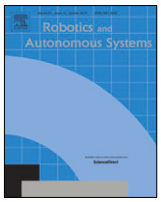
Robotics and Autonomous Systems 162 (2023) 104367

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

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



Online task segmentation by merging symbolic and data-driven skill recognition during kinesthetic teaching☆

Thomas Eiband [a,*], Johanna Liebl [a], Christoph Willibald [a], Dongheui Lee [a,b]

[a] German Aerospace Center (DLR), Institute of Robotics and Mechatronics, Wessling, 82234, Germany
[b] TU Wien, Autonomous Systems, Vienna, 1040, Austria

ARTICLE INFO

Article history:
Available online 20 January 2023

Keywords:
Learning from demonstration
Programming by demonstration
Robot
Symbolic
Subsymbolic
Data-driven
Task segmentation
Action segmentation
Skill recognition
Task representation
Interactive robot programming
Intuitive robot programming
Force-based
Tactile
Online segmentation
Kinesthetic teaching

ABSTRACT

Programming by Demonstration (PbD) is used to transfer a task from a human teacher to a robot, where it is of high interest to understand the underlying structure of what has been demonstrated. Such a demonstrated task can be represented as a sequence of so-called actions or skills. This work focuses on the recognition part of the task transfer. We propose a framework that recognizes skills online during a kinesthetic demonstration by means of position and force–torque (wrench) sensing. Therefore, our framework works independently of visual perception. The recognized skill sequence constitutes a task representation that lets the user intuitively understand what the robot has learned. The skill recognition algorithm combines symbolic skill segmentation, which makes use of pre- and post-conditions, and data-driven prediction, which uses support vector machines for skill classification. This combines the advantages of both techniques, which is inexpensive evaluation of symbols and usage of data-driven classification of complex observations. The framework is thus able to detect a larger variety of skills, such as manipulation and force-based skills that can be used in assembly tasks. The applicability of our framework is proven in a user study that achieves a 96% accuracy in the online skill recognition capabilities and highlights the benefits of the generated task representation in comparison to a baseline representation. The results show that the task load could be reduced, trust and explainability could be increased, and, that the users were able to debug the robot program using the generated task representation.

1. Introduction

Recent years have shown a shift in the production cycle from large product batches to small ones, as the demand for customized products grows [1]. At the same time, automation of the production cycle has become a major goal for many companies, as robots become more versatile and cheaper, and their use becomes attractive in labor-intensive production chains. These two trends combined lead to the demand of robots that learn new tasks quickly and efficiently in order to keep up with the changing production cycle. The traditional form of robot programming stands in the way of this change, as people working in production often lack the background and also the time to implement these changes themselves, and experts must provide the code instead.

Therefore, methods that allow teaching robots new tasks without having to implement code have become a much researched topic. Programming by Demonstration (PbD) is one of these approaches, which allows the teacher to demonstrate new behaviors that the robot learns through imitation. As this is similar to the way humans teach one another, it is intuitive to use and requires no background knowledge in robotics or traditional programming [2]. While PbD allows non-experts to program robots, it sill does not mean that all human demonstrators are good teachers. The demonstrations non-experts give to machines are often sub-optimal [3], which stems from a mismatch of the mental model the users have of the robot and the robot's real knowledge [4]. To improve the user's teaching capability by resolving this mismatch, it is helpful to provide the user with feedback about what the robot has learned [5]. Furthermore, an understandable representation of the robot's knowledge is an essential part of giving a robot the ability to explain itself. This helps to build trust in robots, which is necessary to make non-experts feel at ease when using robots and gives them the insight necessary to debug incorrect robot programs [6].

Therefore, we propose a skill recognition technique that helps users to understand what a robot has learned from a demonstration. We define skill recognition as a technique to segment a task into a sequence of meaningful, understandable steps that are

☆ This work was supported by the Helmholtz Association.
* Corresponding author.
E-mail address: thomas.eiband@dlr.de (T. Eiband).

https://doi.org/10.1016/j.robot.2023.104367

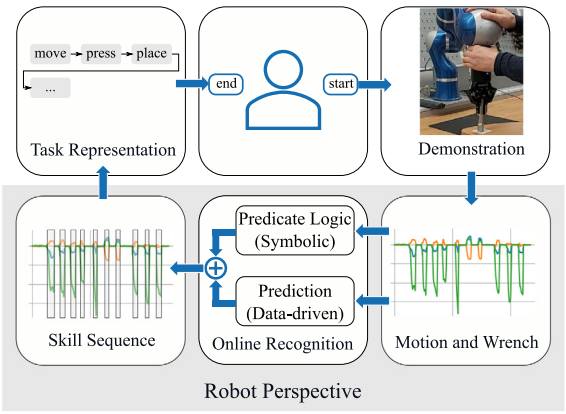


Fig. 1. An overview of the information flow in our framework. The user provides a demonstration, which is segmented by the system to recognize appropriate robot skills. At the same time, the user can monitor the evolving task representation, which can be edited at any time.

represented as robot skills. This is achieved by solving a segmentation and identification problem simultaneously. We introduce a combination of symbolic and data-driven skill recognition to extract a rich task representation consisting of manipulation skills as well as force-based skills. This combination is proposed to overcome the limitations of existing approaches that use only one of the techniques. The overview of our approach is shown in Fig. 1, where the human is in the loop as a teacher and able to monitor the robot's understanding of the task.

Many existing works represent a demonstrated task in the form of a sequence of actions. In the literature, these are also known as activities, planning operators, subtasks, and skills. We stick to the term skills, as defined in [7], which are a meaningful representation in industrial use cases. We divide the state of the art of skill recognition in two major categories, based on their recognition methodology. On the one hand, we see approaches that make use of symbolic action descriptions as well as pre- and post-conditions to recognize skills from a demonstration. On the other hand, we see data-driven approaches that detect skills based on a trained model, which requires a dataset of previously seen skills. Both categories have their advantages, which are best understood in the following example of kinesthetic teaching. Assume that the end user opens the robot's gripper during the task demonstration. This event can be easily described on a symbolic level, for instance evaluating if the symbol `gripperOpen` is true or false. Such events help to build preconditions for robotic skills such as *open-gripper* or *place*

Using a symbolic approach here is straightforward, since the underlying conditions can be easily designed, implemented, and evaluated with minimal cost and with high accuracy. Training a classifier on opened and closed gripper states would be computationally inefficient. Assume now that the user moves the robot tool along a surface while applying force onto the plane. This results in a time-series of measurements that can be hardly described by manually designed rules. Therefore, a trained classifier is a good choice to recognize an appropriate skill for this phase of the demonstration.

1.1. Symbolic approach

In a symbolic approach, names are assigned to specific robot states in order to define the meaning of represented knowledge. These names are called symbols, which can be used to form predicates in order to evaluate a skill's pre- or post-condition. Based on a survey about robot knowledge representations [8], a well-known syntax to define these conditions is the Planning Domain Definition Language (PDDL) [9]. These conditions can be evaluated during demonstration in order to identify a skill that matches them. We call this the symbolic approach of recognizing a skill from a demonstration. An overview of the state of the art for the understanding of human movements can be found in [10], which focuses on the semantic aspects of recognition algorithms.

The action recognition system presented in [11,12] used a so-called "predicator" that produces a symbolic description of the scene based on visual perception. A definition of a robot skill in an industrial setup is given by [7], describing potential skill architectures with pre- and post-condition checks. Here, the skill sequence comprising a task is programmed manually and later parameterized. In comparison, we recognize the skill type itself on the fly instead of only parameterizing it.

In the motion-sensor PbD approach used in [13], each skill has a set of unique postconditions. These postconditions describe the effects which the skill has on the robot's state as well as on the environment. Other approaches additionally make use of pre-conditions, describing which conditions need to be fulfilled before or during the skill, such as [14]. Here, skills were described in the Planning Domain Definition Language (PDDL). To check if their conditions are fulfilled, a world model was used, providing the positions of all relevant objects in the workspace, as well as the robot's state. Since such a model can diverge from the real world and lead to incorrect skill recognition, [15] used image detection to evaluate which conditions are fulfilled. It introduces Object–Action–Complexes (OACs), where the conditions of skills describe contact changes between objects.

A drawback of pure symbolic segmentation methods can be that pre- and post-conditions of each skill need to be designed by hand. This can make it difficult to find distinguishing symbolic expressions for these conditions and also requires more manual effort, especially for skills that act on the same object but differ only through the forces they apply onto them. There have been attempts to extract pre- and post-conditions [16–18]. Nevertheless, none of these approaches were applied on a variety of force-based interactions, like consecutive in-contact movement primitives. Our approach is capable of distinguishing these skills by combining symbolic segmentation with data-driven segmentation, learning the differences between skills from data instead of manually describing their characteristics.

1.2. Data-driven approach

A data-driven approach learns to discriminate skills based on a pre-trained model or by finding grouping patterns in data. For example, observed motions can be encoded in probabilistic models and later on, similar motions can be recognized by comparing them with the existing models [19]. Similarly, anomalies can also be identified by comparing measurements with previously observed executions [20]. In [21], a framework for task structure extraction was presented, which relies on the activity recognition method presented in [22]. Here, activities are predicted based on vision data. Instead, we propose our own recognition method that works without visual input and relies purely on motion and force data. Our previous work [23] only targeted the recognition of force-based skills, so called *Contact Skills*. A support vector machine (SVM) was trained on such physical contact instances and predicted the skill occurrences as part of a task demonstration. Pick and place skills as well as free motions were not considered.

The approach in [24] segments a demonstration based on the analysis of the variance among demonstrations compared to the variability of the data within a time window. Additionally, different object frames are exploited to assign them to each of the segments. As stated above, this requires again a world

model with associated coordinate frames. This approach works only with multiple demonstrations of the same task, which is a time-consuming programming paradigm.

Template matching with HMMs was used in [25] to visually identify skills based on previously trained templates. The approach in [26] employs SVMs and a sliding window on the demonstration data to detect skills. Additionally, the segmentation lines are improved by a local optimal boundary search. However, both of these approaches also limit the detection of skills to visually observable behaviors and ignore interaction forces or torques. To tackle this limitation, both tactile and proprioceptive signals were exploited in [27], presenting a segmentation algorithm that is based on Bayesian online changepoint detection.

Unsupervised approaches try to find similarities between demonstrations or fit generative models to data for segmentation. In [28], similar skill transition states in repeated demonstrations of a task are identified using hierarchical clustering based on Gaussian Mixture Models (GMM). A first GMM is fitted to every demonstration to obtain candidate transition states, that are then clustered in secondary GMMs in the spatial, sensory and temporal domain. Bayesian nonparametric extensions of the HMM are used in [29–31] where a beta process (BP) prior is leveraged to infer the number of active modes (or skills) per demonstration and additionally allows to share identified modes across different demonstrations. The Beta Process Autoregressive HMM relaxes the conditional independence of observations by describing time dependencies between observations as a Vector Autoregressive process [29,31]. A BP–HMM is combined with a clustering approach in [30] to determine the appropriate level of granularity for identified motion primitives based on clustering performance. In [32], the observed skill sequence can be inferred from a single task demonstration, where each skill's intention and feature constraints are used as grouping mechanisms in the data. The approach combines a GMM in feature space with Inverse Reinforcement Learning to capture the intention and feature similarities of every state–action observation in a joint probabilistic model.

