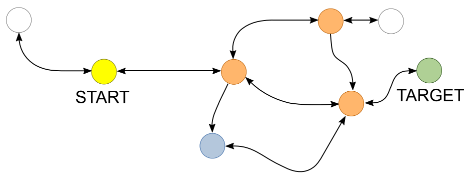


Fig. 1. The proposed graph-based representation of contact tasks in a general form.

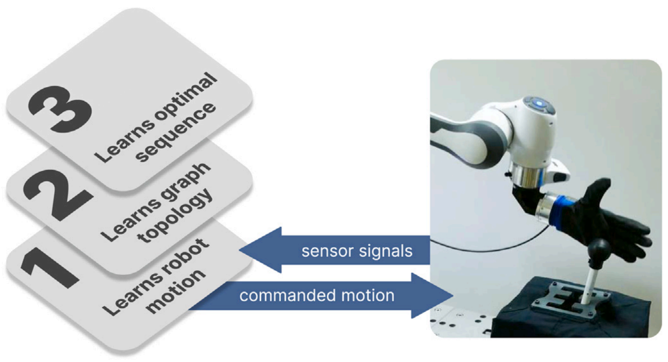


Fig. 2. The proposed hierarchical scheme for learning contact policies represented with graphs.

to continue we must choose a completely different direction or type of motion; for example, a translational motion changes to a rotational or vice versa. A node of degree zero (colored white) represents stages of the task where the motion can only be continued by turning back. The goal is to reach the target node (colored green).

Graphs are often used to represent sequences of assembly and disassembly motions [23–25]. In general, a graph, possibly directed, is a conceptual representation of a sequence of activities. In our approach, a graph represents a single, optimal policy for the execution of a contact task. By joining multiple such graphs, more complex tasks that consist of several contact tasks can be represented (as shown in Fig. 12, where a human demonstrated the necessary movement between two contact tasks).

Let us now assume that we do not know how to execute a contact task in advance, but we do know the starting point of the task and what the target state is. Consequently, we do not know the topology of the graph, the intermediate nodes, and the actions for the transition between individual nodes. Thus the robot should learn this through autonomous exploration. Furthermore, it has to learn the optimal policy from the start node to the target node, i.e. the optimal sequence of movement primitives to accomplish the given task.

3.1. *Hierarchical reinforcement learning*

The complexity of learning problems can sometimes be reduced by hierarchical learning schemes, which split the learning problem into sub-tasks with multiple levels of hierarchy. In this section, we explain our hierarchical scheme (see Fig. 2) on the example of maze learning, where the goal is to move from the start to the target node (see Fig. 5 and Fig. 6). Note that the maze constrains the possible robot motion. It is natural to use a graph representation to represent the points in the maze where there are multiple directions in which the robot can continue its motion or where the robot can only continue its motion by turning back.

The graph representation of the task is not known in advance but must be learned. We initialize the learning process with a graph that has only one node, i.e. the start node. Each node is associated with the robot pose in this state (and, in some cases, several preceding poses). The initial node is associated with the robot pose at the beginning of the task execution. Using the proposed approach, the robot autonomously explores the environment along its constraints (lowest level) and identifies new states (nodes) and actions (edges) that cause a transition between the states (middle level). The actions are represented by the motion trajectories that specify the robot motion between the connected nodes. The reinforcement learning algorithm at the top of the hierarchical scheme learns the optimal movement sequence from the start to the target node.

On the shortcomings of traditional shortest path searching algorithms. The design of the search algorithm at the highest hierarchical level depends on the complexity of the problem. In less complex cases, it is sufficient to find the shortest path from the start node to the target node. In general, however, this is not sufficient; the robot must learn that it may be necessary to visit a specific node in the graph to reach the target node successfully or even that it may have to pass through a certain node more than once. Consider, for example, the previously mentioned gear shifting case, addressed in detail in Section 4.3. The robot must not shift from the first gear directly to the fifth, which would be the shortest path in the graph, but must learn to shift continuously from a lower gear to a higher gear. To address such tasks, we chose reinforcement learning (RL) to guide the search at the highest hierarchical level. In some cases, however, it makes sense to use computationally more efficient graph search algorithms [26].

3.2. *Lowest level: Constraint-space following controller*

The detailed presentation of our hierarchical scheme starts from the bottom up by introducing the Constraint-Space Following (CSF) controller, which is used to move the robot's end-effector along the environmental constraints. At the lowest level, the controller is given a direction (selected by the algorithm at the highest level described in Section 3.4) in which the robot can start exploring the environment from the current node (state). Here the focus is on how to control the robot motion in the selected direction while exploring whether motion in any other direction is possible, which indicates the existence of the next node in the graph.

For this purpose, we developed an impedance controller that is stiff in the current direction of movement and compliant in the orthogonal directions. A similar approach was proposed in [22,27], where they introduced the concept of *Compliant frame* and *Task frame*. In our framework, we formalize this motion control by utilizing the *Frenet–Serret (FS) frame* [28,29], which is attached to the robot tool center point (TCP). A sequence of FS frames is illustrated in Fig. 3. An FS frame at position $\boldsymbol{t}_p$ is defined by a rotation matrix $\mathbf{R}_p$ with the first column aligned with the tangential direction of motion, i.e. $\dot{\boldsymbol{p}}$, and the other two columns orthogonal to it. They are referred to as normal and binormal vectors. $\mathbf{R}_p$ and the corresponding coordinate axes can be computed as follows

$$\mathbf{R}_p = \begin{bmatrix} \boldsymbol{t}_p & \boldsymbol{n}_p & \boldsymbol{b}_p \end{bmatrix}, \tag{1}$$

$$\boldsymbol{t}_p = \frac{\dot{\boldsymbol{p}}}{\|\dot{\boldsymbol{p}}\|}, \quad \boldsymbol{b}_p = \frac{\dot{\boldsymbol{p}} \times \ddot{\boldsymbol{p}}}{\|\dot{\boldsymbol{p}} \times \ddot{\boldsymbol{p}}\|}, \quad \boldsymbol{n}_p = \boldsymbol{b}_p \times \boldsymbol{t}_p,$$

where $\boldsymbol{p} \in \mathbb{R}^3$ are the measured robot TCP positions. For the rotational part of motion, the FS frame is defined as

$$\mathbf{R}_o = \begin{bmatrix} \boldsymbol{t}_o & \boldsymbol{n}_o & \boldsymbol{b}_o \end{bmatrix}, \tag{2}$$

$$\boldsymbol{t}_o = \frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|}, \quad \boldsymbol{b}_o = \frac{\boldsymbol{\omega} \times \dot{\boldsymbol{\omega}}}{\|\boldsymbol{\omega} \times \dot{\boldsymbol{\omega}}\|}, \quad \boldsymbol{n}_o = \boldsymbol{b}_o \times \boldsymbol{t}_o,$$

where $\boldsymbol{\omega} = 2\bar{\boldsymbol{q}} * \dot{\boldsymbol{q}}$ is the angular velocity. Here $\bar{\boldsymbol{q}}$ denotes the conjugate quaternion and we exploit the fact that the product of a conjugate of unit quaternion with its own derivative results in a quaternion with zero scalar part, which can be interpreted as a vector.

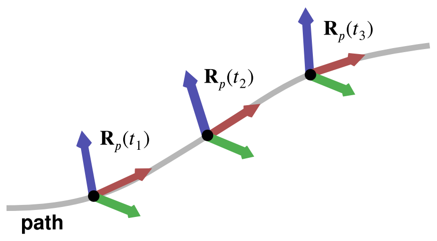


Fig. 3. Frenet–Serret frames $\mathbf{R}_p(t_k)$ for the positional path. The tangential direction is in red, the normal in green, and the binormal in blue.

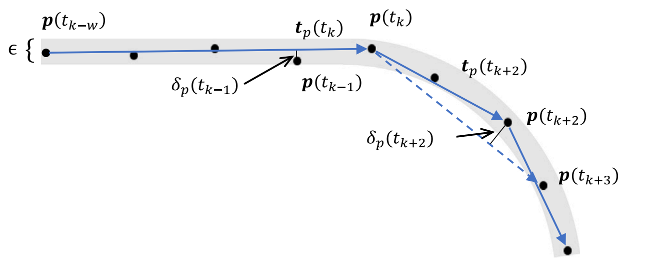

$\boldsymbol{p}(t_{k-w})$
ϵ
$\boldsymbol{t}_p(t_k)$
$\boldsymbol{p}(t_k)$
$\delta_p(t_{k-1})$
$\boldsymbol{p}(t_{k-1})$
$\boldsymbol{t}_p(t_{k+2})$
$\boldsymbol{p}(t_{k+2})$
$\delta_p(t_{k+2})$
$\boldsymbol{p}(t_{k+3})$

Fig. 4. Tangent estimation with the proposed variable rate filter. Solid dots represent the measured robot positions. The dotted line shows the candidate tangent at time t_{k+2} that was discarded because it does not meet the condition $\delta(t_{k+2}) \leq \epsilon_p$.

