DLR-IB-RM-OP-2021-51

Intuitive Knowledge Representations for Interactive Robot Programming

Masterarbeit

Johanna Liebl

Deutsches Zentrum
DLR für Luft- und Raumfahrt

Intuitive Knowledge Representations for Interactive Robot Programming

handed in
MASTER'S THESIS

B.Sc. Johanna Liebl

born on the 19.08.1994
living in:
Caubstrasse 7
80993 Munich
Tel.: 015234567231

Human-centered Assistive Robotics
Technical University of Munich

Univ.-Prof. Dr.-Ing. Dongheui Lee

Supervisor:	M.Sc. Thomas Eiband
Start:	01.08.2020
Intermediate Report:	08.12.2020
Delivery:	15.04.2021

TECHNISCHE UNIVERSITÄT MÜNCHEN
Human-centered Assistive Robotics
UNIV.-PROF. DR.-ING. DONGHEUI LEE

April 6, 2021

M A S T E R ' S T H E S I S
for
Johanna Liebl
Student ID 03659970, Degree EI

Intuitive Knowledge Representations for Interactive Robot Programming

Problem description:

We are aiming towards an era, where robot programming is not only subject to experts but requires shop floor workers and people in daily life situations to program robots. Learning from Demonstration (LfD) has been proven as an appropriate method to do so but if task structures are getting more complex, the user might loose control of what the system has already been learned and what are its limitations. We can overcome this by employing active learning techniques where the system decides what information is missing. Additionally, we want to present the system's knowledge in an intuitive way to increase explainability [1], correctability and parameterization options for constraints that cannot be extracted from demonstration.
The goal of this project is to extend an existing LfD framework [2] with active learning techniques to speed up task learning and at the same time, provide an interactive communication channel that gives the user all necessary options to understand and control the task definition process.

Tasks:

- Literature survey on robot LfD, active learning and interactive robot programming
- Development of an active learning method for graph-based task structures with semantic annotation of task segments (skills) [3]
- Combination of different sensor modalities: motion, force, and optionally vision
- Design and implementation of an interactive feedback channel (GUI and possibly speech) that represents the task and allows correctability
- Experimental evaluation of system capabilities and human-robot interaction performance

Bibliography:

[1] Mark Edmonds, Feng Gao, Hangxin Liu, Xu Xie, Siyuan Qi, Brandon Rothrock, Yixin Zhu, Ying Nian Wu, Hongjing Lu, and Song-Chun Zhu. A tale of two explanations: Enhancing human trust by explaining robot behavior. *Science Robotics*, 4(37), 2019.
[2] T. Eiband, M. Saveriano, and D. Lee. Intuitive programming of conditional tasks by demonstration of multiple solutions. *IEEE Robotics and Automation Letters*, 4(4):4483–4490, Oct 2019.
[3] Scott Niekum, Sachin Chitta, Andrew G Barto, Bhaskara Marthi, and Sarah Osentoski. Incremental semantically grounded learning from demonstration. In *Robotics: Science and Systems*, volume 9. Berlin, Germany, 2013.

Supervisor:	M. Sc. Thomas Eiband
Start:	01.08.2020
Intermediate Report:	08.12.2020
Delivery:	15.04.2021

(D. Lee)
Univ.-Professor

Abstract

Programming by Demonstration (PbD) is an intuitive method to transfer knowledge from a non-expert human teacher to a robot. To allow the non-expert user to intuitively understand what the robot has learned from the demonstration, we propose a framework that detects online which skills the human is demonstrating and builds from that a graph that describes how the task is performed. The skill recognition is achieved by a segmentation algorithm that combines symbolic skill segmentation, which makes use of pre- and postconditions to identify skills, with data-driven segmentation, which uses Support Vector Machines to learn to classify the skills from data. The framework is thus able to detect force-based skills in addition to manipulation skills, to allow the flexible use of robots in assembly production lines. The intuitiveness of the framework is evaluated in a user study that compares the task graph representation of our framework to the time-line based representation of an existing PbD framework that does not make use of skill recognition.

Contents

Chapter 1

Introduction

The recent years have shown a shift in the production cycle from large product batches to small ones, as the demand for customized products grows. 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 thus becomes attractive in labour intense production chains. These two trends combined lead to the demand of robots that learn new tasks fast and efficiently, in order to keep up with the changing production cycle. But this stands in contrast to the traditional use of robots, where a robot expert implements code that the robot performs for long periods of time. This hinders fast changes, as the people working in the production often lack the background to implement these changes themselves, and experts must provide the code instead.
Thus teaching robots new tasks without having to implement code has become a much researched topic. This aims to give people without a background in robotics, such as shop-floor workers, the ability to teach robots, thus enabling them to execute small changes in the robots' programs themselves. To allow the use of robots in a wide variety of production line scenarios, the robots must be capable of learning a large range of manipulation actions from the human teacher. Common tasks for

Figure 1.1: A robot sanding a surface [Rob21].

Figure 1.2: A robot drilling a hole [ABB21].

robots, such as sanding a surface, as shown in Fig. 1.1, or drilling holes, as shown in Fig. 1.2, require the robot to perform actions in contact with its environment. Therefore, to enable the shop-floor workers to teach the robot these tasks, the robot must be able to identify the contact actions shown by the teacher, such as sliding over a surface or pressing against an object. This still remains a challenge, as it requires a detailed and diverse examination of the teacher's demonstration, to ensure that the robot extracts correctly how to perform the task itself.
Furthermore, while a large body of work focuses on the technical side of providing learning mechanics for the robots, only in recent years the focus also shifted on considering the humans who are expected to become teachers for the robots. Because regardless of how good the robot's learning mechanism is, if the human teacher is unable to provide the necessary demonstrations, the robot will fail to perform the task correctly. Therefore it is necessary to consider where challenges for the teacher lie, and instead of only focusing on the robot's learning capabilities, to provide the teacher with a complete framework focused on making the teaching experience as intuitive, effortless and enjoyable as possible.