One problem with data-driven approaches is that they require some amount of training data. For supervised approaches, training data must be also labeled, which can be an expensive process that blocks manpower, hardware, and computational resources. Running data-driven approaches also comes with a delay in prediction time due to computational cost and limited classification accuracy. These disadvantages should only be accepted if the problem cannot be solved by manually implementing rules that are easy to interpret.

1.3. Combined approach

A combined approach uses symbolic descriptions and data-driven algorithms in a meaningful symbiosis. It has been shown that symbols can even automatically emerge from the observation of a task [33]. Here, probabilistic learning of system states lets new symbols emerge automatically. The main reason to extract these symbols is that they can be used in task planning, which works with a discrete and abstract state space via the Planning Domain Definition Language (PDDL). Such symbols can be used as pre- and post-conditions and are extracted based on a learning problem. The problem is that the algorithm requires a lot of manually collected samples in the task space, such that the data covers all possible eventualities that could occur in the robot's environment. This also assumes that simple symbols such as `gripperOpen` are learned, which is contradicting to our motivation.

We manually define symbols only for features, whose evaluation can be easily implemented and which are not overly sensitive to a threshold in order to accurately detect them. An example is the opening state of a gripper, which can be evaluated by implementing a few lines of code. The defined symbols are then able to segment the data in an event-based manner whenever they change their value. Here, the evaluation of a symbol can be based on a single measurement only, which is the fastest possible response time. Although in [33], data-driven and symbolic approaches have been combined, the approach does not support online recognition of skills during user demonstration. On the other hand, only complex features should be learned that cannot be easily described by the system designer. For example, it would be hard for a programmer to take care of multi-dimensional and mutually interacting dimensions, which is also known as cross-talk. An example of cross-talk can be found in surface electromyography, where the same muscle might affect multiple EMG electrode channels at the same time [34]. Another challenge would be to manually define rules for classifying data points that are not linearly separable. Data-driven classification algorithms can address this with nonlinear decision boundaries [35]. Furthermore, information could be rather hidden in temporal sequences than in single states because a single state cannot express the process dynamics. For instance, a work about force thresholds in robotic assembly [36] could be improved by using the temporal information of force transients instead [37]. Combining the symbolic with the data-driven skill recognition utilizes the strength of both approaches. We therefore define manipulation skills such as *pick*, *place* mostly through symbols that can be easily described and implemented by simple rules. Contact skills, which are characterized by their time-series and temporal dynamics are recognized through data-driven methods.

We propose an approach for task segmentation that combines symbolic evaluation with supervised data-driven methods to recognize both manipulation and contact skills. This enables the system to build a task representation on the fly while the user performs the task demonstration. The task representation reflects how the system interpreted the demonstrated task and helps to consolidate the user's understanding about the robot's knowledge. Furthermore, it offers future options to correct it, and thus supports non-expert users in their role as programmers.

In detail, we propose a framework that (1) reduces the amount of training data due to 'outsourcing' simple skills like pick and place, which can be easily described in terms of symbolic preconditions/effects, and only learning classified skills; (2) exploits only proprioceptive data and end-effector forces while avoiding vision related issues such as problematic lighting conditions, occlusion, and inaccuracies due to calibration; (3) segments the demonstration online by two simultaneous segmentation pipelines; (4) reduces over-segmentation of data-driven classifications by usage of a segments pool and reconsideration of combined segments candidates; (5) provides live feedback in the form of a human readable task representation; and (6) enables the user to debug the graphical task representation via a graphical user interface, which is evaluated in a user study.

2. Online skill recognition

The information flow in our framework is shown in Fig. 1. The whole process starts with a user that demonstrates a task to the system. The robot uses its proprioceptive measurements to observe the human demonstration. A measurement $\boldsymbol{m} = [\boldsymbol{p}, \boldsymbol{o}, \boldsymbol{f}, \boldsymbol{\varrho}, g, h]$ consists of position $\boldsymbol{p} \in \mathbb{R}^3$, orientation $\boldsymbol{o} \in \mathbb{R}^4$ in quaternions, force $\boldsymbol{f} \in \mathbb{R}^3$, torque $\boldsymbol{\varrho} \in \mathbb{R}^3$, gripper opening width $g \in \mathbb{R}$, and gripper status $h \in \{-1, 0, 1\}$ (with -1: no object in gripper, 0: gripper moving, 1: object in gripper). At each time-step t, a measurement $\boldsymbol{m}^{(t)}$ is received by the online skill recognition. The output of the algorithm is a sequence of skill labels along with their corresponding demonstration data segments.

2.1. Skill definition

We first describe the rules for our skill definition process. We start with the skills that can be symbolically described and call them *Logic Skills*, which can be found in Fig. 2 (top table). Here, each skill is described by a human understandable precondition (pre:) and postcondition (post:). From literature surveys [8,10], it is known that manipulation skills such as *pick* and *place* are commonly used in robotics, which inspired the set in Fig. 2 (top table). An abstract skill called *contact* can be recognized by evaluating its pre- and post-conditions, which are described by the symbols `ContactRisingEdge` and `ContactFallingEdge`. These symbols are evaluated by checking if the absolute force and torque measurement exceeded or fall below a threshold respectively.[1] The skill is marked as abstract in Fig. 2, which means that it will only appear in our processing pipeline and finally be refined by our algorithm, which is explained in more detail later. A *move* skill defines a section of a demonstration where the end-effector is in motion without physical contact and where the gripper fingers are not actuated. The pre- and post-conditions `AnyLogicSkillPre` and `AnyLogicSkillPost` define the logical disjunction of all *Logic Skills*' pre- and post-conditions respectively. This leads to the fact that a *move* skill fills all existing gaps between the other available *Logic Skills*. The rest of the described skills involve gripper operations and are termed *Gripper Skills*.

Next, we define the skills that are challenging to describe by manually defined rules. We call them *Classified Skills* and present them in Fig. 2 (bottom table). We mainly rely on the identification in [23], which resulted in explainable symbols as confirmed by an independent group of subjects that classified these skills with a 90% accuracy. Note that we also added the *move* skill to be classified. This has the reason that a classifier can make a better prediction between free motion and contact as a symbolic predicate can do, which only evaluates a fixed threshold in the force domain. Additionally, a number of skills that are grouped together as *Contact Skills* can be found. All skills from this group require a physical contact with the environment. Such interactions are hard to be evaluated by manually defined conditions and are therefore classified by a data-driven model. References to exemplary applications and possible skill implementations can be found for *press* [38,39], *slide* [40,41], *contour* [41,42], *peg-in-hole* [43,44], and *user*, as for instance hand overs that are triggered by touching the robot's structure [45].

After having all skills in our framework defined, we describe how each of the recognition approaches work and how they are combined in one pipeline.

2.2. Symbolic skill recognition

The symbolic approach of this work evaluates each skill's pre- and post-conditions to recognize a suitable skill during the demonstration. The symbolic skill recognition process using exemplary data is shown in Fig. 3, step (1a). The input of the algorithm is a stream of measurements $\boldsymbol{m}$, marked as `<in>` in the figure. The output is a list of segmentation lines and skill labels, marked as `<out>`.

The pre- and post-conditions for each skill, which are specified in Fig. 2 (top) are evaluated at each time-step. These are conditions that need to be fulfilled before a skill can take place and after a skill has taken place. In our work, the conditions are computed from the movement of the robot's gripper fingers as well as the measurements from the FT-sensor installed between the gripper and the robot. As these symbols are not enough to distinguish between the different *Contact Skills*, the symbolic approach is limited to detecting an abstract *contact* skill instead. Overall, the symbolic segmentation can detect the skills specified in Fig. 2 (top).

Algorithm 1 describes the process of evaluating the pre- and post-conditions in the symbolic recognition. All skills start with their status set to `IDLE`, until the symbolic conditions indicate that the preconditions are met for any of the skills. Then, a skill's status is set to `ACTIVE`, which it continues to be as long as its post-conditions are met. Once its postconditions are met, its status is set to `DONE` and the skill is added to the list of detected skills together with its start and end time. If the `ACTIVE` state temporally overlaps for multiple skills, the one that first fulfills its post-conditions dominates the skill recognition and causes a reset of the `ACTIVE` state of all other skills. An exemplary output can be found in Fig. 3, step (1a), labeled as `<out>`.

Logic Skill Definitions

contact (abstract)	pre: `ContactRisingEdge` post: `ContactFallingEdge`
move	pre: `AnyLogicSkillPost` post: `AnyLogicSkillPre`
pick	pre: `not HoldingObject and GripperOpen` post: `HoldingObject and not GripperOpen`
place	pre: `HoldingObject and not GripperOpen` post: `not HoldingObject and GripperOpen`
close-gripper	pre: `not HoldingObject and GripperOpen` post: `not HoldingObject and not GripperOpen`
open-gripper	pre: `not HoldingObject and not GripperOpen` post: `not HoldingObject and GripperOpen`

Gripper Skills

Classified Skill Definitions

move	Move tool without contact and gripper movement
press	Press tool on surface
slide	Press and slide tool parallel to surface
contour	Tool follows nonlinear outer object contour
peg-in-hole	Insert one object into another
user	Interaction between robot and user

Contact Skills

Fig. 2. Logic and classified skill definitions.

2.3. Data-driven skill recognition on time window

Fig. 3, step (1b) shows an exemplary process of the data-driven skill recognition. The input (labeled as `<in>`) is again a stream

[1] The thresholds for force and torque were set to 5 N and 2 Nm respectively based on preliminary experiments and as inspired by [38].

Algorithm 1 Symbolic Recognition

Input: Measurements: $\boldsymbol{m}$, Set of *Logic Skills* (Fig. 2): LS
  while $\boldsymbol{m}_t$ **do**
    $condvals \leftarrow$ evaluate_conditions($\boldsymbol{m}_t$) ▷ Compute value of each predicate
    for $s \in LS$ **do**
      update_status(s, $condvals$) ▷ set IDLE, ACTIVE or DONE
      if $s.status$ == DONE **then**
        $segments([s.start : s.end]) \leftarrow s.name$ ▷ label each sample with skill name
        for $s \in LS$ **do**
          $s.status \leftarrow$ IDLE
        end for
        break
      end if
    end for
  end while
Output: *segments*

of measurements $\boldsymbol{m}$ and the output (labeled as <out>) is a list of labels. The data-driven skill recognition proposed in this work uses a support vector machine (SVM) with a sliding window approach similar to [26]. At each time-step t, a feature vector $\boldsymbol{F}^{(t)}$ is computed from the measurements $\left[\boldsymbol{m}^{(t-W+1)}, \ldots, \boldsymbol{m}^{(t)}\right]^T$ in the current time window of length W. The SVM uses this feature vector to predict which skill is most likely performed in the given time window.