The above equations require accelerations, which are usually low and therefore very noisy during operations like assembly and disassembly. Different measures were proposed to provide for a robust FS frame estimation, such as using Kalman filtering [30], Bishop frames [31], and an optimization-based calculation [32]. All these methods require knowledge of the entire trajectory before processing and thus cannot be used for a real-time FS frame calculation, as it is required by our approach. On the other hand, our policy learning algorithm requires a controller stiff in the tangential direction of motion and compliant in the normal and binormal direction of the corresponding FS frame. In most practical cases it is best to assume equal compliance in the normal and binormal direction. In such cases (see the lemma in the Appendix), the robot control torques are independent of the direction of the normal and binormal vector. Hence, given the tangential direction, the other two columns of the coordinate frame can be selected arbitrarily in the plane orthogonal to the tangential direction of motion. In our approach, we take the second and third columns of the coordinate frame from the previous sample and then apply a Gram–Schmidt orthogonalization procedure to obtain an orthonormal basis for the current coordinate frame. We call the resulting coordinate frames *modified FS frames*. The initial modified FS frame is computed with an arbitrary selection of vectors $\boldsymbol{n}$ and $\boldsymbol{b}$ that are further orthogonalized.

The remaining concern is how to estimate the tangential direction of motion in a robust way. Niemeyer and Slotine [33] proposed a spatial filter, which does not affect the normalization of the velocity vector. The spatial filter extended for the rotational movement was used for the tangent estimation in our previous research [6,7]. However, since the spatial filter is a first-order filter, it introduced a lag. In our experiments, it turned out that the main problem for tangent estimation is not the sensor noise but the compliance of the tool and the tolerances of the environment, which allow small motions in other directions, although the robot's motion is primarily constrained to only one direction. Therefore we applied a filter with a variable rate, inspired by the approach proposed in [34]. For the positional part of the motion, the tangent is estimated using

$$\boldsymbol{t}_p(t_k) = \begin{cases} \dfrac{\boldsymbol{p}(t_k) - \boldsymbol{p}(t_{k-w})}{\|\boldsymbol{p}(t_k) - \boldsymbol{p}(t_{k-w})\|}, & \delta_p(t_{k-1}) \leq \epsilon_p \\ \dfrac{\boldsymbol{p}(t_k) - \boldsymbol{p}(t_{k-1})}{\|\boldsymbol{p}(t_k) - \boldsymbol{p}(t_{k-1})\|}, & \text{otherwise} \end{cases}, \tag{3}$$

where t_k is the current time, w a suitably chosen delay factor that determines the smoothing of the digital filter, $\delta_p(t_{k-1})$ is the distance between the line $[\boldsymbol{p}(t_k), \boldsymbol{p}(t_{k-w})]$ and position $\boldsymbol{p}(t_{k-1})$, and ϵ_p a constant that determines the switching between the filtered and non-filtered estimation of the tangent. To prevent filter chattering, a small hysteresis is usually applied to ϵ_p. The filtering of sampled points $\boldsymbol{p}(t_k)$ is illustrated in Fig. 4. For a rotational motion represented by a quaternion trajectory, the filter takes the form

$$\boldsymbol{t}_o(t_k) = \begin{cases} \dfrac{2\log(\boldsymbol{q}(t_k) * \bar{\boldsymbol{q}}(t_{k-w}))}{\|2\log(\boldsymbol{q}(t_k) * \bar{\boldsymbol{q}}(t_{k-w}))\|}, & \delta_o(t_{k-1}) \leq \epsilon_q \\ \dfrac{2\log(\boldsymbol{q}(t_k) * \bar{\boldsymbol{q}}(t_{k-1}))}{\|2\log(\boldsymbol{q}(t_k) * \bar{\boldsymbol{q}}(t_{k-1}))\|}, & \text{otherwise} \end{cases}, \tag{4}$$

where log is the quaternion logarithm.

Given the FS frame, we need a control law that enables the application of arbitrary compliance along the FS frame axes. Our approach is based on the passivity-based variant of impedance control for manipulators with flexible joints [35]. We implemented a modification to set the compliance along the Cartesian space trajectory with FS frames attached. The commanded torque $\boldsymbol{\rho}_c \in \mathbb{R}^N$, which is passed to the robot motors, is calculated as

$$\boldsymbol{\rho}_c = \mathbf{B}\mathbf{B}_\Theta^{-1}\boldsymbol{u} + (\mathbf{I} - \mathbf{B}\mathbf{B}_\Theta^{-1})\boldsymbol{\rho}, \tag{5}$$

$$\boldsymbol{u} = \mathbf{J}^{\mathrm{T}}(\boldsymbol{\theta})\left(\begin{bmatrix}\boldsymbol{f}_c\\ \boldsymbol{m}_c\end{bmatrix} + \begin{bmatrix}\boldsymbol{f}_s\\ \boldsymbol{m}_s\end{bmatrix}\right) + \mathbf{g}(\boldsymbol{\theta}) + \mathbf{N}(\boldsymbol{\theta})\dot{\boldsymbol{\rho}}_0, \tag{6}$$

where N is the number of robot joints, $\boldsymbol{\theta} \in \mathbb{R}^N$ is the vector of joint angles computed from the motor angles $\boldsymbol{\Theta} \in \mathbb{R}^N$ [35], $\mathbf{J} \in \mathbb{R}^{N\times 6}$ is the manipulator Jacobian, while $\mathbf{B}$, $\mathbf{B}_\Theta \in \mathbb{R}^{6\times 6}$ denote the positive definite diagonal matrices of the actual and the desired joint inertia, respectively. The aim of the term $\mathbf{B}\mathbf{B}_\Theta^{-1}$ is to reduce the joint inertia. $\boldsymbol{\rho}$ is the vector consisting of the measured joint torques and $\mathbf{g}(\boldsymbol{\theta})$ is the gravity vector [36]. To control the configuration of the robot, a nullspace term is added [37], where $\mathbf{N}(\boldsymbol{\theta}) = \mathbf{I} - \mathbf{J}^{\mathrm{T}}(\boldsymbol{\theta})\mathbf{J}^{+T}(\boldsymbol{\theta}) \in \mathbb{R}^{n\times n}$ is the null space projection operator, $\mathbf{J}^{+}(\boldsymbol{\theta})$ denotes Moore–Penrose pseudo-inverse of the Jacobian and $\dot{\boldsymbol{\rho}}_0 \in \mathbb{R}^n$ is the null space joint torque vector. $\boldsymbol{f}_s$ and $\boldsymbol{m}_s$ are additional forces and torque vectors in task coordinates, which are used for searching for a feasible motion direction. The selection of the probing forces $\boldsymbol{f}_s$ and torques $\boldsymbol{m}_s$ depends on a task. It is discussed in Section 3.3 for planar maze learning and in Section 4 for other tasks. The motor torque controller (5) reduces the motor inertia and compensates for the robot's non-linear dynamics, while Eq. (6) provides for the desired impedance and damping, additional task force, gravity compensation, and null space motion. The task command input $[\boldsymbol{f}_c^{\mathrm{T}}, \boldsymbol{m}_c^{\mathrm{T}}]^{\mathrm{T}}$ is computed as

$$\boldsymbol{f}_c = -\mathbf{R}_p\mathbf{D}_p\mathbf{R}_p^{\mathrm{T}}\dot{\boldsymbol{p}} + \mathbf{R}_p\mathbf{K}_p\mathbf{R}_p^{\mathrm{T}}\boldsymbol{e}_p, \tag{7}$$

$$\boldsymbol{m}_c = -\mathbf{R}_o\mathbf{D}_o\mathbf{R}_o^{\mathrm{T}}\boldsymbol{\omega} + \mathbf{R}_o\mathbf{K}_o\mathbf{R}_o^{\mathrm{T}}\boldsymbol{e}_q, \tag{8}$$

where position and orientation tracking errors are defined as $\boldsymbol{e}_p = \boldsymbol{p}_d - \boldsymbol{p}$ and $\boldsymbol{e}_o = 2\log(\bar{\boldsymbol{q}} * \boldsymbol{q}_d)$. $\mathbf{K}_p$ and $\mathbf{K}_o \in \mathbb{R}^{3\times 3}$ are the diagonal matrices defining the positional and rotational stiffness along and around coordinate axes, respectively. $\mathbf{D}_p$ and $\mathbf{D}_o \in \mathbb{R}^{3\times 3}$ are diagonal damping matrices, which are set as diagonal elements of the block diagonal matrix[1]

$$\mathbf{D} = 2\sqrt{\mathbf{B}_\Theta + \begin{bmatrix}\mathbf{K}_p & 0\\ 0 & \mathbf{K}_o\end{bmatrix}}\,. \tag{9}$$

With the proposed approach, the robot is able to move along the environmental boundaries while probing for possible movements in

[1] In [35] the authors proposed double-diagonalization method to shape the damping. However, the resulting damping matrix is not diagonal, as our approach requires. Our experiments showed that the performance degradation due to diagonal damping matrices was negligible.

other directions, given that we apply high stiffness in the direction of motion and low gains in the orthogonal directions.