1.1 Problem Statement

The aim of this thesis is to provide an interactive robot programming framework, that is focused on the teacher's needs, while ensuring that the robot is capable of learning how to perform a large variety of tasks. Therefore, the focus of this thesis is twofold: On the one hand, it lies on the robot's ability to identify which steps are necessary to perform a task, so that it is then able to execute the task. As this framework is designed to enable the use of robots in production lines, the robot must be able to learn a large selection of contact-based interactions with its environment, thus allowing it to flexibly operate tools and manipulate objects.
On the other hand, the teacher's role is also considered. As the intended users for this framework are non-robotic experts, the framework should enable them to be good and effective teachers, without requiring a steep learning curve. This requires an intuitive and comprehensible representation of the robot's knowledge, giving the users the necessary feedback to understand what the robot is learning. Therefore, this thesis aims to solve the following challenges:

- How can we represent a robot's knowledge such that:
  - the robot can learn a large variety of contact-based manipulation actions
  - the robot's knowledge is intuitively understandable for humans
  - the robot's knowledge can be corrected and expanded by the teacher in an interactive fashion
- How can we design a framework focused on the teachers needs, such that:

- it is intuitive to use
- it communicates the robot's knowledge correctly to the human teacher
- it provides the necessary feedback for the teachers to perform the teaching to the best of their abilities

The first points address how the robot's knowledge is encoded. There are many approaches in literature on how robots can learn, ranging from how this knowledge is extracted from demonstrations to how this knowledge is then used to perform the task. In this work, the focus lies on finding an approach that is flexible enough to teach a robot multiple options to manipulate objects, while allowing a representation of the robot's knowledge that is understandable to non-experts. We therefore choose to encode the robot's knowledge on a *skill* level. This means that a task is broken down into smaller, semantically meaningful steps, the so-called *skills*. These skills describe the purpose of the step, such as *Pick* up an object or *Press* with a tool on this surface. By describing the task on the skill level, a task-graph can be built depicting the sequence of skills used to perform this task. The teacher can then go through the steps of this task-graph to understand how the robot will execute the task. As this step-by-step representation is self-descriptive, it is intuitively understandable for non-experts.
Furthermore, by creating a broad selection of skills, the robot is enabled to learn how to operate tools and interact with the environment. Thus a skill level task encoding serves both to create a large and flexible knowledge base for the robot, as well as to communicate the robot's knowledge to the human.
Yet the challenge remains to transfer the user's demonstration into the skill level. While there is a large body of literature focusing on how to extract skills from demonstrations, addressed in Chap. 2, there is a lack of focus on skills used for contact-based manipulation purposes. Therefore, this thesis introduces an approach that is able to extract a larger variety of *contact skills*, skills that are performed in contact with the environment, enabling a flexible manipulation of objects. As these *contact skills* are defined by how the robot interacts with the environment, they are harder to extract from the demonstration, compared to simpler skills such as *Pick* or *Place*, as the characteristics of the interaction needs to be evaluated in detail. Thus we propose an approach that enables the detection of both a large variety of *contact skills* and simpler manipulation skills, such as *Pick* and *Place* by combining a semantic approach, that describes skills on a symbolic level, with a data-driven approach, that is able to learn the nuances required for the differentiation between *contact skills*. The introduced approach is able to extract the performed skills online from the user's demonstration, thus creating the center piece of our Interactive Task Programming (InTAP) framework.
InTAP aims to solve the second set of aforementioned challenges by creating an interactive teaching framework and providing a GUI, that guides the user through the teaching process. The user is able to transfer her knowledge by using kinestetic

teaching to physically guide the robot through the task, allowing an intuitive demonstration of the task. This demonstration is then segmented online into the skills used, shown as a task-graph to the user in the GUI. This gives the teacher an immediate feedback about what the robot is learning from the demonstration, helping her to understand the robot's knowledge and allowing her to react accordingly. In cases where the teacher made a mistake during the teaching process, or the robot learned the task incorrectly, the framework also enables the user to correct the robot's knowledge by providing options to manipulate the task-graph, making it easier for the teachers to ensure they are teaching the robot correctly.
This thesis gives an overview of the state of art in skill based PbD in Chap. 2, before introducing our skill segmentation approach in Chap. 3. This is followed by the introduction of the InTAP framework in Chap. 4. The intuitiveness of the InTAP framework is then investigated in a user study in Chap. 5. Finally, in Chap. 6, a summary of the work is given, as well as a discussion of the limitations of the current work and an outlook on future work.