The features used in this approach consist of two parts. The first part uses features that are extracted with the publicly available library *tsfresh* [46]. We use their proposed minimal feature set, consisting of seven features, which are standard deviation, sum of values, maximum, median, minimum, variance, and mean. The feature computation maps a uni-variate time series of length W to seven feature values, defined as

$$f_{\text{tsfresh}} : \mathbb{R}^W \mapsto \mathbb{R}^7.$$

Consider the time series

$$\boldsymbol{M}_{\text{tsfresh}} = \left[\boldsymbol{m}_{\text{tsfresh}}^{(t-W+1)}, \ldots, \boldsymbol{m}_{\text{tsfresh}}^{(t)}\right]^T \in \mathbb{R}^{W\times 4}$$

with a number of W samples, where a single sample is defined as

$$\boldsymbol{m}_{\text{tsfresh}}^{(t)} = [\|\dot{\boldsymbol{p}}\|, \|\dot{\boldsymbol{o}}\|, \|\boldsymbol{f}\|, \|\boldsymbol{\varrho}\|] \in \mathbb{R}^4.$$

In order to obtain $\boldsymbol{m}_{\text{tsfresh}}^{(t)}$, we compute the velocity $\dot{\boldsymbol{p}}$ from the Cartesian position $\boldsymbol{p}$ and the angular velocity $\dot{\boldsymbol{o}}$ from the orientation $\boldsymbol{o}$ of the end effector frame. Additionally, as the skills should be described independently of a task-specific frame, we compute the Euclidean norm over all dimensions of a variable $\boldsymbol{x}$ as $\|\boldsymbol{x}\|$. This gives us the magnitude of each considered modality and makes these new features invariant to a specific Cartesian frame. We obtain the feature vector $\boldsymbol{F}_{\text{tsfresh}}^{(t)} \in \mathbb{R}^{28}$ by applying f_{tsfresh} dimension-wise on $\boldsymbol{M}_{\text{tsfresh}}$ and by stacking the results.

The second part consists of manually designed features mainly taken from a previous work [20], which are denoted as $\boldsymbol{F}_{\text{add}}^{(t)}$. They are shown in Table 1 and address physical relations between motion and force as they are expected to occur in the *Contact Skills*.

A combined feature vector is constructed as $\boldsymbol{F}^{(t)} = \left[\boldsymbol{F}_{\text{tsfresh}}^{(t)}, \boldsymbol{F}_{\text{add}}^{(t)}\right]$. Finally, $\boldsymbol{F}^{(t)}$ is normalized dimension-wise to mean $\mu = 0$ and standard deviation $\sigma = 1$ for improved numerical stability in the classification.

As the strength of the data-driven segmentation lies in detecting skills by their force and torque profile, the SVMs are trained

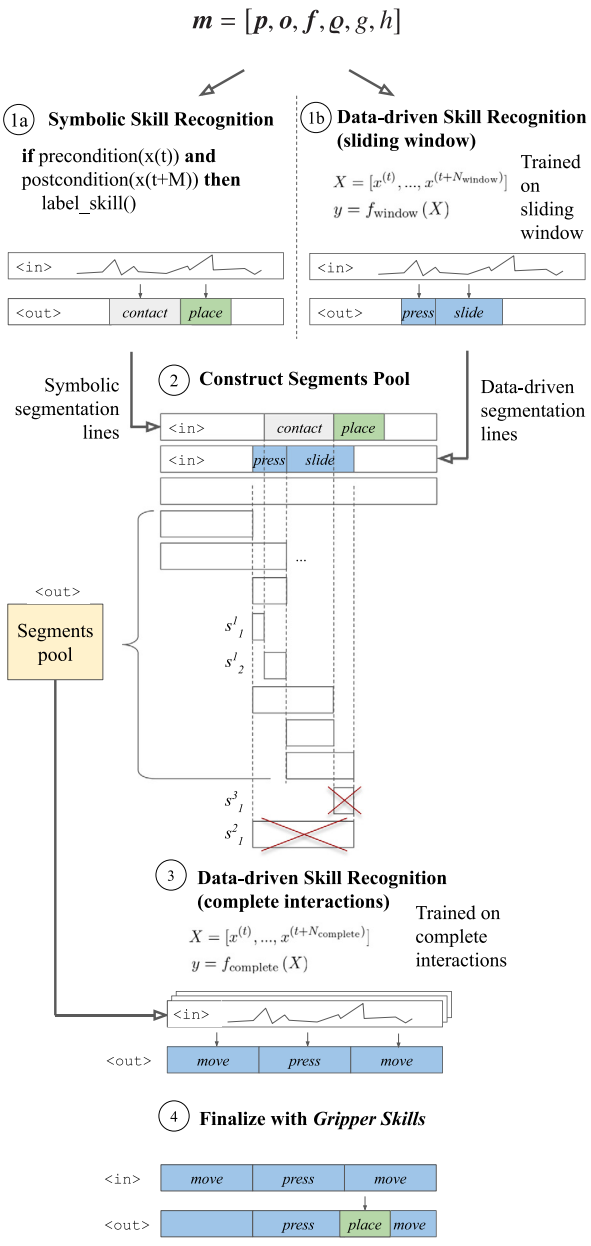


Fig. 3. Skill Recognition Pipeline. The color code is the same as in Fig. 2, where the logic skill definitions are in green, and the classified skill definitions are in blue. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

to predict the *Classified Skills* that are specified in Fig. 2 (bottom). The *Gripper Skills* specified in Fig. 2 (top) (*pick*, *place*, *open-gripper*, *close-gripper*) are not learned in a data-driven manner, as they can be robustly and efficiently identified by the symbolic segmentation without any uncertainty. The SVMs used in this approach use a Radial Basis Function (RBF) kernel. The regularization parameter and the kernel bandwidth are found through a grid search. The SVMs are trained on examples of the skills both consisting of the whole skill as well as sub-segments of the skills with the same sample length W as the sliding window.

Algorithm 2 clarifies the computational steps required for the data-driven recognition approach. While the window is sliding over the incoming measurements stream, the SVM keeps on predicting for each new sample and the predicted label is appended to a list. Note that it is no problem to predict online with a rate

Table 1
Additional contact skill features.

Feature	Definition
mean translation power	$p_f = \frac{1}{L}\sum_{t=1}^{N} \boldsymbol{f}^{(t)T} \cdot \dot{\boldsymbol{p}}^{(t)}$
mean rotation power	$p_\varrho = \frac{1}{L}\sum_{t=1}^{N} \varrho_t^T \cdot \dot{\boldsymbol{o}}^{(t)}$
slope to max force	$\Delta f = \frac{\max\left(\hat{f}^{(1)},\dots,\hat{f}^{(N)}\right) - \hat{f}^{(1)}}{\underset{n}{\operatorname{argmax}}\left\{\hat{f}^{(n)}\right\}}$ with $\hat{f}^{(t)} = \lvert\lvert \boldsymbol{f}^{(t)} \rvert\rvert_2$
slope from max force	$\nabla f = \frac{\hat{f}^{(N)} - \max\left(\hat{f}^{(1)},\dots,\hat{f}^{(N)}\right)}{N - \underset{n}{\operatorname{argmax}}\left\{\hat{f}^{(n)}\right\}}$
contact duration	N
contact distance	$d = \lVert \boldsymbol{p}^{(N)} - \boldsymbol{p}^{(1)} \rVert$
force–torque correlation factor	Coefficient of determination R^2, obtained from a linear regression that fits $f : \varrho \longrightarrow \boldsymbol{f}$.

of 50 Hz since an SVM is known to be computationally very efficient.

Algorithm 2 Data-driven Segmentation & Recognition

Input: Measurements: $\boldsymbol{m}$, Set of *Classified Skills* (Fig. 2): CS
  $y_{\text{last}} \leftarrow$None
  $t_{\text{last}} \leftarrow 0$
  while $\boldsymbol{m}_t$ **do**
    $\boldsymbol{F}^{(t)} \leftarrow$ compute_features($[\boldsymbol{m}^{(t-W+1)}, \cdots, \boldsymbol{m}^{(t)}]$) ▷ Compute features over batch of measurements with length W
    $y =$ SVM.predict($\boldsymbol{F}^{(t)}$) ▷ Predict skill label
    if y_{last} is not y **then**
      $segments[t_{\text{last}} : t] \leftarrow y_{\text{last}}$
      $y_{\text{last}} \leftarrow y$
      $t_{\text{last}} \leftarrow t$
    end if
  end while
Output: *segments*

2.4. Combined recognition

This section describes the steps ②, ③, and ④ of the recognition pipeline in Fig. 3, where both previous recognition steps from (1a) and (1b) are merged into a common pipeline.

2.4.1. Construction of segments pool

The next stage of the skill recognition pipeline is shown in Fig. 3 ② and combines the output of both recognition sources. Here, a so-called segments pool is filled. The inputs are the segmentation lines and skill labels of both symbolic (1a) and data driven skill recognition (1b). The output is a list of segment candidates, which are exemplary shown under step ②. These candidates are constructed by the following three rules:

1. A segment that is intersected by a segmentation line of the other recognition source leads to the addition of both split parts as segment candidates. An example is the addition of the segment candidates in Fig. 3 labeled as s_1^1 and s_2^1.
2. If two segments from the same recognition source, which is either symbolic or data-driven, follow each other, e.g. *press* and *slide*, it leads to the addition of the concatenated segment. An example is given with the segment candidate in Fig. 3 labeled as s_1^2. This enables the recognition of skills that consume more time and are composed of multiple steps, such as *peg-in-hole*.
3. If a segment is intersected by the segmentation line of a *Gripper Skill*, it does not lead to the addition of new segment candidates and the overlapping segment candidate is removed from the pool. An example for excluded segments is given in Fig. 3, step ②, with the crossed out segment candidates, labeled as s_1^2 and s_1^3.

Rule (3) emerged since we consider the recognition of *Gripper Skills* to be highly reliable without uncertainty, and therefore do not create new segment candidates at regions that overlap with one of the *Gripper Skills*.

2.4.2. Data-driven skill recognition of complete interactions

As soon as the segment candidates are available in the *segments pool*, they can be classified by the data-driven approach. The procedure in Fig. 3, step ③ uses the same implementation as the one used in step (1b). However, it is operated with these differences:

1. The SVM is trained solely on complete examples of *Classified Skills* instead of examples observed in a time window of fixed length.
2. The SVM predicts skills based on features that were computed from the segments in the segments pool instead of features computed on a time window of fixed length.

Note that the segment candidates vary in their duration. However, computing the features over one segment always leads to a single feature vector $\boldsymbol{F}$ of fixed length. This means that each feature is designed to produce a scalar when computed on a time series of arbitrary length. See Table 1 for some examples.