3.3. Middle level: Graph topology determination

The CSF controller described above enables the robot to move along the environmental constraints while applying probing wrenches that indicate the possibility of moving in directions other than the commanded tangential direction. The next task in our hierarchical learning scheme is to determine when the robot has arrived to the next node in the graph. When the robot arrives to the next node for the first time, the newly identified node should be added to the graph. In this section, we first define the representation of nodes and edges, outline the criteria for node detection, and present the algorithm that systematically explores the graph topology.

3.3.1. Representation of nodes and edges

Each node in the graph is characterized by the current robot pose and possibly some previous poses that the robot has reached in the nodes immediately preceding the current node. Such a sequence of poses defines a state s_k associated with the k-th node of the graph:

$$s_k = \{\boldsymbol{p}_k, \boldsymbol{q}_k, \boldsymbol{p}_{k-1}, \boldsymbol{q}_{k-1}, \dots, \boldsymbol{p}_{k-\kappa}, \boldsymbol{q}_{k-\kappa}\}, \tag{10}$$

where $\boldsymbol{p}_k \in \mathbf{R}^3$ is the position and $\boldsymbol{q}_k \in \mathbb{R}^4$ a unit quaternion representing the orientation. $0 \leq \kappa < k$, i.e. the number of poses in s_k, is chosen so that s_k satisfies the Markov decision property, i.e. each transition is determined only by the current node and the action selected in that node. To ensure the Markov decision property, it is sometimes necessary to include previous poses in the state description, i.e. $\kappa \geq 1$. This effectively means that some parts of the graph are explored again, since this results in the creation of new states (see Fig. 8). This way it becomes possible to solve problems such as opening locked doors, where some states only become reachable after a certain other state has been visited. In practice, the parameter κ is chosen manually by the user. In most of the experiments described in Section 4, κ was set to 0.

The transition from one node to another defines an edge in the graph. In the proposed system, a robot motion trajectory starting in one and ending in another node is encoded with a speed-scaled Cartesian space dynamic movement primitive (CDMP), which handles Cartesian space policies [38] and non-uniform velocity scaling [39]. The benefit of such encoding is twofold: it allows a compact, smooth, and scalable representation of the learned policy, and it removes the explicit time dependence of the trajectory. This enables the robot to slow down or speed up the execution of the learned assembly policy if needed [39]. The CDMPs starting in k-th node are stored in a set $\mathcal{A}(s_k) = \{a_k^i\}_{i=0}^d$, where d is the degree of the node. The i-th CDMP in state s_k is represented as a_k^i:

$$a_k^i = \{\boldsymbol{w}_k^i, \boldsymbol{g}_k^i, \tau_k^i\}, \tag{11}$$

where $\boldsymbol{w}_i$ are the CDMP weights, $\boldsymbol{g}_i$ is the CDMP goal, and τ_i is the temporal scaling factor of the CDMP. See [38] for more details.

3.3.2. Node discovery by examining each branch with different probing wrenches

We explain our search algorithm on the example of the planar maze learning problem, where there are at most two possible directions to continue at each node, other than turning back (see the graphs in Fig. 5). In the graph representation of such a maze, the maximum node degree is 2.

In our previous paper [7] we proposed an algorithm that discovers new graph nodes by probing for the possible crossings that indicate the nodes in small steps. However, this approach is slow because the robot must constantly stop to make probes with search wrenches in different directions. In the approach proposed in this paper, the maze is explored using a method inspired by the wall following algorithm [40]. In this approach, each possible motion (branch in the graph) is examined twice by adding a constant probing force in either the positive or negative direction of the normal.[2] Next, we compare the robot paths executed during the two passes. If the path splits, we have found a new node of degree 2. If there is no split but the robot cannot continue in the same tangential direction, we have found the node of either degree 0 (where the robot can only turn back) or degree 1 (where the robot can continue only by changing the type of motion). The robot continues until it reaches the target node. To prevent the robot from entering a loop, we check that the robot has not already visited the same segment with the same probing force.

As explained above, the example shown in Fig. 5 treats maze learning as a planar problem. Therefore we only applied probing forces along the normal axis of the FS frame. An extension of the proposed algorithm to the 3D case, rotational motion, or nodes of a higher degree is straightforward. Depending on the problem, probing wrenches in all relevant directions must be given as input to the node discovery algorithm. For each additional probing wrench, additional searches from the currently visited node are necessary, which increases the search time. In a practical implementation, it is, therefore, advisable to consider only those search directions that are really necessary. For most problems, these can be defined by a user in advance. We provide some examples in Section 4.

3.3.3. Exploring the entire graph

The approach described above explores the selected branch with all possible probing forces. To ensure that the entire graph is explored, it is necessary to examine each node s_k and examine all possible directions d_k^i.

Initially, the problem description contains only the start node defined by the initial robot pose. The motion can be initiated by moving the robot in one of the four orthogonal directions aligned with the coordinates of the maze.[3] In the proposed system, this is characterized as $\mathcal{D}(s_k) = \{d_k^i\}_{i=0}^n$, where d_k^i denotes the possible directions to continue the motion and n is the maximum number of them. In the case of the maze learning $n = 3$. In all our experiments, the initial set of search directions was the same for all nodes.

Starting from the first node, we search in all possible directions with all probing wrenches $[\mathbf{f}_s^{\mathrm{T}}, \mathbf{m}_s^{\mathrm{T}}]^{\mathrm{T}}$ to find other nodes in the graph. If the robot cannot move in the given direction, d_k^i is removed from the set $\mathcal{D}_k(s_k)$. The process repeats until all nodes have been fully examined.

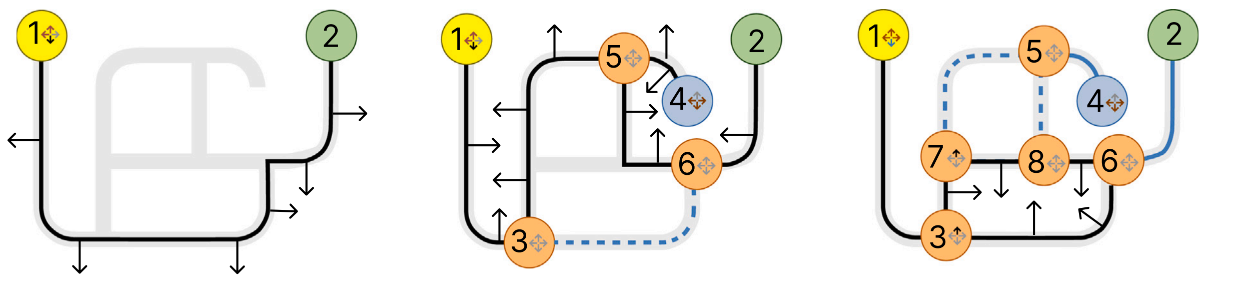
At this stage, we can uniquely construct all edges a_k^i in the graph using the trajectories stored during the search process. The developed procedure for the discovery of new nodes and edges is summarized in Algorithm 1.

3.4. Highest level: Reinforcement learning

The algorithm at the highest level learns the optimal sequence of movements from the start to the target node. Nodes and edges in the graph-based representation of the task are closely related to the states and actions in RL algorithms (see Section 3.3.1). In RL, an action causes a transition from one state to another. In our graph-based task representation, this corresponds to following the CDMP associated with the chosen edge from the current node to the next node.

[2] In the maze exploration literature, this type of search procedure is referred to as the left-hand and right-hand rule.

[3] As the robot is compliant, we can apply a suitable wrench at the robot's tip so that the robot attempts to move in the given direction. If the applied motor command results in motion, we can continue the motion using the approach described in Section 3.2. Note that due to robot compliance, the direction of the applied force or torque does not need to be perfectly aligned with the direction of the free motion.



(a) Search direction 3 and positive probing force from node 1 (b) Search direction 3 and negative probing force from node 1 (c) Search direction 1 and negative probing force from node 3

Fig. 5. Key steps in the search process to explore a planar maze. (a) The robot starts in node 1 and selects a direction to explore (down) after trying two other directions before. It applies a positive force and continues until it reaches the target node (marked green). The route is marked with a black line. (b) The robot examines the same branch again with a negative force. Doing so the robot must also turn back and discovers node 4. As the path diverges from the previous (blue dashed line), the algorithm discovers new nodes 3, 5 and 6. The edges between nodes 1 and 3, 4 and 5 and 6 and 2 have been explored with both probing forces and do not need to be examined anymore. (d) The robot continues with the closest node with unexamined branches. It goes up in node 3 and applies a positive force, which leads to the discovery of further nodes and edges. The arrow emblems indicate continuation directions (gray for unexamined, black for selected, blue for examined, and red for impossible).