Chapter 2

Related Work

2.1 Programming by Demonstration

As robots are introduced in further parts of production cycles and the demand for customized products reduces the life cycle of programmed routines, new alternatives to the traditional programming of robots are emerging. They are required to allow for fast adaptations of the robot's routines as well as to give non-experts the possibility to program the robot in an intuitive way. Programming by Demonstration (PbD), also referred to as Learning from Demonstration or Imitation Learning, is one of these approaches. It allows the teacher to demonstrate the new behaviours, which the robot then learns through imitation. As this is similar to the way humans teach one another, it is intuitive to use and requires no background in robotics or traditional programming [BCDS08] .
The demonstrations can be given through different means. In the work of Zhu and Hu [ZH18], three main groups are identified: kinesthetic demonstrations, motion-sensor demonstrations and teleoperated demonstrations. During kinesthetic demonstrations, the teacher guides the robot directly to perform the task. Here the advantage is that the robot is able to experience the demonstrations with its own sensors, thus there is no correspondence problem between the demonstration and the following execution. This correspondence problem needs to be solved for motion-sensor demonstrations. There, the human gives the demonstration with his own body and the motion is tracked, for example with the help of a marker-based tracking device. The demonstration then needs to be translated to fit the movement capabilities of the robot, which can be a challenge. The last demonstration type, teleoperation, allows the teacher to control the robot with the help of a control device. Again there is no correspondence problem, as the robot is able to perform the demonstration and record it with its own sensors.
Apart from the above mentioned correspondence problem, in [PHAS09] two more major challenges for PbD are pointed out: generalization and robustness against perturbation. As it is not feasible to give a demonstration for every single variation of a task, the system needs to be able to generalize the execution of a task

from only a few examples. Furthermore, as slight changes in the environment are difficult to suppress, the system also needs to be able to perform the task reliably against perturbations. Two of the most popular approaches to overcome these challenges are to encode the demonstration either on a trajectory level or on a semantic level. As our approach also aims to encode the demonstrations on a semantic level, hence splitting the demonstration into skills which then describe how to perform the newly learned task, this will be discussed in more depth in Sec. 2.2. To encode the demonstrations on a trajectory level, approaches are commonly based on statistical modelling or dynamical systems [BCDS08]. For statistical modelling, Hidden Markov Models (HMMs) as in [HSM96] are commonly used, as well as Gaussian Mixture Models (GMMs) as in [CGB07]. To encode the skills in dynamical systems, Dynamic Movement Primitives (DMPs) [INS01] are commonly used. A survey over the different approaches which use these three solutions in an assembly setting can be found in [ZH18]. How PbD can be applied specifically for humanoid robots can be found in [CL18].
As our work expands an existing PbD approach [WEL20], the learning and execution of the tasks will be done with the trajectory encoding of that approach, while the contribution of this works is to add a semantic encoding to represent the robot's knowledge for a better explainability, as discussed in Sec. 2.3.

2.2 Segmentation

The aim of our approach is to present the robot's knowledge in a way intuitively understandable to the human teacher. The level of detail in which this knowledge is presented is a crucial part of the design process. The commonly used levels can seen in Fig. 2.1.
Here the lowest level is that of primitives, so basic movements such as opening a gripper. The next higher level is that of skills, which describe an action consisting of multiple primitives, such as picking up an object. The highest level is that of a task, so the overall goal of performing the actions, such as sorting objects.
Choosing the task level to represent the robot's knowledge is too coarse to help the human teacher grasp the details of what the robot has learned. The primitive description on the other hand provides such a high level of details that it requires the user to first summarize the primitives to understand their goal, making it difficult for the user to understand their purpose at first glance. Therefore, the skill level was chosen to represent the robot's knowledge in our framework.
These skills will then be arranged in a task-graph which describes which skills the robot would execute to perform a certain task. To find this description, the tasks must thus be available on a skill level. Therefore, it is necessary to split the demonstrations given by the teacher into the relevant skill segments. To find these skill segments, different approaches can be applied, of which an overview is given in the following sections.

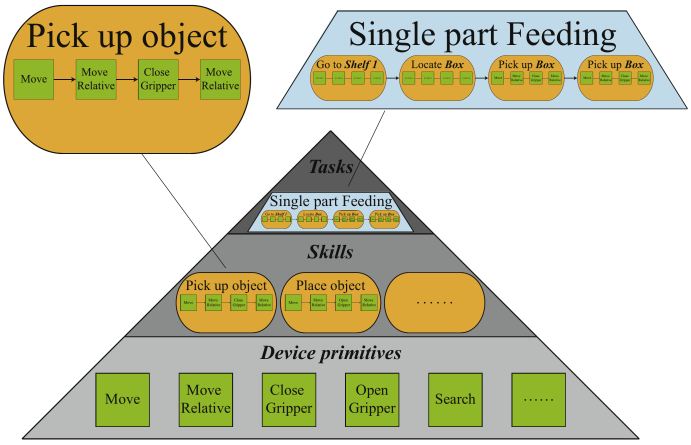


Figure 2.1: The three levels of detail describing a robot's task knowledge according to [PNA+16]

2.2.1 Movement Primitive Segmentation

Even though our approach uses the skill level to describe the robot's knowledge, this section will briefly touch on approaches which segment a demonstration into primitives, as many of those methods can be altered to segment skills instead.
One of the largest fields of research which uses primitive segmentation is that of human motion segmentation. There, a human's movement is split into primitives to find models of how a human moves. These can then be analyzed to evaluate the motion, for example in the setting of physical rehabilitation as in [HKK14] or used to teach robots as part of PbD. An concise overview over different segmentation methods, as well as a framework to evaluate them, can be found in [LKK16]. There, the methods are grouped in online, semi-online and offline, as can be seen in Fig. 2.2. Online approaches tend to find possible segmentation points by identifying changes in the movement pattern rather then by identifying the primitive in the demonstration. A commonly used method is to identify points where the velocity crosses zero, so called Zero Velocity Crossing (ZVC), as used by [FMJ02]. These points represent stopping points in the continuous movement, indicating when a motion ends and a new one begins. Alternative methods find metrics to compare how similar two consecutive parts of the demonstration are, thus finding segmentation points where a threshold of this metric is crossed. Common metrics are the Euclidean-distance as in [MKZ+09] or the Mahalanobis distance as in [BSP+04]. The disadvantage of online approaches is that they tend to over-segment, as they lack the knowledge of what the primitives are. Furthermore, they only give possible segmentation points, but no label for what primitive is performed in the segment. This label needs to be found in an additional step. An example for this is [KOL+12], which identifies primitives by stochastic segmentation, making use of the assumption that segments