From the segments pool, the final result is found by a greedy search. Here, each segment obtains a score for each possible skill class using the results of the SVM's decision function, given as

$$D_{\text{SVM}} : \mathbb{R}^{N_{\text{F}}} \mapsto \mathbb{R}^6 \quad \text{with } D_{\text{SVM}}(\boldsymbol{F}) = \boldsymbol{z}, \tag{1}$$

which maps the feature vector $\boldsymbol{F}$ to a vector $\boldsymbol{z}$ that holds the prediction scores for each of the six possible *Classified Skills*, where N_{F} is the number of dimensions in the feature vector $\boldsymbol{F}$. The skill with highest accuracy is extracted by

$$\overset{*}{s} = \underset{s}{\operatorname{argmax}}\{\boldsymbol{z}\}, \tag{2}$$

given a feature vector $\boldsymbol{F}_n$ for a segment s.

The skill with the highest score is appended to the task representation. All other segments that overlap this segment are removed from the *segments pool*. This is repeated until the whole demonstration has been segmented. Thus, the *move* and *Contact Skills* are found.

2.4.3. Finalization with gripper skills

In Fig. 3, step ④, the *Gripper Skills* are added to the task representation. They are directly taken from the symbolic recognition results and overwrite what is existing in the task representation. Exemplary, the input of this step is labeled as `<in>`, coming from the data-driven results of step ③. After overwriting the input with the *Gripper Skills*, the output can be found labeled as `<out>`.

In a final post-processing step, all segments that are below a predefined minimum length are split in the middle and the resulting parts are merged into the segments before and after, to reduce over-segmentation.

2.5. Extension to online segmentation

The previously described method of combining the symbolic and data-driven recognition requires the demonstration to be

completed, i.e. fully observed by the system. This is a requirement to run the approach offline. To overcome this, we propose an online segmentation procedure in a two step manner. The pseudo-code can be found in Algorithm 3.

Algorithm 3 Online Segmentation

```
Input: Measurements: m, Skill Classes: skills
  while m do
    s ← get_symbolic_segmentation_results(m)
    if s.state == DONE and s.type == contact skill then
      get_data_driven_segmentation_results()
    else
      show_preliminary_segmentation()
    end if
    if s.type == gripper skill then
      do_combined_segmentation(s)
    end if
    skill_sequence.append(s)
    show_combined_segmentation()
  end while
Output: skill_sequence
```

In the first step, the symbolic recognition detects online which skill is currently performed. When it detects a *contact* skill, the SVMs of the data-driven approach predict the most likely skills as specified in the *Contact Skills*. The detected skills are used to build the task representation online. An example can be seen in Fig. 4, first row. Once the symbolic skill segmentation detects the end of a *Gripper Skill*, such as *pick* or *place*, an intermediate fixed segmentation point is set, since it will not change during the rest of the recognition process. Therefore, the part of the demonstration that ends with this point can already be treated as a complete demonstration part, and thus be processed with the combined recognition algorithm. The results of the combined recognition then replace the preliminary recognition in the task representation (Fig. 4, middle row). This lets the representation constantly evolve, where past skills of the demonstrations are already finalized while the most recent skills are displayed as preliminary result. This process continues until the demonstration is finished and the whole task representation has been processed with the combined recognition algorithm (Fig. 4, bottom row).

2.6. Task representation as feedback

Due to the nature of the online skill recognition, the user is able to receive an immediate feedback about what the robot has already learned during the demonstration. This task representation is constantly updated as the demonstration progresses and the recognition results change from the preliminary results to the final results. The confidence of the online recognition process can be indicated by a color for each skill, which is exemplary shown in Fig. 4. Here, the robot continually provides a visual task representation where the system's confidence is increasing during the recognition progress.

3. Experiments

The framework is first assessed offline using a dataset of demonstrations (Section 3.1). Next, its segmentation and skill recognition performance is compared to only symbolic and only data-driven approaches using an exemplary task demonstration (Section 3.2). Table 2 shows all parameter values that were used throughout the experiments. The SVM parameters were chosen empirically by analyzing preliminary experiments. We suggest that such parameters should be subject to future optimization, e.g. by a grid search combined with cross validation.

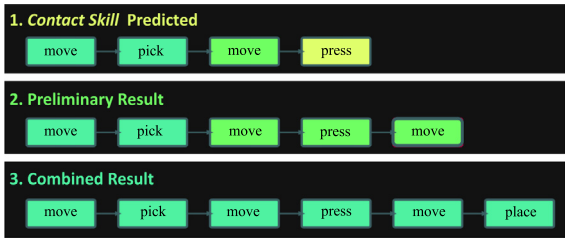


Fig. 4. Consolidation of task representation during online kinesthetic teaching. The colors allow the user to monitor to which extent each skill is consolidated, i.e. which step of the online recognition it has already passed. Yellow: SVMs currently predicting the label. Green: the preliminary recognition is done. Turquoise: the combined recognition is finished. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 2
Parameter values used in the experiments.

Parameter name	Value
sampling rate	50 Hz
SVM feature window length W	0.4 s
SVM regularization parameter C	100
SVM RBF kernel coefficient gamma $= \frac{1}{2\sigma^2}$	0.01
minimum skill length before merging	0.4 s

3.1. Skill recognition performance

The recognition algorithm was trained and evaluated on a dataset based on demonstrations of three people. In the dataset, approximately 100 demonstrations were provided per each skill. The dataset was recorded in batches of skill demonstrations, meaning that each user demonstrated e.g. 10 times the *press* skill, then 10 times the *slide* skill and so forth. The skills were separated by demonstrating a free motion without contact. This process was repeated until each skill was recorded for about 100 times.

The recognition algorithm was tested in a five-fold cross-validation experiment, using 80% of the data-set for training and the remaining 20% for testing. The ground truth was specified by annotating the skills manually that were used in the demonstration. The evaluation is based on two commonly used metrics [47], which are temporal tolerance and classification by data point label. The temporal tolerance indicates how close the algorithmic segmentation lines lie to the ground truth, without considering the labels. All algorithmic segmentation lines that lie in a region of $\pm t_{\text{err}}$ around the ground truth are considered as True Positive (TP), while all algorithmic segmentation lines that lie outside of such a region are considered as False Positive (FP). If a ground truth segmentation line has no corresponding algorithmic segmentation line within its region, it is considered as False Negative (FN). The region margin $\pm t_{\text{err}}$ was empirically chosen as 0.2 s in this evaluation. The overall accuracy is computed as F1-score as

$$F1 = \frac{2 \cdot TP}{2 \cdot TP + FN + FP}. \tag{3}$$

Our framework has an F1 score of 0.93, which means that 93% of the found segmentation lines lie within a region of ± 0.2 s around the ground truth. The accuracy of the segmentation lines matters, as even if a skill has been mislabeled, as long the segmentation lines are correct, the user could correct the label of the skill, without having to further adapt the segmentation result.

The second metric evaluates the classification accuracy by data point label, denoted as

$$C = N_{\text{correct\_labels}} / N_{\text{all\_labels}}. \tag{4}$$

It represents the ratio of how many of the data point labels l have been chosen correctly by the algorithm. From this, a confusion

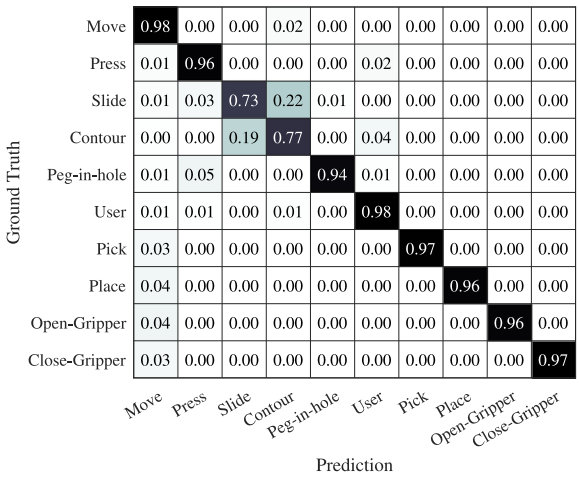


Fig. 5. Confusion matrix for the classification by data point label.

matrix can be calculated showing what percentage of ground truth labels l_g have been labeled with which label l_a by the algorithm. The confusion matrix of our approach is shown in Fig. 5. The average accuracy is 92.2%, which means that such percentage of data points is correctly labeled. Most confusion exists between the *contour* skill that is used to follow the outline of an object, and the *slide* skill that is used to move over the planar surface of an object while maintaining contact [23]. This might be caused by similar interaction principles in the user's demonstration.

3.2. Comparison of approaches

We compare the results of the purely symbolic approach from stage (1a), the data-driven approach from stage (1b), and our proposed approach from stage ④. As exemplary data, the experimenter demonstrated numerous skills in a toy environment. A part of this environment is shown in Fig. 6 (left), which was also used for demonstrating a *peg-in-hole* skill Fig. 6 (right). The segmentation results are shown in Fig. 7. The top row of this figure shows the symbolic recognition results, where the *Gripper Skills* were correctly recognized but the areas of contact just lead to an abstract *contact* skill. This representation is missing the specific skill type of the *Contact Skills* and it does not resolve multiple sequenced skills within one *contact* block, for example in the contact area starting at time = 25 s. The middle row shows the data-driven approach and how it oversegments the demonstration. It produces numerous artifacts in the form of very short skills. For example, right after time = 10 s, the skills of *contour*, *slide*, and *peg-in-hole* were recognized sequentially, although the demonstration contained only a single *peg-in-hole* skill in this time-frame. This problem can be explained by the fact that a *peg-in-hole* could itself consist of a sequence of other skills. Such skills are then favored by the greedy search, given that only a smaller time-frame is considered. Our approach is shown in the bottom row. Since we employ a segments pool as depicted in Fig. 3, the data-driven algorithm is also able to classify combined sequences of skills and finds the most likely candidate by the greedy search. This allows to handle situations where the observed skills significantly differ in their duration. Summarized, our algorithm handles the oversegmentation problem of the data-driven approach and recognizes the full skill palette that we introduced in the beginning.

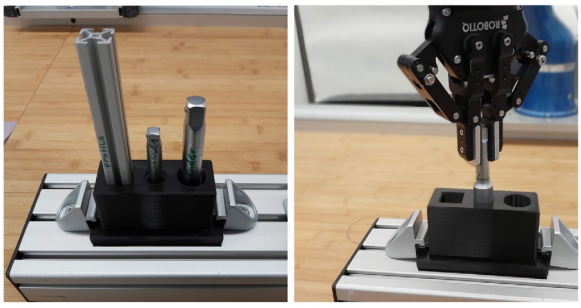


Fig. 6. Left: Three different constellations for a *peg-in-hole* skill with a square, small round, and large round peg. The left and right constellation are contained in the dataset for SVM training. Right: exemplary peg-in-hole task as contained in the segmentation analysis in Fig. 7.

3.3. Discussion

For the assessment of the presented framework, we evaluated the skill recognition performance based on a dataset (Section 3.1) and by an exemplary demonstration, where we compared a symbolic, a data-driven and our approach. Our method uses only proprioceptive sensor values and does not require recognizing or tracking objects in the environment to build a task representation. This enables the skill recognition to be trained on one set of objects and to be deployed in different setups with unknown objects. However, changes in the scene would not be recognized by the system due to the missing visual input.