Algorithm 1: Graph topology determination algorithm for $\kappa = 0$. The algorithm identifies all nodes and edges in the graph by making sure that at every node, the robot searches in all possible continuation directions with all probing wrenches. The algorithm terminates when all nodes are fully examined.

```
Input: Initial robot pose, probing wrenches $[\mathbf{f}_s^{\mathrm{T}}, \mathbf{m}_s^{\mathrm{T}}]^{\mathrm{T}}$, search directions set $\{d_i\}_{i=0}^n$
Output: Graph representation of the problem
1  initialize the graph with start node $s_1$ at the initial robot pose, $K = 1$
2  assign the possible search directions $\mathcal{D}(s_1) = \{d_i\}_{i=0}^n$
3  while $\exists k \le K$ so that $d_i \in \mathcal{D}(s_k)$ and not all probing wrenches have been applied yet in direction $d_i$ do
4      select node $s_k$ closest to the start node that fulfills the above condition
5      select a not yet fully examined direction $d_i \in \mathcal{D}(s_k)$
6      select probing wrench $[\mathbf{f}_s^{\mathrm{T}}, \mathbf{m}_s^{\mathrm{T}}]^{\mathrm{T}}$ that has not yet been applied while examining the direction $d_i$
7      if motion in the selected direction possible then
8          while not in target node and not in loop do
9              apply probing wrench $[\mathbf{f}_s^{\mathrm{T}}, \mathbf{m}_s^{\mathrm{T}}]^{\mathrm{T}}$ while following the environmental constraints (see Section 3.2)
10             record trajectory
11             if not reached existing node then
12                 if motion cannot be continued in any other way but turning back then
13                     $K = K + 1$, create node $s_K$ at the current robot pose, $\mathcal{D}(s_K) = \{d_i\}_{i=0}^n$
14                 if motion diverged from a path explored in one of the previous iterations or rejoins an existing path then
15                     $K = K + 1$, create node $s_K$ at the current robot pose, $\mathcal{D}(s_K) = \{d_i\}_{i=0}^n$
16         mark $d_i \in \mathcal{D}(s_k)$ as examined with the selected probing wrench
17     else
18         remove $d_i$ from $\mathcal{D}(s_k)$
19 identify the graph edges associated with search directions $d_i \in \mathcal{D}(s_k)$ by analyzing the stored search trajectories
20 encode trajectories associated with the identified edges as CDMP $a_k^i$
```

In our approach, the selection of the next action to be explored is guided by ϵ-greedy strategy, which is defined as follows

$$\pi(s) = \begin{cases} \underset{a}{\operatorname{argmax}}\ Q(s,a), & \text{with probability } 1-\epsilon \\ \text{random action}, & \text{with probability } \epsilon, \end{cases} \tag{12}$$

where parameter ϵ is the ratio between the exploration and exploitation [41].

Since finding optimal paths in a graph is a finite horizon discrete problem [42], we can apply any classical RL algorithm. In our experiments, we used off-policy Q-learning [41]. We assign a positive reward when the robot reaches the goal state. No intermediate rewards are assigned before reaching the goal state. In every state s_k, the action-value function $Q(s_k, a_k)$ is updated according to the Q-learning algorithm

$$Q(s_k, a_k) \leftarrow Q(s_k, a_k) + \alpha(r_k + \gamma \max\{Q(s_{k+1}, a_{k+1})\} - Q(s_k, a_k)), \tag{13}$$

where s_k is the k-th state, a_k denotes 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 algorithm at the highest level returns a walk from the start node to the end node, which defines the learned sequence of CDMPs.

The learned CDMPs can be further improved with various robot policy refinement methods such as ILC [10].

4. Experimental evaluation

This section experimentally verifies the proposed hierarchical learning of contact policies. All experiments were performed with a seven-degrees-of-freedom collaborative robot Franka Emika Panda. For this purpose, we implemented the CSF controller using the libfranka library and the ros_control framework in C++. The highest hierarchical level with the Q-learning RL algorithm and the middle level with the graph topology detection algorithm were implemented in Matlab and communicated with the CSF controller using ROS at 100 Hz. The Q learning and CSF controller parameters were the same for all experiments. They were set to $\alpha = 0.9$, $\gamma = 0.92$ and $\epsilon = 0.8$. The delay factor w used in the filtering was chosen by trial and error to be $w = 5$. The reward assignment was also the same in all cases. When the robot reached the desired state, we assigned a reward of $r = 20/k$, where k is the number of steps in Eq. (13) to reach the desired state. The gain of the CSF controller was set to 2000 N/m and 60 N/rd in the tangential component and to zero in the normal and binormal of the FS frame. The probing force and torque magnitudes f_0 and m_0 were set to 7 N and 0.1 N m, respectively. An anthropomorphic qb SoftHand [43] was used in the door opening and gear shifting experiments. In the remaining experiments, the original Panda's two-finger gripper was used.

4.1. Maze learning

In the first experiment, we applied our approach to the maze learning problem (see Fig. 6). We already studied maze learning in our

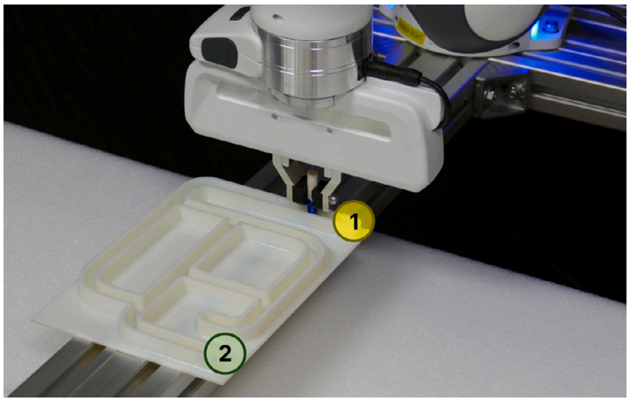


Fig. 6. Panda robot during the maze learning process. The start and the goal states are shown in yellow and green, respectively.

previous research [7], where we demonstrated the greatly improved efficiency of hierarchical learning compared to a single-level RL scheme. In this research, we additionally improved the efficiency of learning, because the proposed approach does not need to stop in small steps to check for the existence of new nodes (see Section 3.3). To initiate the graph search, we manually guided the robot to the start node, denoted with 1 in Fig. 6. This was the starting point in all subsequent search iterations. The normal direction of the modified FS frame was aligned with the global z axis, which is orthogonal to the maze. As the robot motion during maze exploration is planar, the tangent to the robot motion is guaranteed to lie in the plane defined by the maze. The third orthogonal axis (binormal) can thus be easily computed as the vector product of the tangent and the z axis.

Once all branches are investigated, the algorithm cannot find new nodes. The key stages of the search procedure are explained in Fig. 5. The algorithm at the highest level has all the necessary information and can continue learning without the robot having to perform the movements in the maze. In the case of our maze, the CSF controller at the lowest and the search algorithm at the middle level needed three iterations to find all eight nodes, and two more to ensure no further nodes exist. The Q-learning needed five or fewer episodes to discover an optimal sequence of nodes on average. The convergence of the Q-learning is shown in Fig. 7 left.

The selection of the algorithm for discovering new nodes does not affect the learned policy. However, it affects how fast the robot moves along the corridor during learning. The total learning time with the proposed search strategy was approx. ten times shorter than in our previous experiment [7]. This difference is not due to the greater or lesser efficiency of the individual algorithm but rather due to the robot movements generated by the individual algorithm. In the previous algorithm, the robot advances in small steps as it searches for path forks. In contrast, the new algorithm generates continuous paths where branch points are sought by applying a force perpendicular to the robot's direction of motion. The video of the experiment is accessible in supplementary materials as Maze Learning video.

We repeated the learning algorithm for a more complex case where the maze exit is locked. To unlock it, the robot must first visit state four. The Q-learning convergence for this example is shown in Fig. 7 right. Note also that an algorithm for finding the shortest path in graphs at the highest hierarchical level could not solve this problem. We considered this problem as many mechanisms require a specific sequence of states to be visited before reaching the desired state. An example from everyday life is a retractable ballpoint pen. Another such example, door opening, where the doors are equipped with a multi-point locking mechanism, is discussed next.

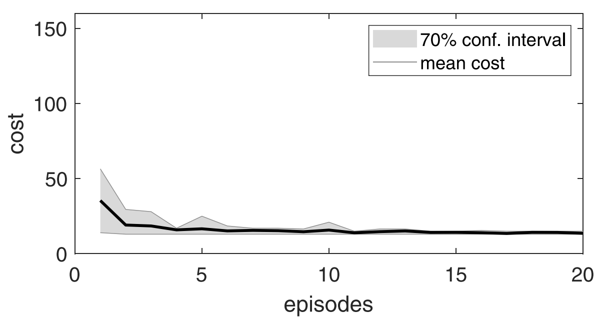
4.2. Door opening learning

In order to demonstrate the limitations of traditional graph algorithms for searching the shortest path, an experiment was conducted in which a door opening task was performed, as depicted in Fig. 8 left. The results of this experiment reveal why not all nodes can be discovered by these methods, as new nodes might become reachable only after some others have been visited. The door was equipped with mechanical locking for lever-handle-operated doors. To open the locked door, it was first necessary to turn the hook up to unlock it before proceeding with the ordinary door-opening procedure (see Fig. 8). The robot's task was to learn the policy for door opening autonomously. The door hook pose was defined with kinesthetic teaching.