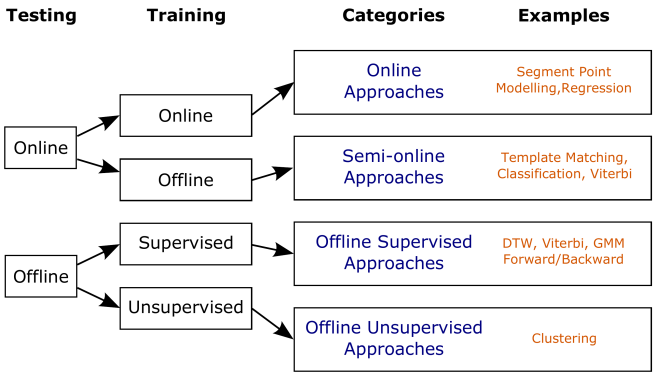


Figure 2.2: The segmentation methods are grouped into online, semi-online and offline by [LKK16].

belonging to the same primitive also have the same underlying probability distribution. This then allows the creation of a hierarchical tree structure which clusters the detected motion primitives, allowing the incremental learning of motion primitives and their grouping.

Instead, semi-online methods can also be used. They use preexisting knowledge about the primitives to identify which part of the demonstration belongs to a primitive. This preexisting knowledge is used in the offline training phase to build some kind of model of the primitives, which are then used in the online testing phase. One method is to apply template matching. There, a template is created from certain features of the primitives and then used to compare different segments of the demonstration to this template. A common method to encode the features sequences are HMM as in [LK13]. In that approach, in the first step, possible segmentation points are found online, as shown in Fig 2.3. This is done by evaluating which joints

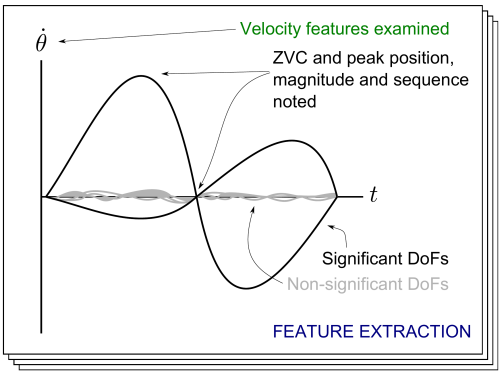


Figure 2.3: The first step in template matching is to identify possbile segmentation points. In [LK13], ZVCs and velocity peaks are used to identify them.

are active at each time step, denoted as significant Degree of Freedom. Then their velocity profile is extracted and multiplied with one another. From the thus gained velocity feature the ZVCs and velocity peaks are identified. The authors assume that each primitive consists of three ZVCs separated by two velocity peaks. This way the demonstration is split online into possible segments. Then the forward algorithm is used to find the best match from the template HMMs, which have been created with the Baum-Welch algorithm. The best match is chosen and the segment is assigned to belong to this primitive.
An alternative approach [MTS12] encodes the primitives in the form of DMPs instead. A library of DMPs is thus created from the learned primitives. During the demonstration, segmentation points are found online and the segments are encoded as DMPs. These are then compared against the library to find matches. A challenge for template matching approaches can occur when the demonstrations of the same primitive vary strongly, as finding a suitable template thus becomes harder.
Alternatively, classification methods can be used to identify the primitives in the demonstration. This can be done with a sliding window, where for each time-step a surrounding window is taken and classified, for example with the help of Support Vector Machines (SVMs) as in [BVL12]. A challenge for methods like these are temporal variations in the primitives, as the window size influences the result of the segmentation drastically.
When both the training phase as well as the testing phase are performed offline, approaches are considered offline. This can be necessary when the whole demonstration must be available before the segmentation can be performed, or when the computational costs are too high to run the algorithm online.
A common method is Dynamic Time Warping (DTW) [SC78]. In DTW, the problem of matching demonstrations to primitives performed at different speeds is solved. This is done by warping the time vector such that the error between the demonstration and the primitive is minimized. A shortcoming of DTW is that poor warping can lead to the warping of disproportionately large parts of the demonstration to small parts of the primitive, thus distorting the result. Furthermore, DTW becomes too computationally expensive when performed on high dimensional data.
An alternative to DTW is to use HMMs with the Viterbi algorithm [Rab89]. There, each primitive is encoded in a HMM and the Viterbi algorithm is used to find the sequence of HMM states, which fit the demonstration most likely. An example of this method can be found in [VRL+09]. Another approach using HMMs is [LO11], which introduces a forgetting factor, allowing the incremental refinement of the motion primitives by forgetting old examples and substituting them with newer, more relevant ones.
The method introduced in [NOKB12] uses Beta Process Autoregressive Hidden Markov Models to parse the demonstrations into reoccurring segments, which can be saved as a primitive library. This way new primitives can be detected without previous knowledge, as well as those already present in the library recognized as well. Additionally, for each primitive the coordinate frame is found by detecting

which object the primitive interacts with, providing further information to match the primitive. A downside to this approach is that the computational cost for using the primitive library is growing drastically with the number of primitives contained.