One limitation of the system is the robustness towards demonstration speed in combination with the minimum allowed skill duration. If a user demonstrates in a too fast pace and the overall duration of a specific skill's data segment lies below the minimum skill length (Table 2), the skill will not be recognized at all. This could be avoided by familiarizing the user with the system. Alternatively, the minimum skill length could be decreased, which would potentially lead to very small segments. In consequence, the user could make use of a GUI that helps to decide which of these skills shall be kept or merged into others. Our segmentation approach makes use of the combined features from $\boldsymbol{f}_{\text{add}}$ and $\boldsymbol{f}_{\text{tsfresh}}$. Preliminary experiments pointed us to this selection, but in a future work, an ablation study should be conducted to evaluate the importance of possible combinations of these two feature sets.

The online segmentation capability distinguishes our approach from purely data-driven approaches like [28–30,32]. By combining symbolic with data-driven segmentation based on SVMs, we are able to efficiently evaluate the segmentation result as described in Section 2 at every time step during the user demonstration. The methods [29,32] employ a sampling-based inference approach, which cannot be evaluated online. In [30], the segmentation result of different demonstrations is forwarded to a classification task, which also renders it unsuitable for online segmentation. Furthermore, our method can segment a single task demonstration, whereas the approaches [28–31] require multiple demonstrations of the same task. The approaches based on a BP–AR–HMM for segmentation [29,31] use the variation of skill endpoints from different demonstrations to detect the relevant coordinate frame of every skill. Similarly, [28] analyzes skill endpoints of different demonstrations to obtain transition states for segmentation. Our approach uses a predefined skill library to avoid the need for multiple user demonstrations. This may limit the flexibility of detecting unknown skills, but we expect that in a production environment, reusable domain specific skills are usually known beforehand and are sufficient to handle most situations.

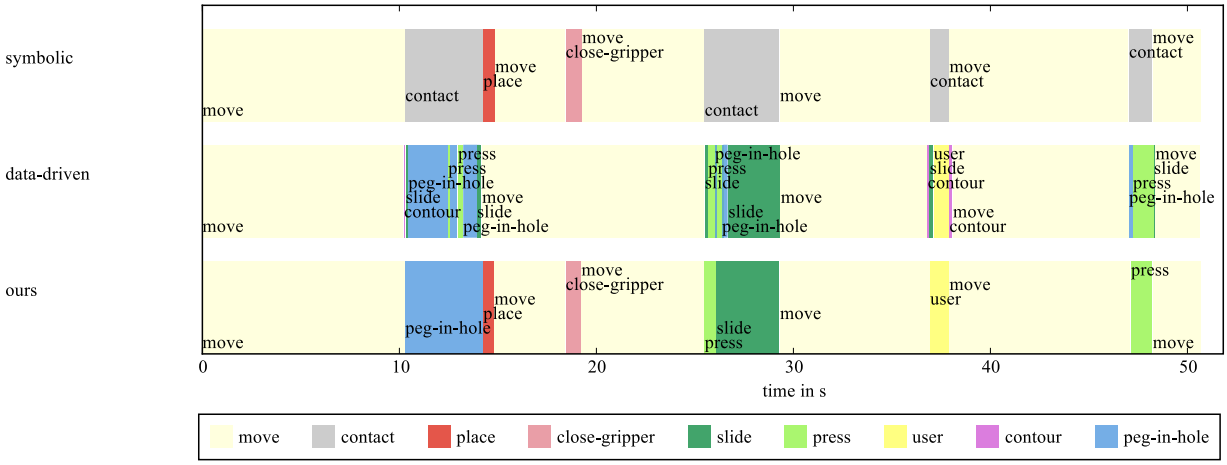


Fig. 7. Comparison of approaches.

4. User study

The user study in this section has the following aims: (1) assess the segmentation accuracy and correctness of the recognized skill sequence in a laboratory study with six subjects; (2) evaluate if users are able to successfully teach robots and are able to debug incorrect robot programs with the help of the task representation that our framework automatically assembles from the recognized skills.

All participants have some technical background in different fields of robotics, but they have neither used our framework nor have they participated in the data collection for our training set.

4.1. Procedure of the laboratory study

The laboratory study consists of a teaching phase (points 1 . . . 5) and a debugging phase (points 6 . . . 9) corresponding to the following procedure.

1. Experimenter shows instruct. video for method A
2. Experimenter shows instruct. video for task 1
3. Subject demonstrates task 1 with method A
4. Experimenter shows instruction video for task 2
5. Subject demonstrates task 2 with method A
6. Experimenter explains that a task can be debugged given a task representation of method A
7. Experimenter asks subject for expected task outcome of debug task $D(x, y)$ and lets subject correct it if desired using method A
8. Experimenter asks subject again for expected task outcome and lets subject correct if desired
9. Subject observes the task execution according to the current task representation
10. Subject evaluates overall framework by questionnaire concerning method A
11. Subject evaluates comparison between method A and B by questionnaire

Task 1 and 2 refer to the stamping task and assembly task respectively. Method A and B refer to the skill-based and time-line representation, which is further detailed in Section 4.4. The above steps from 1 to 10 are then repeated but with method B instead of method A. Throughout the whole study, the sequential order of tasks $\in \{1, 2\}$, debug tasks $D(x, y) \in \{D(1,1), D(1,2), D(2,1), D(2,2)\}$, and methods $\in \{A, B\}$ were permuted by a Latin Square design [48]. Before the experiment, the participant gets some time to familiarize with the robot in gravity compensation control.

In the teaching phase (points 1 . . . 5), the participant programs two robot tasks with two different forms of visual feedback, leading to four programming sequences. The participant is informed that after each demonstration, the robot will replay it to also record the robot's repetition of the task. For this, we used a Cartesian impedance controller. From the user demonstration and robot repetition data, we construct a Gaussian Mixture Model for each skill. Gaussian Mixture Regression is then applied on this model to obtain a trajectory for task execution. More details about this procedure are described in [49].

In the debug phase (points 6 . . . 9), the participant examines previously programmed task representations that were demonstrated by the experimenter either in a correct or intentionally wrong manner. These debug tasks use the same environment of the stamping and assembly task. This leads to four different debug task conditions with associated task representations, which are: D(1,1): stamping task — correct; D(1,2): stamping task — wrong; D(2,1): assembly task — correct; D(2,2): assembly task — wrong. The incorrect tasks showed a wrong skill sequence with respect to the correct skill sequence of the stamping and assembly tasks. The subject's goal is to validate if those task representations correctly perform the tasks. To see if the task representation alone offers enough information, the users are asked to give their opinion on the correctness before they are allowed to watch the robot execute the program. If they find a task representation to be incorrect, they are asked to correct it. Here, the users get the chance to correct a faulty robot program by selecting the skill which appeared to be faulty in the task representation. Next, the robot moves to the correct start location. Finally, the user can give a new demonstration that is subsequently segmented, resulting in a new, updated task representation.

4.2. Skill recognition for online task representation

The main purpose of our system is to build a task representation in the form of robot skills, while the user teaches the robot. Therefore, we requested two different users to program a DLR LWR IV [50] robot with the help of our framework. The robot was equipped with a Robotiq 85 two-finger gripper that is attached to a wrist-mounted FT-sensor, measuring the wrench that acts on the gripper. This setup can be seen in Fig. 8. The user transferred the task by kinesthetic teaching and used a button to open and close the gripper. We designed two small programming tasks as described in the following.

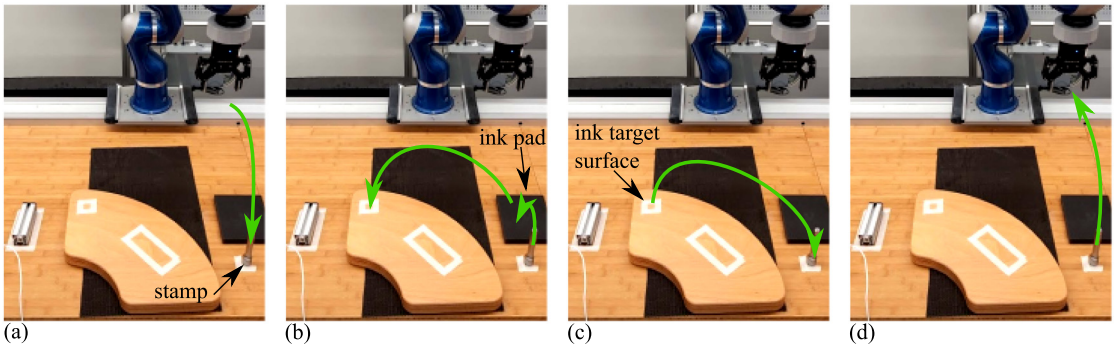


Fig. 8. Stamping task instructions. (a) pick the stamp, (b) push it into the ink pad and then, push it into the ink target surface, (c) place it at its original location, and (d) move gripper up to the home position. The task demonstration is shown in the accompanying video for the stamping task.

4.2.1. Stamping task

The visual task description for the stamping task is shown in Fig. 8. The textual instructions are provided in the figure's caption. Here, we requested from the user to demonstrate a task that involves two force-based interactions with the environment, one when pressing the stamp into the ink pad, and another one when pressing the stamp onto the ink target surface. Both times, we expect the system to detect a *press* skill.

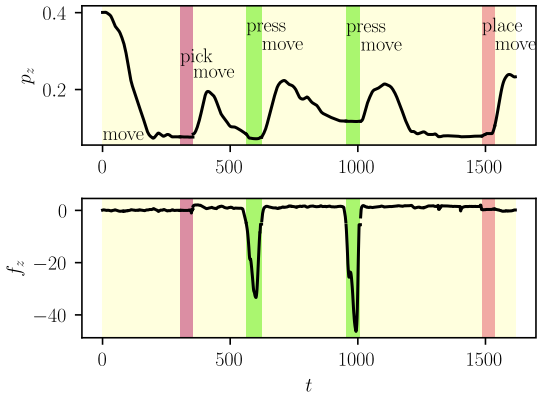
We evaluated how both users programmed the robot based on the task description. We show the skill recognition results alongside the gripper position and applied force in the z-axis respectively since these are the dimensions that represent the task best. The results are shown in Fig. 9 for both users, who achieved a segmentation accuracy of 96.9% (f1-score = 0.75) and 98.2% (f1-score = 1) respectively. We can see that the robot moved to the stamp (*move*), picked it (*pick*), moved to the ink pad (*move*), pressed on it (*press*), moved to the ink target surface (*move*), pressed on it (*press*), moved to the stamp's place location (*place*), placed the stamp (*place*), and moved up to the robot's home pose (*move*). Both force-based interactions were recognized as *press* skills, which can be seen by the regions of large force magnitude in the bottom plot of each user.