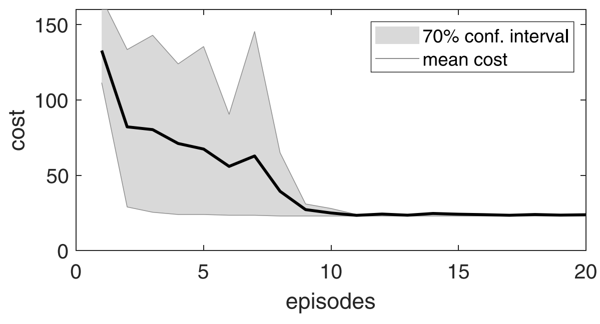
After the robot grabbed the hook, it generated forces and torques in random directions. When a randomly selected force caused a motion, it continued in that direction as described in the 3.2 section. This way, it tested all possible combinations of actions and learned the optimal sequence of movements to open the door, regardless of whether it was unlocked or locked. Moreover, it autonomously learned a general policy for opening unlocked and locked doors in approx. seven episodes. Due to the limited workspace, the robot could not fully open the door. Therefore, we assigned a positive reward when the door opening angle exceeded 20 deg. The exploration algorithm autonomously discovered the graph which describes the above learning process, presented in Fig. 8. Since learning to open a door with multi-point locking is more complex than a standard locking system and requires knowledge of previous actions to fulfill the MDP property, the graph states were determined with $\kappa = 1$ in Eq. (10). The learned policy starts in state 1 and continues visiting states denoted by 2 and 3, the final state describing the door open state. If the door is locked, we set $\kappa = 1$ as explained in Section 3.3.1. The robot can initially not proceed to state 6. Instead, it moves to states 3, 4, 5, 6, and 7 to unlock and open the door. The algorithm finds all possible actions autonomously and encodes them as DMPs in the set $\mathcal{A}(s_i)$. State 3 deserves attention since the system applying action 1 remains in the same position B. This way, the algorithm recognizes that the door is locked. Note again that the state labeling depends on the initial robot motion and the initial state of the door. However, state labeling does not affect the learned policy. Fig. 8 right shows the case where the robot started to turn the hook down on the initially unknown state (locked/unlocked) of the doors. The convergence of the RL algorithm at the top of the proposed hierarchical scheme, which learns the appropriate action sequences for the door opening, is shown in Fig. 9. In this experiment, the Q-learning algorithm (13) was enhanced with eligibility traces [41]. On average, the algorithm took six episodes to learn the policy.

Previous research has often discussed learning to open doors, allowing us to compare the effectiveness of these approaches. First, we mention that none of the studies addressed the mechanical door-locking mechanism and limited themselves to the problem of learning to open a standard door. With our graph-based approach, we can immediately notice that no decision is required when opening a standard door. Consequently, the CSF controller can discover the required sequence of moves in a single trial. In contrast, standard door opening learning by PI^2 algorithm required more than 300 trials to learn the policy [44]. In [45], authors investigate accelerated learning of door opening with two robot agents. They learned the final policy across 20 consecutive trials while learning with one agent was much slower. The approach proposed in [6], which combines PI^2 learning and compliant controller, required 9 roll-outs on average. Another approach [46] proposed policy learning in the simulated environment and application of learned policy in a real environment. Learning in a simulated environment took approx 300 trials. Although compared approaches also perused additional objectives, such as end-to-end learning, the comparison demonstrates the efficiency of the proposed framework.

Learning to open doors and execution of the learned policy is shown in the Door Opening video in the supplementary materials.

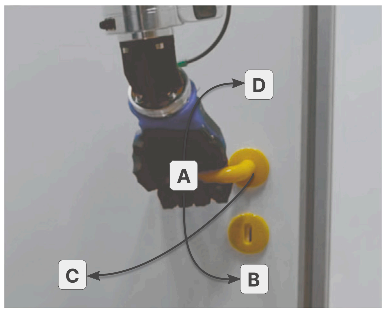


(a) Basic case where the robot has to find only the shortest path to exit the maze

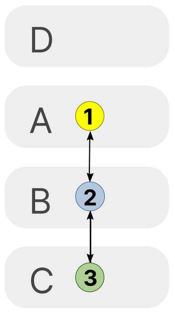


(b) A more complex case where the robot has to visit state 4 to unlock the exit.

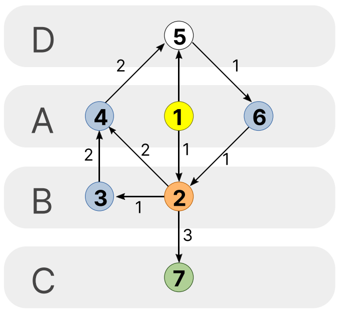
Fig. 7. Convergence of the RL for maze learning shows learning cost vs. episodes. Learning cost is the path length for escaping the maze. The shaded region denotes a 70% confidence interval of 20 epochs of learning.



(a) Door opening stages

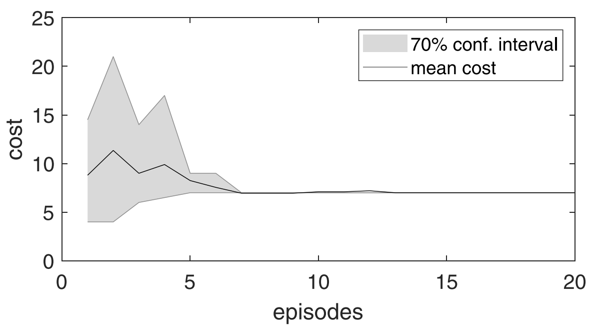


(b) No locking mechanism ($\kappa = 0$)

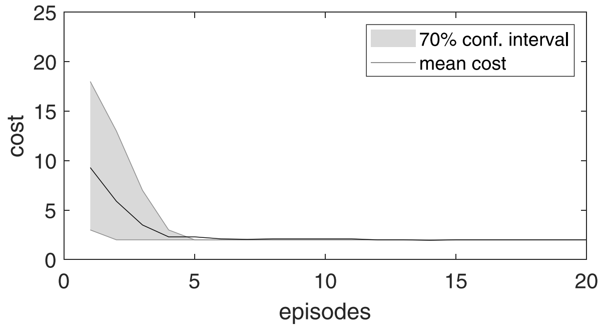


(c) Locking mechanism ($\kappa = 1$)

Fig. 8. Different stages of door opening are shown in scheme (a). If the door is locked, it is first necessary to unlock the mechanism by going to position D before proceeding with the ordinary opening procedure (going from A through B in position C). The graph in subfigure (b) corresponds to the situation when the locking mechanism is disabled. The graph shows that no learning is required at the top level of the hierarchy in this case. The graph in subfigure (c) shows the resulting graph from an episode of the learning procedure when the door is initially locked. Note that multiple nodes are created as the nodes also contain information about the previously visited nodes. The action labels describe the semantic meaning of the action: 1 - pushing down, 2 - pushing up, and 3 - pulling the hook.



(a) Learning convergence for locked doors



(b) Learning convergence for unlocked doors

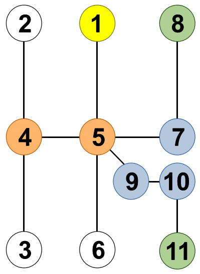
Fig. 9. Convergence of the RL for door opening learning shows learning cost vs. episodes. The cost is the number of states the robot visits before opening the door. Due to the different costs for opening unlocked and locked doors, the learning convergence is shown in two plots for unlocked and locked doors, respectively. Note also that the aim is to learn the general policy regardless of whether the doors are locked or unlocked. However, if we knew the door's state in advance, the cost of opening the locked door would be 4. During the learning, the algorithm occasionally finds this policy but rejects it as it does not fit the general case of the unknown door's state.

4.3. Learning to shift car gears

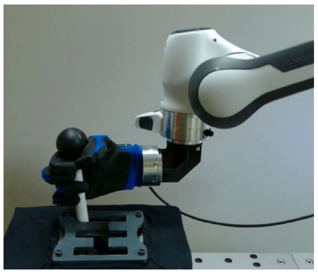
In this experiment, the robot autonomously learns to shift manual car gear transmissions (See Fig. 10). The goal was to learn how to shift from the neutral position to gears 1, 2, 3, 4, and 5 and from the neutral position to the reverse gear. First, we show the robot how to grip the gear lever with kinesthetic guidance. As the initial pose for learning, we chose third gear, but any other position could be chosen. The robot autonomously tests all possible ways to move the gear lever in all Cartesian axes. We instructed the robot not to test the gear lever orientations. Therefore, the robot was all the time compliant in all orientational d.o.f. The semantic mapping between the robot pose and the gear numbers is given in advance. The goal was to learn the policies of how to shift into any gear from any position.