2.2.2 Skill Segmentation

As skills are more complex then primitives, it is harder to detect them through changes in the motion of the demonstration, as multiple different movements may belong to the same skill. For example, to perform a peg-in-the-hole skill, the insertion process can consist of sliding the peg along the edge of the hole until it is inserted and then pushing it down to fit completely. Therefore, the approaches to segment skills make use of semantics such as knowledge about which skills can be performed, for instance in the form of template matching, or with which objects the robot is interacting with.
Apart from the division in online and offline methods, the approaches can be divided depending on how they learn the skills and then detect them during the demonstration. Semantic approaches use symbolic conditions to define skills, while data-driven approaches learn the skills from data. Examples for semantic approaches are the montion-sensor PbD approaches used in [NM01, NM03], where each skill has a set of unique postconditions. These postconditions describe the effect the skill has on the robot's state as well as on the environment. They are designed by hand depending on what effects a skill is expected to achieve. During the demonstration, each skill is monitoring it's postconditions. When they are fulfilled, a skill is considered executed and the segment is assigned to that skill. Furthermore, skills have preconditions, which hold information about which skills must have been performed before they can be executed. But unlike the postconditions, these are not designed by hand beforehand, but instead learned from the demonstrations. By comparing the order in which skills are performed during the demonstration, a behaviour network is learned that describes when skills can be executed. This information can then be used to refine the detection of skills, as well as to provide the robot with information about which skills it can choose from to solve a task during task execution. A downside to this approach is that by only using the preconditions to describe the order in which skills can take place, but not the external or internal conditions which mark the beginning of a skill, it is implied that every skill starts exactly where the previous skill ended, omitting transition phases in the representation.
Therefore, other approaches explicitly define preconditions for skills as well, not only describing the order in which skills can take place, but also which conditions the environment and robot state need to fulfill. This is done in the kinestetic PbD approach by [SNS19], where skills are described in the Planning Domain Definition Language. There, the skills' pre- and postconditions, as well as over-all conditions, so conditions which need to be fulfilled during the whole skill execution, are defined. These definitions are then used to detect the skills in the demonstration. To check if the conditions are fulfilled, a world model is used to monitor all relevant objects in

the work-space, as well as the robot's state. An example for an instance of a world model can be seen in Fig. 2.4. The world model holds the information of where

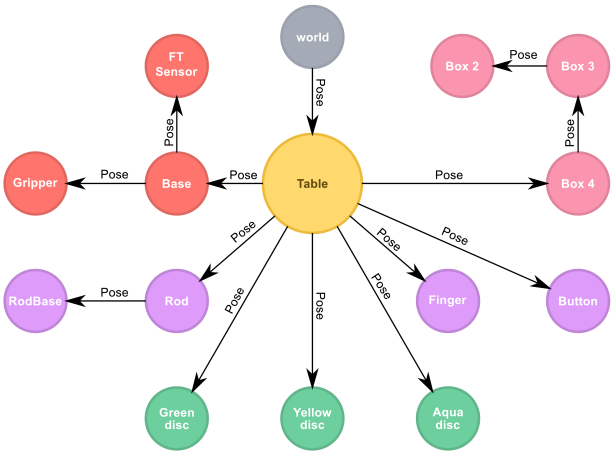


Figure 2.4: An exemplary world model used in [SNS19] to monitor the objects and the robot's pose in the work-space. The colour indicates what kind of object the nodes belong to. Red marks parts that belong to the robot, yellow the surface of the work-space and the other colour different classes of objects, as for example cylinders.

objects are, as well as which other objects they are in contact with, shown with an edge in the graph. In Fig. 2.4 for instance it can been seen that Box3 is placed on top of Box4, which stands on the table. As there is no camera used to directly obtain this information from the scene, a world observer is used to calculate this information from the actions performed by the robot. Thus it can update the world model continually. The skill recognition algorithm then uses the information from the world model to update the state of the candidate skills, depending on which of their conditions are fulfilled. When the preconditions of a skill are fulfilled, the information about which objects are used in that specific execution of the skill is saved in the form of a skill-value combination. Once the postconditions of this skill are also fulfilled, the skill is marked as detected and the skill-value combination is saved for the segment.
This approach has the advantage that it achieves a robust skill detection by using the objects' positions to check for conditions such as 'close-to', without having to work with image detection algorithms that lead to higher computational cost, and can thus be performed online. But this also has the drawback that the objects' exact positions and orientations need to be available to the world model at the beginning of demonstrations, which leads to an extra overhead, as well as a potential source of errors for teachers with little experience. Furthermore, the approach is unable to detect unexpected changes in the world, like for instance the sliding of an object in the gripper or an interference by the human teacher, which then lead to a mismatch between the world model and the reality.