4.2.2. Assembly task

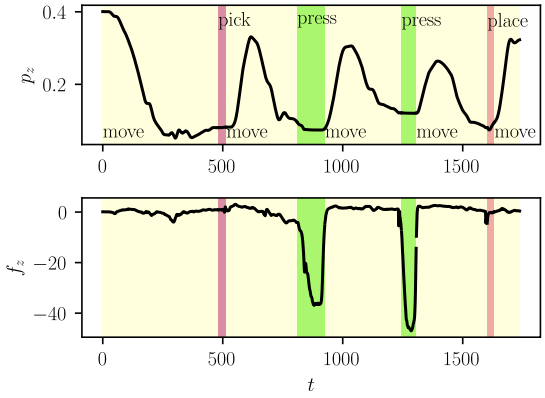
We designed an assembly task, where the users were asked to apply glue to a surface and then mount an object on it. The visual task description can be seen in Fig. 10, where the figure's caption contains the textual task instructions.

Again, we represent the end effector position and force in the z-axis and the task segmentation and skill recognition results in Fig. 11 for subject 1 and 5. For subject 1, we see a validly programmed task, where the glue stick is picked (*pick*), then sled over the glue contact surface (*slide*), and placed back at the original location (*place*). Next, the object is picked (*pick*) and pressed on the glue contact surface (*press*). While still being in contact with the environment, the gripper is opened, leading to the recognition of a *place* skill. Finally, the robot is moved to its home pose (*move*). We observe that the user decided to press on the object while it is still grasped, which meets the task description to assemble the object by applying pressure onto the glue bonding. Considering the strategy of subject 1, we can see that the skill recognition seamlessly detected the transition from *press* to *place* without an intermediate *move* (Fig. 11(a)). We emphasize that the presented framework is able to recognize transitions from forceful interactions to gripper actions, where the contact break with the environment is caused by the gripper opening itself.

Subject 5 followed the strategy of subject 1 until the picking of the object. Instead of using the sequenced (*press*) and (*place*) strategy, subject 5 placed the object without applying significant force (*place*), then moved the gripper above the object (*move*), closed the gripper without object, (*close-gripper*), moved down to the object (*move*), pressed onto the object with the empty and closed gripper (*press*), and finally moved the end effector to the home position (*move*). Although the strategy differed compared to subject 1, both users were able to achieve the task goal, which is mounting the object on the object target by applying pressure onto the glue bonding. Our method correctly detected the skills which were performed by the two users.



(a) Subject 1



(b) Subject 5

Fig. 9. Stamping task results for subject 1 and 5. p_z shows the position in z-axis and f_z the force in z-axis.

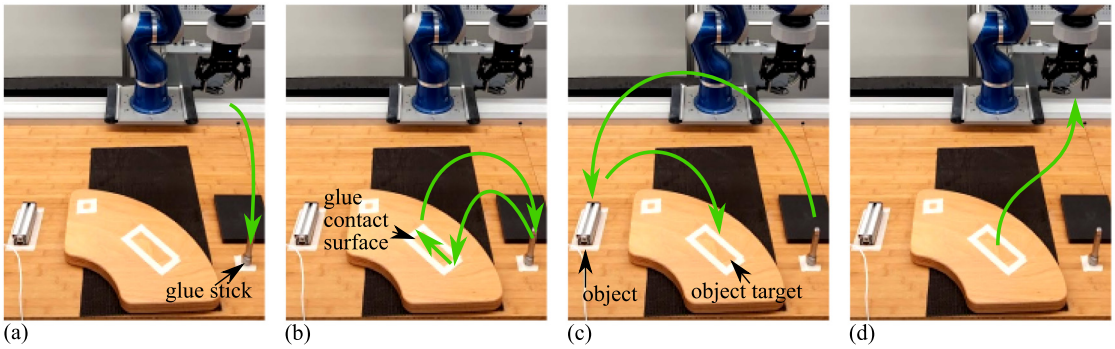
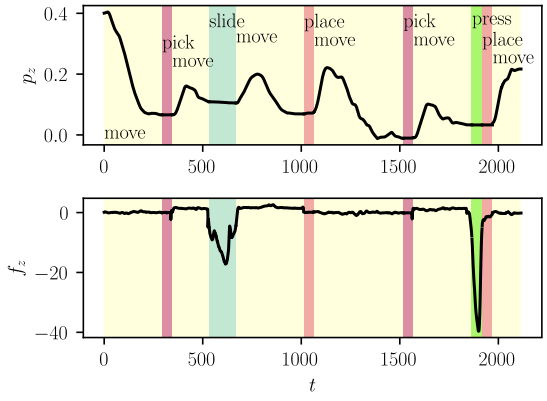
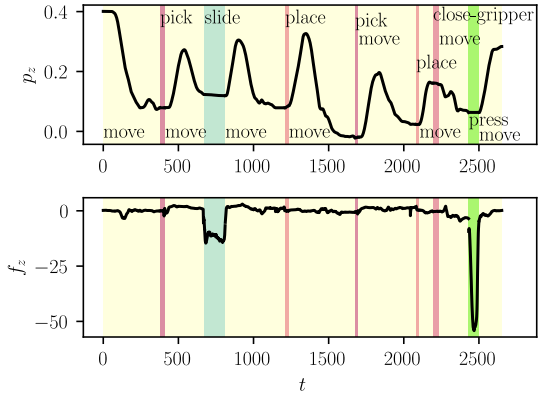


Fig. 10. Assembly task instructions. (a) pick the glue stick, (b) slide it over the contact surface and place it back at its original location, (c) pick the object and press it onto the object target. (d) move gripper up to the home position. The task demonstration is shown in the accompanying video for the assembly task.



(a) Subject 1



(b) Subject 5

Fig. 11. Assembly task results for subject 1 and 5. p_z shows the position in z-axis and f_z the force in z-axis.

4.3. Performance metrics

All participants were given the same video instructions about how to program the stamping and assembly task, which is also represented in Figs. 8 and 10. Regarding the assembly task, there was no specific gluing strategy required, namely the sliding action could be performed as desired by the user. For instance, some users decided for a single and possibly curved sliding interaction and others decided for multiple sliding interactions with the glue contact surface.

Table 3 shows the task representation results of the stamping and assembly task for each of the subjects numbered from one to six. We also state the ground truth (GT) for a successful skill sequence, as it is instructed to the participants.

Even though the initial instruction video suggests a skill sequence that solves the task, other skill sequences that originate from the user's own strategy can still lead to a successful task execution. We marked successful executions in column "Succ". with ✓and unsuccessful ones with –. Special cases that are marked with (*) are referred to in the discussion Section 4.6. Table 4 summarizes the quantitative performance of the framework when confronted with the untrained subjects of this study. The metrics were already introduced in Section 3.1. The laboratory study achieved an F1-score of 0.87 and an accuracy of 96%.

4.4. Evaluation of task representation

The evaluation consists of two parts, namely the laboratory study, as described in Section 4.1, in which six participants program a robot using our framework, and a remote study in which 26 users were asked to evaluate the comprehensibility of our approach. There were different participants in each of the parts. We name the style of our task representation skill-based. As a baseline, we define a so-called time-line task representation that does not provide a skill annotation for each demonstration segment. It has been introduced in [51], and its segmentation approach uses the Douglas–Peucker line simplification algorithm to detect notable points of a trajectory, which are then used to encode the trajectory. As these points have no semantic meaning, the resulting task representation describes the learned task through a time-line, showing the detected points as well as gripper movements. Since we focus on the evaluation of the task representation, we used the same segmentation points found in our approach to construct the time-line task representation. The time-line task representation shows the segmentation result without skill annotation and only the openings and closings of the gripper are displayed. The same task, but in different task representations can be seen in Fig. 12. The *contact skills* used in the experiments were limited to *press* and *slide*, to decouple the classification results from the evaluation of the task representations. The time-line representation, like the skill-based representation, is updated online when a new task segment is found; it displays the current segment while the task is being replayed; and it allows the robot to be moved to a specific point or to delete points through the user interface, e.g., for debugging purposes.

After performing all steps for one of both task representations in the laboratory study, the users were asked to fill out questionnaires. The NASA-TLX was used to measure the workload and the ISO 9241-110 questionnaire for Interaction Principles was used to evaluate the quality of interaction between the robot's user interface and the human. To further investigate the explainability

Table 3
Task representation results.

Subject ID	Task representation	Succ.
stamping GT (as instructed)	*move*, *pick*, *move*, *press*, *move*, *press*, *move*, *place*, *move*	
1, 2, 3, 5	*move*, *pick*, *move*, *press*, *move*, *press*, *move*, *place*, *move*	✓
4	*move*, *pick*, *move*, *press*, *move*, *press*, *move*, *press*, *place*, *move*	✓*
6	*move*, *pick*, *move*, *press*, *move*, *place*, *move*	–
assembly GT (as instructed)	*move*, *pick*, *move*, *slide*, *move*, *place*, *move*, *pick*, *move*, *press*, *place*, *move*	
1	*move*, *pick*, *move*, *slide*, *move*, *place*, *move*, *pick*, *move*, *press*, *place*, *move*	✓
2	*move*, *pick*, *move*, *slide*, *move*, *press*, *move*, *place*, *move*, *pick*, *move*, *press*, *place*, *move*, *close_gripper*, *move*, *press*, *move*	✓**
3	*move*, *pick*, *move*, *slide*, *move*, *slide*, *move*, *place*, *move*, *pick*, *move*, *place*, *move*, *close_gripper*, *press*, *open_gripper*, *move*	✓**
4	*move*, *pick*, *move*, *slide*, *move*, *press*, *move*, *place*, *move*, *pick*, *move*, *place*, *move*, *press*, *move*	✓***
5	*move*, *pick*, *move*, *slide*, *move*, *place*, *move*, *pick*, *move*, *place*, *move*, *close_gripper*, *move*, *press*, *move*	✓**
6	*move*, *pick*, *move*, *slide*, *move*, *place*, *move*, *pick*, *move*, *place*, *move*	–

Table 4
Overall performance metrics.

	stamping	assembly	average
$F1$ score (see. Eq. (3))	0.90	0.84	**0.87**
C (accuracy, see. Eq. (4))	0.97	0.94	**0.96**

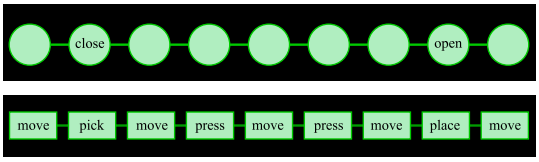


Fig. 12. Top: stamping task in the time-line representation; bottom: stamping task in the skill-based representation.