Learning began with exploiting additional force in the positive normal axis of the FS frame. The resulting robot trajectory is denoted with red in Fig. 10(d) Given that choice, the search algorithm labeled states as shown in Fig. 10(b). Note again that a new node is added

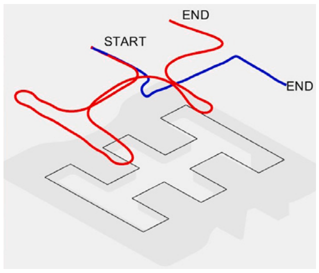
(a) A 5-speed car gear lever



(b) Graph representation

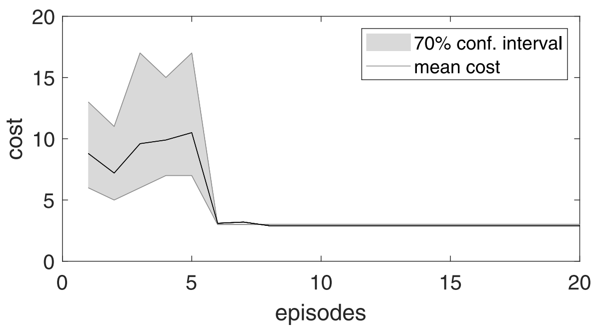


(c) Robot shifts mock-up car gear

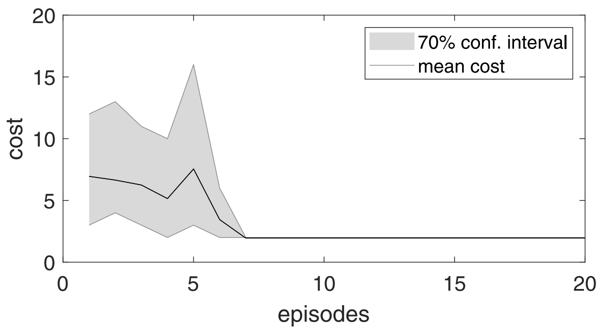


(d) Trajectory visualization

Fig. 10. (a) Standard gear lever. (b) Graph with states during the learning to shift manual transmission, as discovered and labeled by the search algorithm. Circles in two different colors represent states that can be considered target states, depending on the task requirements. (c) Robot shifts a mockup manual transmission. (d) The trajectory discovered with additional force in the normal direction and search force in the binormal direction of the FS frame are denoted with red and blue, respectively.



(a) Shifting from the second gear to the third.



(b) Shifting from third gear to the fourth.

Fig. 11. Convergence of the RL for gear shifting learning shows learning cost (i.e. number of states visited) vs. episodes for two shifting sequences.

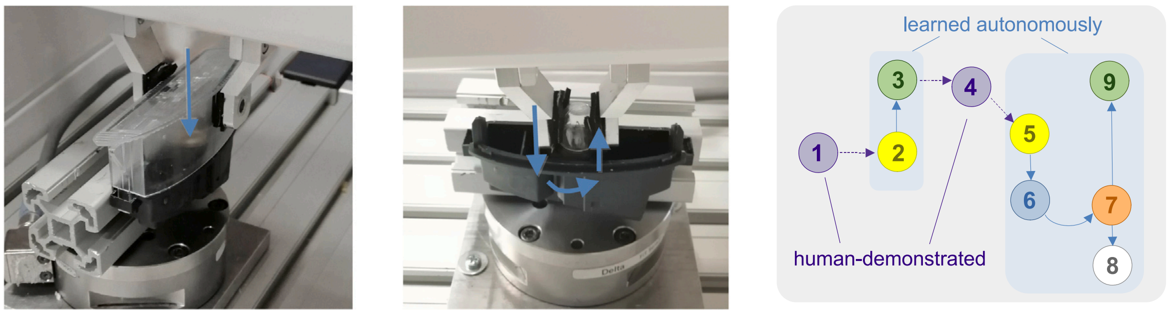
whenever the robot has to search in a completely new direction and whenever there are multiple ways to continue the motion. We set the fifth gear as a target node. Search with additional force in a negative normal direction did not discover any new nodes. Next, we repeated the search with an additional force in the negative binormal direction. The binormal axis of the FS frame in this experiment almost always coincides with the global z coordinate. The robot found new nodes this time and ended in the reverse gear. The resulting robot trajectory is denoted with blue in Fig. 10(d). After the robot discovered all possible nodes, it also learned how to shift from the neutral to the first, second, fourth, and fifth gear. The exception is shifting from the second to the third gear and from the third to the fourth gear because the robot did not encounter this combination while discovering new nodes. Q learning at the top of the hierarchical scheme was assigned to discover two missing aforementioned policies. The algorithm learns them in 6 to 7 cycles, as shown in Fig. 11.

A demonstration can be found in the Gear Shift video in the supplementary materials.

4.4. Car licence plate light disassembly and assembly

With this experiment, we intended to show how to learn the assembly/disassembly of an object which consists of multiple parts, such as a car license plate light. It consists of a base part with the bulb case, bayonet bulb, and transparent cover. The base of the light is firmly attached to the base plate, as shown in Fig. 12 left and center. Again, we start the disassembly process with a manual guide to the suitably chosen pose, where the robot gripper can firmly grasp the cover of the light. The cover is attached to the base with two side pins. To release them, it is necessary to apply a force in the z direction and a torque around the y axis. The CSF controller finds the appropriate direction by exploration as described in Section 3.2. The corresponding graph is trivial in this case. For the bayonet bulb disassembly, the actions are more complex, as it is necessary to push the bulb down in z axis, rotate it around z axis, and pull it in z axis. At this stage, it is also possible to push the bulb in the z axis, but in this case, the robot arrives at a node of degree 0 and has to turn back. The corresponding graph is shown in Fig. 12 right. Note that this graph differs from the others presented in this work as it also includes actions demonstrated by a human. State 1 describes the manual guidance of the robot to the light cover. State 4, on the other hand, describes the actions where the human operator guides the robot to the place where he releases the cover and, after that, to the place where he grabs the light bulb. The robot could also perform these actions autonomously using robotic vision. However, this was not the subject of our research. The role of RL at the highest hierarchical level, in this case, is minor since the graph has a single decision state, and the learning algorithm finds the final solution in two steps in the worst-case scenario.

During disassembly, the applied forces and torques inherently align the bulb to slide along the casing, as the center of compliance is in the robot gripper. During assembly, however, the situation is reversed, and it is necessary to actively control the bulb orientation to align it with the casing. In robotics, this is the well-known PiH problem, where it is necessary to obtain the remote center of compliance using an appropriate force control strategy [47] or apply additional learning [19]. This is also one of the reasons why the disassembly is easier than the corresponding assembly. This makes it easier to learn the assembly semantics through disassembly easier, as already suggested in [7]. The robot generates the corresponding force and torque with an impedance controller by applying a displacement in z coordinate and a rotation around the z coordinate. This experiment is also demonstrated with the corresponding Licence Plate video in the supplementary materials.



(a) Removing cover (b) Removing bulb (c) Two-stage disassembly graph

Fig. 12. Panda robot during the license car plate light disassembly and assembly. (a) removal of the transparent cover. (b) disassembly of the bayonet bulb. (c) disassembly graph. The states in violet color represent human demonstration.

5. Conclusion

Autonomous learning of robotic tasks in close contact with the environment is one of the still open challenges in modern robotics. This paper presents a novel approach based on a task representation with directed graphs. Based on this formulation, we propose a three-level hierarchical learning scheme to solve the learning problem. The highest hierarchical level makes decisions based on exploratory movements generated by the newly developed CSF controller at the lowest level. The distinguishing feature of the CSF controller is that it allows the specification of variable compliance along the robot motion. The middle level is dedicated to the discovery of new nodes and edges in the task graph. The main advantage of the hierarchical scheme is accelerated learning, which requires only a few learning roll-outs for typical tasks encountered in everyday life and industrial plants. Another advantage is that it generates continuous-time control policies using classical discrete-time RL methods such as Q-learning or SARSA.

We also considered the possibility of using alternative algorithms to find the optimal sequence of transitions at the highest hierarchical level of the proposed framework. Some problems can be successfully solved by finding the shortest paths between the nodes in a graph. These methods are generally faster than RL algorithms but less general. For example, learning to assemble a car license plate light, gear shifting, and the first maze learning problem could also be solved using graph exploration methods (breadth-first search or depth-first search) and Dijkstra's algorithm [48]. On the other hand, graph searching with RL can solve more complex problems where finding the shortest paths in the graph fails. Two of them were considered in our research, i.e., opening the door equipped with a lever-handle-locking mechanism and the second example of maze learning. A truly autonomous robot should be able to solve various problems regardless of their complexity. Thus RL is a more reasonable choice for finding the optimal path through the graph in our hierarchical scheme.