This is why other approaches use image detection to ensure a correct world model, as done in [WSA+13]. There, a marker based motion capture system with multiple cameras is used to track the objects and the user to perform motion-sensor based PbD. But unlike the world model introduced before, the extracted world model here only contains information about which objects are in contact, not their positions or orientations. The authors argue that skills are used to interact with objects, thus only points in the demonstration where contacts between either objects and the user or objects and other objects happen, can be start- and endpoints for skills. Therefore, the information obtained by the motion capture system is used to detect moments where contacts change, which are then marked as keyframes. The contacts at these keyframes are then stored in the world model. To detect skills in the demonstrations, pre- and postconditions are also used. But whereas the pre- and postconditions of skills here are limited to whether there is contact or not, they also hold information about what kind of objects these contacts can take place with. These combinations of skill and object information are referred to as Object-Action-Complexes (OACs) and are stored in a library. To detect the OACs in the demonstration, keyframes and their corresponding world models are extracted. These are then compared to the OACs to find matches. An example of the detection process can be seen in Fig. 2.5. In the upper pictures the first

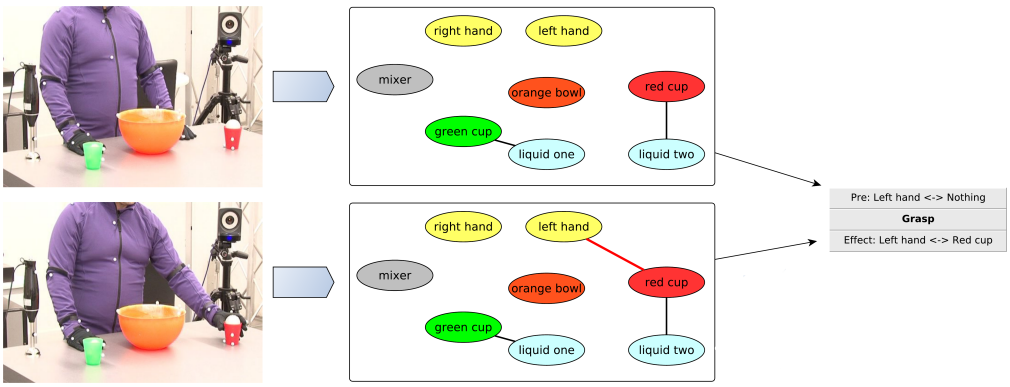


Figure 2.5: An example for the skill detection in [WSA+13]. On the left, the user's demonstration is shown for the extracted keyframes. The corresponding world model is shown in the middle. All tracked objects are shown as nodes and their contacts through edges. The right shows the pre- and postconditions for a grasping skill, which are fulfilled by the contact changes.

keyframe is shown. There, both hands are free, so the precondition for a grasping skill is fulfilled. The next keyframe shows that now the left hand is in contact with the red cup. Therefore, the postcondition for the grasping skill is fulfilled as well. Thus the grasping skill is detected with the objects 'left hand' and 'red cup'. While this approach is able to detect unforeseen changes in the world, it also limits what kind of skills can be detected, as no information about what kind of contact the

objects are in is considered. Both pressing down on an object and sliding it along a surface have the same contact changes, but have different goals. Furthermore, the additional effort for the user to wear the motion capture equipment and extra space needed for the set-up makes it unsuitable for quick and frequent teaching sessions.
Another approach working with object information for skill recognition is [ZPKD05]. There, the demonstration is first split in segments, starting with pick skills and ending with place skills. These segments are then further split with the help of the available information about the objects. For instance an object can be screw-able, so a contact with such an object is defined as a screw skill. The skills of the segment are then further decomposed into movement primitives by analyzing their trajectories, obtaining a hierarchical segmentation. The drawback of this approach is the strong limitation of how skills can be recognized by reducing them to solely depend on the objects they are performed with. An approach introduced in [KKGB12] uses Reinforcement Learning (RL) together with Changepoint Detection to segment a demonstration into skills. The authors first use RL to learn the value function for a demonstration, then use Changepoint Detection on it to find the points where the goal of the robot changes. The thus gained segments are then compared against a library of abstractions which describe all possible skills, and mapped to the one which fits the segment's value function best. This mapping provides further information about what the goal of that segment is, allowing further refinement with RL. This approach has the drawback that the value function needs to be learned before the segmentation is performed, thus only offline segmentation can be achieved and even that is time-wise costly, especially when complex skills are to be learned.
A similar method to [WSA+13] also uses changes in contact to detect keyframes, but learns the skills from contact sequences directly and hence belongs in the data-driven category. This method is called Semantic-Event-Chains (SEC) and has been an active research topic [AAWD10, AAD+11, SBS+17, AAW19]. There, the sequence of contact changes during a skill is used to describe it. A visual representation of the skill learning process can be found in Fig. 2.6. To obtain the information of when objects are in contact, computer vision is also used. But instead of using a marker tracking system, an image segmentation algorithm is used. It detects which parts of the image belong together and is thus able to detect the objects in the work space, as shown in Fig-2.6 (a)-(c). From these segmentations a graph is constructed with edges between objects that are in contact, as shown in step (d). This is done for all time-steps, thus creating graphs that describe the temporal changes in contacts, shown in steps (e)-(f). From this a SEC is created in the form of a matrix, in which each column represents a moment in time, as can be seen in step (g). For each combination of objects a row is created, in which the information about their relation is saved. Possible relations are touching, overlapping, no contact and that one of the segments no longer exists. To learn the representing SEC for a skill, this process is repeated for multiple demonstrations of that skill. By comparing the extracted matrices, the essential contact changes are extracted. These are then saved as a SEC for that skill. To segment a demonstration into the thus learned