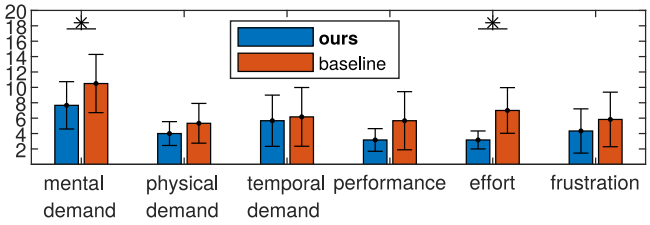


Fig. 13. The results of the NASA-TLX questionnaire answered by the participants of the laboratory study. The users score the categories on a scale from 1 (best) to 20 (worst). The plots show the mean of the scores given for the individual categories for the skill-based task representation used in our approach (blue) and the time-line task representation (red). The ⋆ marks that a p-value of $p < 0.05$ was found with a Wilcoxon signed rank test. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

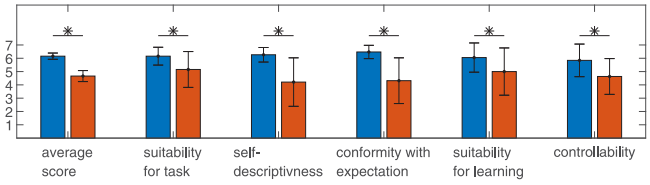


Fig. 14. The combined results of both the online and the laboratory study for the ISO questionnaire. The scores rate how much the users agree to the category from 1 (Strongly Disagree) to 7 (Strongly agree). For details about the ⋆ and colors please refer to Fig. 13.

of the task representations, based on [6], questions with regards to three more categories were used: trust in the robot, ease of debugging and match of the user's mental model to the real robot.

The remote study was designed as an extension of the laboratory study. It uses the same robotic task setting and considers only the debug phase. Since the subjects could not program a real robot, they were asked to match a task description presented as a video to a task representation. Instead of correcting the robot program, they labeled it only as unsuccessful.

The subjects were split into two equally sized groups, each only seeing one of the task representations. To evaluate the representations, the ISO 9241-110 questionnaire and the explainability questions were given.

4.5. Subjective results

A number of subjective results were collected throughout the user study with the help of questionnaires. The perceived workload when using the approaches with different task representations can be seen in Fig. 13. A clear reduction of the mental workload and effort is visible for the use of the skill-based representation in comparison to the time-line representation. The intuitiveness and usability of the representations is compared in Fig. 14, where the results of the ISO 9241-110 questionnaire can be seen. On average, the users rated the usability of the skill-based task representation 32% better than the time-line representation. Especially in the categories evaluating the intuitiveness of the representation, the skill-based task representation drastically outperformed the time-line task representation: The self-descriptiveness of the skill-based task representation was rated 48% better and the conformity with the user's expectation 50% better.

To investigate further in which aspects of the teaching procedure the representation of the robot's knowledge influences the teacher, the results shown in Fig. 15 are grouped in the aforementioned categories as proposed in [6]. In the debugging category, the users were asked to rate the ability to detect errors in the robot's learned program with the help of the task representation. From the results, a clear preference for the skill-based task representation can be seen. The users' self evaluation is also confirmed by the number of times the users were able to predict if a task representation would perform a task as desired, for which the results can be seen in Fig. 16. It shows that, with the help of the skill-based task representation, 84% of the participants were able to identify that the visualized task representation would not match the intended task goal. In contrast, only 64% could do so with the time-line task representation. In the categories *match of mental model* and *explainability*, the skill-based task representation also outperformed the time-line task representation drastically (see Fig. 15). The influence of the task representation was also noticeable in the laboratory study. Most participants caused the detection of a *press* skill before the placing of objects, as they would forcefully set them on the workspace. In

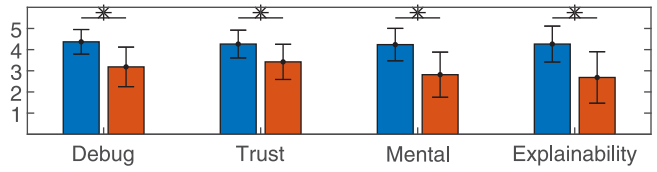


Fig. 15. The combined results of both the online and the laboratory study for the explainability questions. For details about the ⋆ and colors please refer to Fig. 13. The questions were grouped in the categories 'Ease of debugging', 'Trust in the robot', 'Match of the user's mental model to the robot's real knowledge' and 'Explainability of the task representation', shown here from left to right. The users rated on a 5-point Likert scale from 1 (Strongly Disagree) to 5 (Strongly Agree). The higher the result, the better.

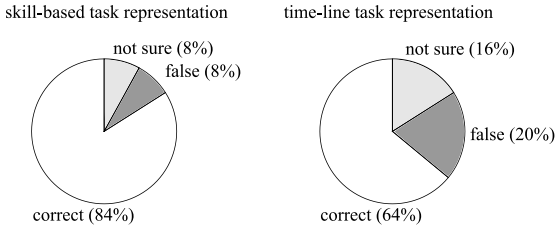


Fig. 16. The results of the debug tasks. The participants were asked whether a task representation would execute the task successfully. The diagrams show the percentage of answers where the user's predicted outcome matched the task outcome. The answers were evaluated to be correct, false, or undecided (not sure) for the skill-based task representation (left) and the time-line task representation (right).

the time-line representation, none of the users chose to correct this behavior, which could clearly be seen in the robot's execution of the task, while, in the skill-based task representation, 50% of the users removed this *press* skill, as the graph did not match their expectation of how the task should be executed.

The users also rated their trust in the robot higher (24%), even though the robot's execution of the taught programs was independent of the task representation. In a final oral interview in the laboratory study, the participants showed a clear preference for the skill-based task representation. When asked for their first impressions, half of the participants found the time-line representation easier to understand when seeing it for the first time, as it conveys less information compared to the skill-based representation. However, their preference shifted to the skill-based representation when asked which one they found easier to work with.

4.6. Discussion

The user study has shown that the strategy of different subjects might differ from each other for the same task. However, most subjects reached the task goals with their own strategy in each task. Variations between task description and user's teaching behavior are marked with the symbol * in Table 1. Table 3. For the stamping task, the symbol * denotes that one user pressed the tool twice on the ink target surface, which has been correctly recognized by the system, but was not required from the task description. For the assembly task, the symbol ** marks users that decided for another strategy, which involves closing of the gripper before applying pressure on the object. In this case, the skills were also correctly recognized by the system and the task was performed successfully. User 4 (***) demonstrated a skill sequence that involved two additional press skills caused by contacting the environment with the tool. This might be due to low expertise in kinesthetic teaching, but did not lead to unsuccessful task execution.

Considering the human and robot perspective, we see a number of future benefits from the obtained task representation. First, it contains the knowledge about the recognized skills, which can be used by the robotic system to efficiently reproduce them. Here, the robot can rely on a library of existing skill implementations to execute the skills using an optimized strategy. Such implementations are known from the state of the art, for instance considering a touch skill [38], a slide and contour skill [41,52], or a peg-in-hole skill [43]. Second, the user could adapt skill-related parameters given a graphical interface without demonstrating the task again. An example is to adapt the pressing force in the *press* skill. Third, the user could adapt the task representation itself without providing a new demonstration, for instance, exchanging skill types that fit better to the task requirements. An example is that the system identified a skill of type "slide" although the user wants to employ a skill of type "polish". With that, the associated data segment would be reused to parameterize another skill implementation. Fourth, the user can observe what the robot is currently performing, which helps the robot to explain itself and enables trust in the system.

The results of the user study show that a skill-based task representation outperforms a time-line task representation in regards to its explainability and usability. The users could find errors in the task representation at first glance for the skill-based representation, while the time-line representation forced users to go through every single step of the task and estimate how many steps this might correspond to in the time-line. When asked to correct a faulty program, the users chose to insert new demonstrations in the time-line task representation at points where the gripper opened or closed, as these were the easiest to detect, while the skill-based task representation allowed the users a more nuanced correction of the task representation. Furthermore, the users of the laboratory study showed a stronger indifference towards correcting their own taught programs when they were presented in the time-line task representation. Especially in the aforementioned case of the detected *press* skills before the *place* skills, only in the skill-based representation users closely inspected the task representation and noticed the necessity to correct this mistake. This stronger engagement with the task representation in the skill-based form is coupled to the users' stronger control over the robot's knowledge. Thus, the combination of a stronger incentive to engage with the taught program and a reduced workload makes the skill-based task representation more suited to be used by non-experts.

The accuracy of the skill recognition lies by 96% within the user study. Therefore, it can be argued that the proposed framework is robust enough to work with users that were never trained on the system before.

5. Conclusion

We introduced an online task segmentation and skill recognition algorithm that combines symbolic segmentation in the form of evaluating preconditions and postconditions and data-driven segmentation and recognition in the form of a classifier that predicts appropriate skills. This combination enables the detection of a wider range of skills, involving *Contact Skills*, while still allowing the algorithm to run online. The experimental results confirm the recognition capabilities and showcase how task representations emerge from kinesthetic user demonstrations. These task representations are consolidated on the fly, whenever the incoming data increases the confidence of the system. With that, the users receive immediate feedback about the currently processed skills and finally receive a consolidated task representation

about how the robot interpreted their demonstration. This makes the framework suitable for non-expert robot programmers by building trust in the robot and closing the gap between the user's mental model of the robot and the robot's actual knowledge.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Dongheui Lee reports financial support was provided by Helmholtz Association of German Research Centres.

Data availability

Data will be made available on request.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.robot.2023.104367. Video S1 introduces the proposed method in brief, followed by the two experimental tasks as part of the user study.

References

[1] A. Kumar, From mass customization to mass personalization: A strategic transformation, Int. J. Flexible Manuf. Syst. 19 (4) (2007) 533–547.

[2] S. Calinon, D. Lee, Learning control, in: P. Vadakkepat, A. Goswami (Eds.), Humanoid Robotics: A Reference, Springer, 2019, pp. 1261–1312.

[3] M. Cakmak, A.L. Thomaz, Eliciting good teaching from humans for machine learners, Artificial Intelligence 217 (2014) 198–215.

[4] N. Koenig, L. Takayama, M. Matarić, Communication and knowledge sharing in human–robot interaction and learning from demonstration, Neural Netw. 23 (8–9) (2010) 1104–1112.

[5] A. Sena, M. Howard, Quantifying teaching behavior in robot learning from demonstration, Int. J. Robot. Res. 39 (1) (2020) 54–72.

[6] R. Sheh, "Why did you do that?" Explainable intelligent robots, in: AAAI Workshop-Technical Report, 2017, pp. 628–634.

[7] M.R. Pedersen, L. Nalpantidis, R.S. Andersen, C. Schou, S. Bøgh, V. Krüger, O. Madsen, Robot skills for manufacturing: From concept to industrial deployment, Robot. Comput.-Integr. Manuf. 37 (2016) 282–291.

[8] D. Paulius, Y. Sun, A survey of knowledge representation in service robotics, Robot. Auton. Syst. 118 (2019) 13–30.

[9] M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, D. Wilkins, A. Barrett, D. Christianson, PDDL - The Planning Domain Definition Language, Tech. rep., Yale University, 1998.

[10] K. Ramirez-Amaro, Y. Yang, G. Cheng, A survey on semantic-based methods for the understanding of human movements, Robot. Auton. Syst. 119 (2019) 31–50, http://dx.doi.org/10.1016/j.robot.2019.05.013.

[11] K.R. Guerin, C. Lea, C. Paxton, G.D. Hager, A framework for end-user instruction of a robot assistant for manufacturing, in: IEEE International Conference on Robotics and Automation, ICRA, 2015, pp. 6167–6174.