Our framework for autonomous learning of contact policies was verified in four experiments. In the first experiment, we chose the well-known maze learning problem because it nicely illustrates the essence of our approach based on directed graphs and algorithms for finding paths through such graphs. Next, we performed two experiments from everyday life: opening the door and shifting the car gearbox. Finally, we provide an example of learning assembly tasks by autonomous disassembly.

Our experiments focused on tasks where movement is allowed in a single, constantly changing direction. There are many such tasks in our daily lives and in industrial production processes. In addition to the discussed and numerous other assembly and disassembly tasks, there are tasks such as screwing, connecting BNC connectors, handling valves and levers, opening drawers and cabinets, etc. The proposed approach is directly applicable to such tasks. Moreover, it can be extended to tasks that allow movement in several degrees of freedom, which will be the subject of our future research.

CRediT authorship contribution statement

Mihael Simonič: Methodology, Software, Visualization, Writing. **Aleš Ude:** Conceptualization, Methodology, Writing. **Bojan Nemec:** Conceptualization, Methodology, Software, Supervision, Visualization, Writing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

The research leading to these results has received funding from the EU Horizon 2020 Research and Innovation Action ReconCycle, GA no. 871352, and program group P2-0076 Automation, Robotics, and Biocybernetics supported by the Slovenian Research and Innovation Agency.

Appendix A

Lemma 1. *Let the robot be controlled using impedance control law defined by Eqs. (5)–(8), with diagonal stiffness and damping matrices specified in Frenet–Serret (FS) coordinate frames (1) distributed along the robot path. If we choose equal stiffness and damping constants in the normal and binormal direction of motion, then the commanded torque (5) and thus the robot motion is independent of the direction of the normal and binormal vector that defines the FS frame.*

Proof. As a first step, let us recall that the control accelerations are computed according to Eq. (7). Assuming that stiffness and damping in the directions orthogonal to the tangential direction are equal, i.e., $k_{yz} = k_y = k_z$, $d_{yz} = d_y = d_z$, we have

$$\mathbf{K}_p = \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_{yz} & 0 \\ 0 & 0 & k_{yz} \end{bmatrix}, \quad \mathbf{D}_p = \begin{bmatrix} d_x & 0 & 0 \\ 0 & d_{yz} & 0 \\ 0 & 0 & d_{yz} \end{bmatrix}. \tag{14}$$

Next we rewrite Eq. (7) with the above stiffness and damping matrices

$$\begin{aligned}\ddot{\boldsymbol{p}}_c &= \begin{bmatrix}\boldsymbol{t} & \boldsymbol{n} & \boldsymbol{b}\end{bmatrix}\begin{bmatrix}k_x & 0 & 0\\ 0 & k_{yz} & 0\\ 0 & 0 & k_{yz}\end{bmatrix}\begin{bmatrix}\boldsymbol{t}^{\mathrm{T}}\\ \boldsymbol{n}^{\mathrm{T}}\\ \boldsymbol{b}^{\mathrm{T}}\end{bmatrix}\boldsymbol{e}_p\\ &\quad+\begin{bmatrix}\boldsymbol{t} & \boldsymbol{n} & \boldsymbol{b}\end{bmatrix}\begin{bmatrix}d_x & 0 & 0\\ 0 & d_{yz} & 0\\ 0 & 0 & d_{yz}\end{bmatrix}\begin{bmatrix}\boldsymbol{t}^{\mathrm{T}}\\ \boldsymbol{n}^{\mathrm{T}}\\ \boldsymbol{b}^{\mathrm{T}}\end{bmatrix}\dot{\boldsymbol{p}}\\ &= (k_x\boldsymbol{t}\boldsymbol{t}^{\mathrm{T}} + k_{yz}(\boldsymbol{n}\boldsymbol{n}^{\mathrm{T}} + \boldsymbol{b}\boldsymbol{b}^{\mathrm{T}}))\boldsymbol{e}_p + (d_x\boldsymbol{t}\boldsymbol{t}^{\mathrm{T}} + d_{yz}(\boldsymbol{n}\boldsymbol{n}^{\mathrm{T}} + \boldsymbol{b}\boldsymbol{b}^{\mathrm{T}}))\dot{\boldsymbol{p}}. \qquad (15)\end{aligned}$$

Let us now define another coordinate frame $\mathbf{R}'_p = \begin{bmatrix}\boldsymbol{t}' & \boldsymbol{n}' & \boldsymbol{b}'\end{bmatrix}$, with the first column defined by the tangent of robot motion, i.e., $\boldsymbol{t}' = \boldsymbol{t}$, but with the other two orthogonal axes $\boldsymbol{n}'$ and $\boldsymbol{b}'$ chosen arbitrarily. If the diagonal stiffness and damping matrices (14) are defined in this frame and with equal stiffness and damping in the direction of $\boldsymbol{n}'$ and $\boldsymbol{b}'$, then the corresponding control acceleration (7) in the robot base coordinate frame is given by

$$\ddot{\boldsymbol{p}}'_c = (k_x\boldsymbol{t}\boldsymbol{t}^{\mathrm{T}} + k_{xy}(\boldsymbol{n}'\boldsymbol{n}'^{\mathrm{T}} + \boldsymbol{b}'\boldsymbol{b}'^{\mathrm{T}}))\boldsymbol{e}_p + (d_x\boldsymbol{t}\boldsymbol{t}^{\mathrm{T}} + d_{yz}(\boldsymbol{n}'\boldsymbol{n}'^{\mathrm{T}} + \boldsymbol{b}'\boldsymbol{b}'^{\mathrm{T}}))\dot{\boldsymbol{p}}. \qquad (16)$$

For any rotational matrix $\mathbf{R}$ it holds $\mathbf{R}\mathbf{R}^{\mathrm{T}} = \mathbf{I}$, hence

$$\mathbf{I} = \boldsymbol{t}\boldsymbol{t}^{\mathrm{T}} + \boldsymbol{n}\boldsymbol{n}^{\mathrm{T}} + \boldsymbol{b}\boldsymbol{b}^{\mathrm{T}} = \boldsymbol{t}\boldsymbol{t}^{\mathrm{T}} + \boldsymbol{n}'\boldsymbol{n}'^{\mathrm{T}} + \boldsymbol{b}'\boldsymbol{b}'^{\mathrm{T}}. \qquad (17)$$

As the tangential components are equal in the above equation, it follows that $\boldsymbol{n}\boldsymbol{n}^{\mathrm{T}} + \boldsymbol{b}\boldsymbol{b}^{\mathrm{T}} = \boldsymbol{n}'\boldsymbol{n}'^{\mathrm{T}} + \boldsymbol{b}'\boldsymbol{b}'^{\mathrm{T}}$, hence $\ddot{\boldsymbol{p}}_c = \ddot{\boldsymbol{p}}'_c$. In the same way, we can prove that also for the commanded angular velocities (8) the relation $\dot{\boldsymbol{\omega}}_c = \dot{\boldsymbol{\omega}}'_c$ holds. Thus under the assumptions of the lemma, the commanded torque (5) is independent of the direction of normal and binormal vectors that defines the FS frame. □

Appendix B. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.rcim.2023.102657.

References

[1] A. Hussein, M.M. Gaber, E. Elyan, C. Jayne, Imitation learning: A survey of learning methods, ACM Comput. Surv. 50 (2) (2017).

[2] J. Kober, J. Peters, Reinforcement learning in robotics: A survey, in: Learning Motor Skills, in: Springer Tracts in Advanced Robotics, vol. 97, Springer, Cham, 2014, pp. 9–67.

[3] D. Bristow, M. Tharayil, A. Alleyne, A survey of iterative learning control, IEEE Control Syst. Mag. 26 (3) (2006) 96–114.

[4] T. Gašpar, M. Deniša, P. Radanovič, B. Ridge, T.R. Savarimuthu, A. Kramberger, M. Priggemeyer, J. Roßmann, F. Wörgötter, T. Ivanovska, S. Parizi, Ž. Gosar, I. Kovač, A. Ude, Smart hardware integration with advanced robot programming technologies for efficient reconfiguration of robot workcells, Robot. Comput.-Integr. Manuf. 66 (2020) 101979.

[5] Z. Liu, Q. Liu, W. Xu, L. Wang, Z. Zhou, Robot learning towards smart robotic manufacturing: A review, Robot. Comput.-Integr. Manuf. 77 (2022) 102360.

[6] B. Nemec, L. Žlajpah, A. Ude, Door opening by joining reinforcement learning and intelligent control, in: 18th International Conference on Advanced Robotics (ICAR), Hong Kong, 2017, pp. 222–228.

[7] M. Simonič, L. Žlajpah, A. Ude, B. Nemec, Autonomous learning of assembly tasks from the corresponding disassembly tasks, in: IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids), Toronto, Canada, 2019, pp. 230–236.

[8] M. Suomalainen, Y. Karayiannidis, V. Kyrki, A survey of robot manipulation in contact, Robot. Auton. Syst. 156 (2022) 104224.