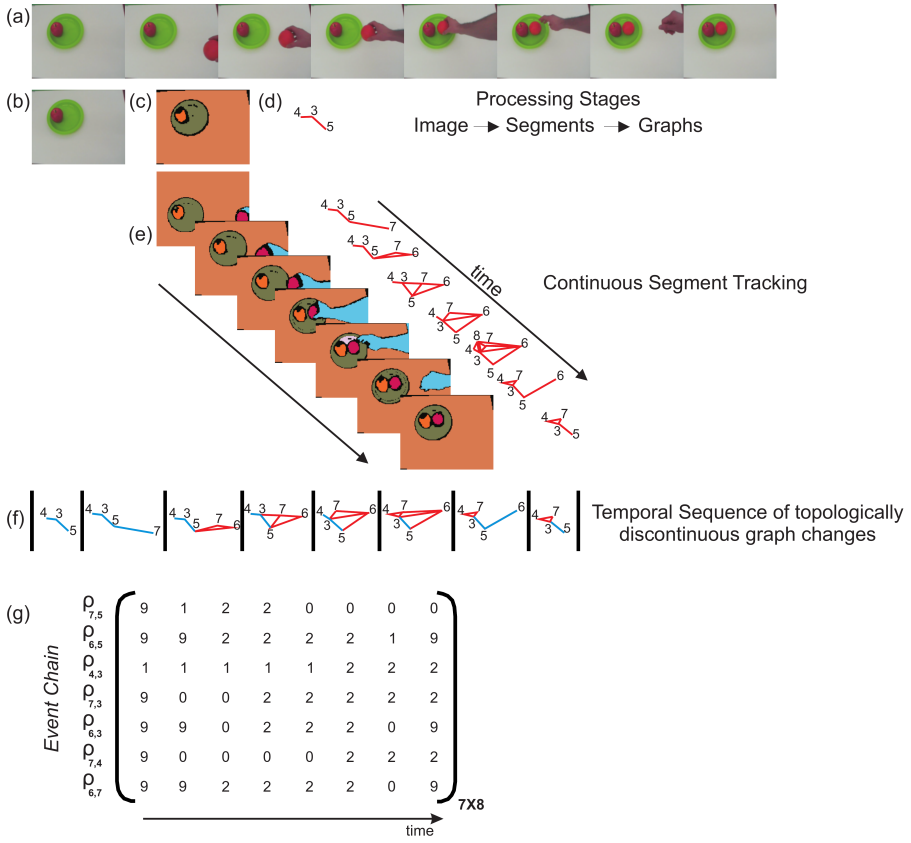


Figure 2.6: An example for the skill extraction used in [AAD+11]. In steps (a)-(c), objects in the work space are detected through segmentation. In steps (d)-(f) the graphs describing the objects' relations are created, which are then translated to a SEC in step (g).

skills, the same process is used. The demonstration is described by a SEC denoting the contact changes at the keyframes. This SEC is then compared with the help of a sub-string search to the existing skills. When a match is found, the skill is recognized. The advantage of this approach is that the set of learned skills can be expanded by non-experts as well, as they only need to demonstrate the new skill instead of having to find suitable conditions to describe it. Furthermore, unlike in the approaches described above, no preexisting knowledge of which objects are in the work space is necessary, removing the necessity to inform the system about them beforehand. Downsides to this approach are the vulnerability to mistakes during the image segmentation phase, the computational cost forcing the method to run offline as well as the lack of information about the dynamics of the contacts, thus limiting which skills can be distinguished. In order to enable such a distinction, this method has been expanded to extract additional information about whether skills are periodic or deterministic in [AZWA16], as well as to describe the spacial relations in [ZAWT17]. While these allow to detect if a contact is dynamic or static,

they are still unable to detect the differences in the dynamics of the contact, so for instance the difference between pushing something and hitting something. Another approach that uses spatial relations is [ZYFA15]. There, the space around objects is split into different areas, describing their relation with the object, such as 'behind' or 'above'. By tracking in which areas the objects are in respect to one another during the demonstration, a descriptor is created. This descriptor is used to encode the learned skills, as well as to recognize them in the demonstration using a custom distance metric. This approach also runs offline. Another drawback of this method is that an unconscious bias in the skill recognition can occur, for example by only providing demonstrations from a right handed teacher, who would use tools in different positions as a left handed teacher.
An approach more similar to the template matching used in primitive detection is introduced by [QELL20], where motion sensor PbD is applied. In the first step, possible segmentation points are detected at moments where the fingertip distance to an object crosses a certain threshold, changes in the hand velocity occur and the rise/ fall profile changes. The skills are encoded in HMMs, and in the second step of the segmentation, the forward algorithm is used to test if segments corresponds to a skill. To reduce over-segmentation, the authors evaluate not only the found segments on their own, but also combinations of segments. These combinations are created by starting from the first segment, then iteratively adding the next segment and evaluating the scores for all possible skills until all segments have been included. The process of building these segments can be seen in the upper part of Fig. 2.7. From that the combination with the highest score is chosen as correct and

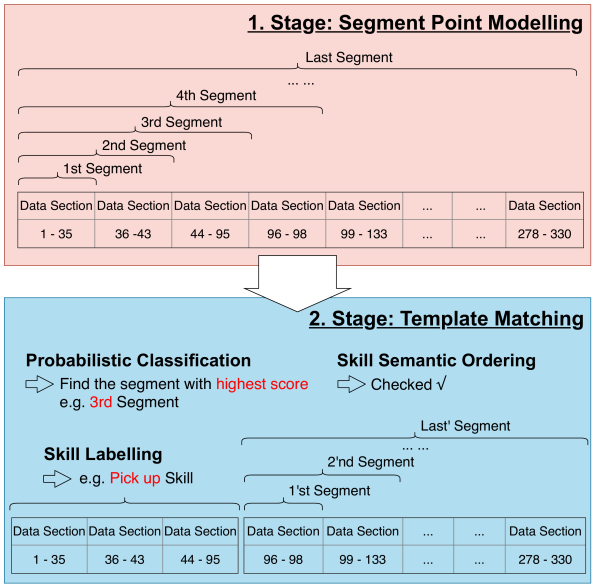


Figure 2.7: The segmentation process used in [QELL20]. The upper red part shows how the different candidate segments are build. The lower blue part shows how these are then matched to the learned skills.