[12] C. Paxton, A. Hundt, F. Jonathan, K. Guerin, G.D. Hager, CoSTAR: Instructing collaborative robots with behavior trees and vision, in: IEEE International Conference on Robotics and Automation, ICRA, 2017, pp. 564–571.

[13] M.N. Nicolescu, M.J. Mataric, Learning and interacting in human–robot domains, IEEE Trans. Syst. Man Cybern. 31 (5) (2001) 419–430.

[14] F. Steinmetz, V. Nitsch, F. Stulp, Intuitive task-level programming by demonstration through semantic skill recognition, IEEE Robot. Autom. Lett. 4 (4) (2019) 3742–3749.

[15] M. Wächter, S. Schulz, T. Asfour, E. Aksoy, F. Wörgötter, R. Dillmann, Action sequence reproduction based on automatic segmentation and object–action complexes, in: 13th IEEE-RAS International Conference on Humanoid Robots, Humanoids, 2013, pp. 189–195.

[16] X. Wang, Learning planning operators by observation and practice, in: Proceedings of the Second International Conference on Artificial Intelligence Planning Systems, AAAI, 1994, pp. 335–340.

[17] N. Abdo, H. Kretzschmar, L. Spinello, C. Stachniss, Learning manipulation actions from a few demonstrations, in: IEEE International Conference on Robotics and Automation, 2013, pp. 1268–1275, http://dx.doi.org/10.1109/ICRA.2013.6630734.

[18] M. Diehl, C. Paxton, K. Ramirez-Amaro, Automated generation of robotic planning domains from observations, 2021, arXiv preprint arXiv:2105.13604.

[19] D. Kulić, C. Ott, D. Lee, J. Ishikawa, Y. Nakamura, Incremental learning of full body motion primitives and their sequencing through human motion observation, Int. J. Robot. Res. 31 (3) (2012) 330–345.

[20] T. Eiband, M. Saveriano, D. Lee, Intuitive programming of conditional tasks by demonstration of multiple solutions, IEEE Robot. Autom. Lett. 4 (4) (2019) 4483–4490, http://dx.doi.org/10.1109/LRA.2019.2935381.

[21] I. Dianov, K. Ramirez-Amaro, P. Lanillos, E. Dean-Leon, F. Bergner, G. Cheng, Extracting general task structures to accelerate the learning of new tasks, in: IEEE-RAS 16th International Conference on Humanoid Robots, Humanoids, 2016, pp. 802–807.

[22] K. Ramirez-Amaro, M. Beetz, G. Cheng, Transferring skills to humanoid robots by extracting semantic representations from observations of human activities, Artificial Intelligence 247 (2017) 95–118.

[23] T. Eiband, D. Lee, Identification of common force-based robot skills from the human and robot perspective, in: IEEE-RAS 20th International Conference on Humanoid Robots, Humanoids, 2021, pp. 507–513.

[24] A.L. Pais, K. Umezawa, Y. Nakamura, A. Billard, Task parameterization using continuous constraints extracted from human demonstrations, IEEE Trans. Robot. 31 (6) (2015) 1458–1471.

[25] Z. Qiu, T. Eiband, S. Li, D. Lee, Hand pose-based task learning from visual observations with semantic skill extraction, in: 29th IEEE International Conference on Robot and Human Interactive Communication, RO-MAN, 2020, pp. 596–603.

[26] Y. Wang, Y. Jiao, R. Xiong, H. Yu, J. Zhang, Y. Liu, MASD: A multimodal assembly skill decoding system for robot programming by demonstration, IEEE Trans. Autom. Sci. Eng. 15 (4) (2018) 1722–1734.

[27] Z. Su, O. Kroemer, G.E. Loeb, G.S. Sukhatme, S. Schaal, Learning manipulation graphs from demonstrations using multimodal sensory signals, in: IEEE International Conference on Robotics and Automation, ICRA, 2018, pp. 2758–2765.

[28] S. Krishnan, A. Garg, S. Patil, C. Lea, G. Hager, P. Abbeel, K. Goldberg, Transition state clustering: Unsupervised surgical trajectory segmentation for robot learning, Int. J. Robot. Res. 36 (13–14) (2017) 1595–1618, http://dx.doi.org/10.1177/0278364917743319.

[29] S. Niekum, S. Osentoski, G. Konidaris, A.G. Barto, Learning and generalization of complex tasks from unstructured demonstrations, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2012, pp. 5239–5246.

[30] E.C. Grigore, B. Scassellati, Discovering action primitive granularity from human motion for human-robot collaboration, in: Robotics: Science and Systems, 2017, http://dx.doi.org/10.15607/RSS.2017.XIII.065.

[31] M. Chi, Y. Yao, Y. Liu, Y. Teng, M. Zhong, Learning motion primitives from demonstration, Adv. Mech. Eng. 9 (12) (2017) 1687814017737260, http://dx.doi.org/10.1177/1687814017737260.

[32] C. Willibald, D. Lee, Multi-level task learning based on intention and constraint inference for autonomous robotic manipulation, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2022.

[33] G. Konidaris, L.P. Kaelbling, T. Lozano-Perez, From skills to symbols: Learning symbolic representations for abstract high-level planning, J. Artificial Intelligence Res. 61 (2018) 215–289.

[34] D.A. Winter, A.J. Fuglevand, S.E. Archer, Crosstalk in surface electromyography: Theoretical and practical estimates, J. Electromyography Kinesiol. 4 (1) (1994) 15–26, http://dx.doi.org/10.1016/1050-6411(94)90023-X.

[35] S. Ghosh, A. Dasgupta, A. Swetapadma, A study on support vector machine based linear and non-linear pattern classification, in: International Conference on Intelligent Sustainable Systems, ICISS, 2019, pp. 24–28, http://dx.doi.org/10.1109/ISS1.2019.8908018.

[36] A. Stolt, M. Linderoth, A. Robertsson, R. Johansson, Force controlled robotic assembly without a force sensor, in: IEEE International Conference on Robotics and Automation, 2012, pp. 1538–1543.

[37] A. Stolt, M. Linderoth, A. Robertsson, R. Johansson, Detection of contact force transients in robotic assembly, in: IEEE International Conference on Robotics and Automation, ICRA, 2015, pp. 962–968.

[38] T. Eiband, M. Saveriano, D. Lee, Learning haptic exploration schemes for adaptive task execution, in: IEEE International Conference on Robotics and Automation, ICRA, 2019, pp. 7048–7054.

[39] S. Stelter, G. Bartels, M. Beetz, Multidimensional time-series shapelets reliably detect and classify contact events in force measurements of wiping actions, IEEE Robot. Autom. Lett. 3 (1) (2018) 320–327.

[40] A. Montebelli, F. Steinmetz, V. Kyrki, On handing down our tools to robots: Single-phase kinesthetic teaching for dynamic in-contact tasks, in: IEEE International Conference on Robotics and Automation, ICRA, 2015, pp. 5628–5634.

[41] V.K. Origanti, T. Eiband, D. Lee, Automatic parameterization of motion and force controlled robot skills, in: 9th International Conference on Robot Intelligence Technology and Applications, RiTA, Springer International Publishing, 2021, pp. 66–78.

[42] S.-C. Chen, P.-C. Tung, Trajectory planning for automated robotic deburring on an unknown contour, Int. J. Mach. Tools Manuf. 40 (7) (2000) 957–978.

[43] 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.
[44] J. Xu, Z. Hou, Z. Liu, H. Qiao, Compare contact model-based control and contact model-free learning: A survey of robotic peg-in-hole assembly strategies, 2019, arXiv:1904.05240.
[45] S. Haddadin, M. Suppa, S. Fuchs, T. Bodenmüller, A. Albu-Schäffer, G. Hirzinger, Towards the robotic co-worker, in: Robotics Research, Springer, 2011, pp. 261–282.
[46] M. Christ, N. Braun, J. Neuffer, A.W. Kempa-Liehr, Time series feature extraction on basis of scalable hypothesis tests (tsfresh–A python package), Neurocomputing 307 (2018) 72–77.
[47] J.F.-S. Lin, M. Karg, D. Kulić, Movement primitive segmentation for human motion modeling: A framework for analysis, IEEE Trans. Hum.-Mach. Syst. 46 (3) (2016) 325–339.
[48] D.A. Grant, The latin square principle in the design and analysis of psychological experiments, Psychol. Bull. 45 (5) (1948) 427.
[49] T. Eiband, C. Willibald, I. Tannert, B. Weber, D. Lee, Collaborative programming of robotic task decisions and recovery behaviors, Auton. Robots (2022) 1–19.
[50] A. Albu-Schäffer, S. Haddadin, C. Ott, A. Stemmer, T. Wimböck, G. Hirzinger, The DLR lightweight robot: Design and control concepts for robots in human environments, Ind. Robot (2007).
[51] I. Iturrate, E.H. Østergaard, M. Rytter, T.R. Savarimuthu, Learning and correcting robot trajectory keypoints from a single demonstration, in: 3rd International Conference on Control, Automation and Robotics, ICCAR, IEEE, 2017, pp. 52–59.
[52] X. Gao, J. Ling, X. Xiao, M. Li, Learning force-relevant skills from human demonstration, Complexity (2019).

Thomas Eiband is a Ph.D. candidate at the Institute of Robotics and Mechatronics of the German Aerospace Center (DLR), Wessling, Germany. He obtained his Master's Degree in Electrical Engineering and Information Science from the Technical University of Munich in 2017. His main research interests are robot skill learning and intuitive programming approaches with a focus on force and contact-based robot skills. His main work is on approaches for recognition and parameterization of robot skills from human demonstrations.

Johanna Liebl obtained her master's degree in Electrical Engineering at the Human-Centered Assistive Robotics department at the TU München.

Christoph Willibald received his B.Sc. in Mechanical Engineering and his M.Sc. in Mechatronics at the Technical University of Munich (TUM), Germany, in 2016 and 2020, respectively. He joined the Institute of Robotics and Mechatronics at the German Aerospace Center (DLR) in Weßling, Germany, for his Master's Thesis in 2019. Since 2020 he is a Research Scientist at the Institute of Robotics and Mechatronics at DLR and a Ph.D. Student at the Human-centered Assistive Robotics group at TUM. His research interests include learning from demonstration, cognitive robotics and human-robot interaction.

Dongheui Lee is Associate Professor at the Department of Electrical and Computer Engineering, Technical University of Munich (TUM). She is also leading the Human-centered assistive robotics group at the German Aerospace Center (DLR). Her research interests include human motion understanding, human robot interaction, machine learning in robotics, and assistive robotics. She obtained her B.S. (2001) and M.S. (2003) degrees in mechanical engineering at Kyung Hee University, Korea and a PhD degree from the department of Mechano-Informatics, University of Tokyo, Japan in 2007. She was a research scientist at the Korea Institute of Science and Technology (KIST) (2001-2004), Project Assistant Professor at the University of Tokyo (2007-2009) and joined TUM as professor. She was awarded a Carl von Linde Fellowship at the TUM Institute for Advanced Study (2011) and a Helmholtz professorship prize (2015).