[9] I. Elguea-Aguinaco, A. Serrano-Muñoz, D. Chrysostomou, I. Inziarte-Hidalgo, S. Bøgh, N. Arana-Arexolaleiba, A review on reinforcement learning for contact-rich robotic manipulation tasks, Robot. Comput.-Integr. Manuf. 81 (2023) 102517.

[10] F.J. Abu-Dakka, B. Nemec, J.A. Jørgensen, T.R. Savarimuthu, N. Krüger, A. Ude, Adaptation of manipulation skills in physical contact with the environment to reference force profiles, Auton. Robots 39 (2) (2015) 199–217.

[11] B. Nemec, T. Petrič, A. Ude, Force adaptation with recursive regression iterative learning controller, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 2015, pp. 2835–2841.

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

[13] R. Martín-Martín, M.A. Lee, R. Gardner, S. Savarese, J. Bohg, A. Garg, Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macao, China, 2019, pp. 1010–1017.

[14] M. Hazara, V. Kyrki, Reinforcement learning for improving imitated in-contact skills, in: IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids), Cancun, Mexico, 2016, pp. 194–201.

[15] Y. Chebotar, M. Kalakrishnan, A. Yahya, A. Li, S. Schaal, S. Levine, Path integral guided policy search, in: IEEE International Conference on Robotics and Automation (ICRA), Singapore, 2017, pp. 3381–3388.

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

[17] Z. Hou, J. Fei, Y. Deng, J. Xu, Data-efficient hierarchical reinforcement learning for robotic assembly control applications, IEEE Trans. Ind. Electron. 68 (11) (2021) 11565–11575.

[18] Y.-L. Kim, K.-H. Ahn, J.-B. Song, Reinforcement learning based on movement primitives for contact tasks, Robot. Comput.-Integr. Manuf. 62 (2020) 101863.

[19] L. Johannsmeier, M. Gerchow, S. Haddadin, A framework for robot manipulation: Skill formalism, meta learning and adaptive control, in: 2019 International Conference on Robotics and Automation (ICRA), Montreal, Canada, 2019.

[20] S. Nair, M. Babaeizadeh, C. Finn, S. Levine, V. Kumar, TRASS: Time reversal as self-supervision, in: IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 115–121.

[21] K. Zakka, A. Zeng, J. Lee, S. Song, Form2Fit: Learning shape priors for generalizable assembly from disassembly, in: IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 9404–9410.

[22] M.T. Mason, Compliance and force control for computer controlled manipulators, IEEE Trans. Syst. Man Cybern. 11 (6) (1981) 418–432.

[23] A.J.D. Lambert, Disassembly sequencing: A survey, Int. J. Prod. Res. 41 (16) (2003) 3721–3759.

[24] J. Aleotti, S. Caselli, Physics-based virtual reality for task learning and intelligent disassembly planning, Virtual Real. 15 (1) (2011) 41–54.

[25] E. Tuncel, A. Zeid, S. Kamarthi, Solving large scale disassembly line balancing problem with uncertainty using reinforcement learning, J. Intell. Manuf. 25 (4) (2014) 647–659.

[26] K. Erciyes, Distributed Graph Algorithms for Computer Networks, Springer, London, 2013.

[27] H. Bruyninckx, J. De Schutter, Where does the task frame go? in: Y. Shirai, S. Hirose (Eds.), Robotics Research, Springer London, London, 1998, pp. 55–65.

[28] R. Ravani, A. Meghdari, Velocity distribution profile for robot arm motion using rational frenet-serret curves, Informatica 17 (1) (2006) 69–84.

[29] B. Nemec, N. Likar, A. Gams, A. Ude, Human robot cooperation with compliance adaptation along the motion trajectory, Auton. Robots 42 (5) (2018) 1023–1035.

[30] M. Pilté, S. Bonnabel, F. Barbaresco, Tracking the frenet-serret frame associated to a highly maneuvering target in 3D, in: IEEE 56th Annual Conference on Decision and Control (CDC), Melbourne, Australia, 2017, pp. 1969–1974.

[31] D. Carroll, E. Köse, I. Sterling, Improving frenet's frame using bishop's frame, J. Math. Res. 5 (4) (2013) 97–106.

[32] M. Vochten, T. De Laet, J. De Schutter, Robust optimization-based calculation of invariant trajectory representations for point and rigid-body motion, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 2018, pp. 5598–5605.

[33] G. Niemeyer, J.-J.E. Slotine, A simple strategy for opening an unknown door, in: IEEE International Conference on Robotics and Automation (ICRA), Albuquerque, NM, 1997, pp. 1448–1453.

[34] J.-O. Lachaud, A. Vialard, F. de Vieilleville, Analysis and comparative evaluation of discrete tangent estimators, in: Proceedings of the 12th International Conference on Discrete Geometry for Computer Imagery, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 240–251.

[35] A. Albu-Schaffer, C. Ott, G. Hirzinger, A unified passivity-based control framework for position, torque and impedance control of flexible joint robots, Int. J. Robot. Res. 26 (1) (2007) 23–39.

[36] C. Ott, A. Albu-Schaffer, A. Kugi, S. Stramigioli, G. Hirzinger, A passivity based cartesian impedance controller for flexible joint robots - part I: torque feedback and gravity compensation, in: IEEE International Conference on Robotics and Automation (ICRA), New Orleans, LA, 2004, pp. 2659–2665.

[37] A. Dietrich, A. Albu-Schäffer, G. Hirzinger, On continuous null space projections for torque-based, hierarchical, multi-objective manipulation, in: IEEE International Conference on Robotics and Automation (ICRA), IEEE, Saint Paul, Minnesota, 2012, pp. 2978–2985.

[38] A. Ude, B. Nemec, T. Petrič, J. Morimoto, Orientation in Cartesian space dynamic movement primitives, in: IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, 2014, pp. 2997–3004.

[39] R. Vuga, B. Nemec, A. Ude, Speed adaptation for self-improvement of skills learned from user demonstrations, Robotica 34 (12) (2016) 2806–2822.

[40] A.M. Sadik, M.A. Dhali, H.M. Farid, T.U. Rashid, A. Syeed, A comprehensive and comparative study of maze-solving techniques by implementing graph theory, in: International Conference on Artificial Intelligence and Computational Intelligence, 2010, pp. 52–56.

[41] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, Second Edition, The MIT Press, Cambridge, MA, 2015.

[42] S. Russell, P. Norvig, Artificial Intelligence, fourth ed., Pearson, Upper Saddle River, NJ, 2020, p. 648.

[43] C.D. Santina, G. Grioli, M.G. Catalano, A. Brando, A. Bicchi, Dexterity augmentation on a synergistic hand: The pisa/IIT SoftHand+, in: IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids), Seoul, Korea, 2015, pp. 497–503.

[44] E. Theodorou, F. Stulp, J. Buchli, S. Schaal, An iterative path integral stochastic optimal control approach for learning robotic tasks, IFAC Proc. Vol. 44 (1) (2011) 11594–11601.

[45] S. Gu, E. Holly, T. Lillicrap, S. Levine, Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates, in: 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 3389–3396.

[46] Y. Wang, L. Wang, Y. Zhao, Research on door opening operation of mobile robotic arm based on reinforcement learning, Appl. Sci. 12 (10) (2022).

[47] J. Jiang, Z. Huang, Z. Bi, X. Ma, G. Yu, State-of-the-art control strategies for robotic PiH assembly, Robot. Comput.-Integr. Manuf. 65 (2020) 1–26.

[48] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, second ed., The MIT Press, Cambridge, MA, 2001.

Mihael Simonič received B.Sc. and M.Sc. degrees in cognitive science from the University of Tübingen, Germany and the doctoral degree from the Faculty of Electrical Engineering of the University of Ljubljana, Slovenia in 2023. He works at the Department of Automatics, Biocybernetics and Robotics, Jožef Stefan Institute, Ljubljana, Slovenia. His current research focuses on the intersection between robot learning and human–robot collaboration.

Aleš Ude received the diploma degree in applied mathematics from the University of Ljubljana, Ljubljana, Slovenia, and the Dr.Eng. degree from the University of Karlsruhe, Karlsruhe, Germany, in 1990 and 1995, respectively. He is currently the Head of the Department of Automatics, Biocybernetics, and Robotics, Jožef Stefan Institute, and a professor at the Faculty of Electrical Engineering, Ljubljana. He is also associated with the ATR Computational Neuroscience Laboratory, Kyoto, Japan. His research interests include robot learning, programming by demonstration, reconfigurable robotic systems, and humanoid robotics.

Bojan Nemec received B.Sc and M.Sc. degrees in electrical engineering and Ph.D. degree in robotics from the University of Ljubljana, Slovenia. He is currently the head of the Humanoid and Cognitive Robotics Lab and Research Councillor with the Department of Automatics, Biocybernetics, and Robotics, Jožef Stefan Institute, and a professor at Jožef Stefan International Postgraduate School, Ljubljana. His research interests include robot control, robot learning, service robotics, and sports biomechanics.