the demonstration is cut from the beginning to the end of that combination. This is repeated until the whole demonstration is labeled, as shown in the lower part of Fig. 2.7. A downside to this approach is that mistakes at the beginning of the segmentation can influence the whole segmentation negatively. Furthermore, the creation of the data sections from the demonstration has no direct link to how the skills look like, thus creating a large number of incorrect possible segments, making the template matching more difficult.
An alternative route is therefore to try and find the skills in the demonstration directly, which is done by [WJX+18], who also use motion-sensor PbD. There, SVMs are trained on selected features to learn the skills, such as the hand gesture performed during a task, the movement angle with respect to the work space, as well as the vertical movement. The same features are then extracted from the demonstrations, and a sliding window is run over. This sliding window is used to obtain a score for each time-step, based on how likely it belongs to the different skills. The points where the prediction of which skill is the most likely changes are taken as segmentation points. An example of this can be seen in Fig. 2.8. To further im-

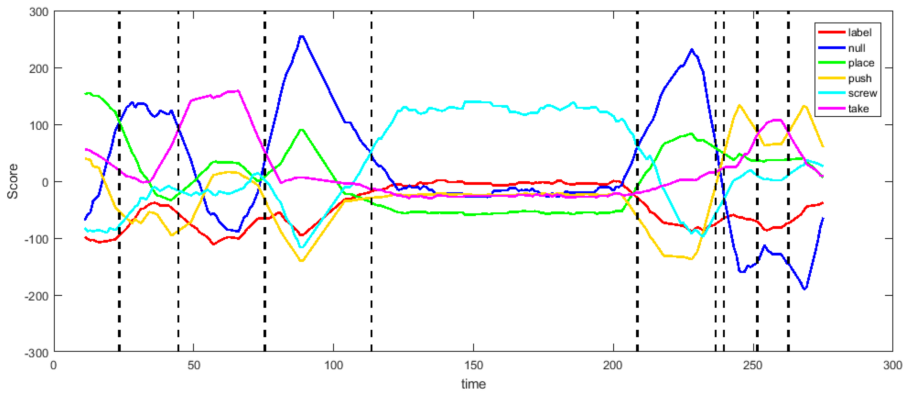


Figure 2.8: The segmentation process used in [AAD+11]. For each frame SVMs are used to obtain a score, marking how well the time frames fit to the different skills. The dotted lines show the segmentation lines.

prove the segmentation, segments which are considered too small to be a skill are merged into their neighbouring segments. Additionally, the segmentation lines are adjusted with a local optimal boundary search. During this search, the start and end points of the segments are moved slightly and the SVMs are used to score each possible combination of start- and endpoints. The highest scoring combination is chosen and the process is repeated for each segment, until the start- and endpoints for all segments have converged. This search is shown to significantly improve the segmentation result. But as the authors point out, it can only be used because it was possible to find highly distinguishing features, due to the choice of skills. For a wider variety of skills, more features must be included, making the search more expensive, possibly even forcing the algorithm to run offline instead of online.

2.3 Human-Robot Interaction

To arrive at a widespread use of robots in the society, robots' capabilities need to be expanded. The problem of making robots easily teachable and capable of generalization is addressed by PbD. It allows even non-experts to teach robots, so that they can take over whatever tasks humans do not want to perform and therefore would like to leverage to robots. But while transporting boxes from one side of a room to another might be a task humans willingly entrust to robots, more dedicated tasks, where a failure might lead to serious damage, are still primarily executed by humans. To introduce the use of robots in these kind of tasks, two problems must be solved: the robots must be able to perform the task without critical failure and the human must trust in the robot's ability to do so [EGL+19]. Most of the approaches introduced in Sec. 2.2.2 try to solve the first part of the problem by using the segmented skills to introduce expert knowledge in the execution of the skills, for example by using skill depended controllers, as in [AAW19]. This improves the robot's performance and can even be used to monitor the success of performing the task. But it still treats the robot's learning process as a black-box problem: the demonstration is its input and the learned task its output.
Therefore, a demand for robots that are able to explain themselves to the human teacher is growing, as argued for in [She17]. There, five advantages are pointed out that can be obtained from good explanations: First and foremost, humans tend to trust robots which they believe to understand. This trust is necessary for humans to even consider using the robot. That a good representation induces trust is also the conclusion of [EGL+19]. There, the authors compare explanations of the robot's behaviour in different forms regarding their success in building trust in the robots capabilities. The results show that semantic representations of the robot's behaviour achieve the best results, supporting our believe that a skill based semantic representation improves the teaching experience.
The second advantage is that having access to a robot's decision process is the first step for introducing accountability in robotic systems. When failures occur, what can never be completely avoided, knowing why it went wrong is necessary to know who is responsible. Furthermore, it makes it easier to correct the specific part that caused the failure, leading to the third point, ease of debugging.
Especially for non-expert users, debugging the robots' behaviours is challenging, as they lack the understanding to know what to expect of robots. Therefore, a good representation of the robot's knowledge is essential for non-experts to see at first glance where the robot failed to produce a satisfactory result, giving them the option to improve it. This also connects to the fourth point, which addresses the human teacher's ability to create good demonstrations, which are the foundation of creating successful robot executions.
As pointed out in [SH20], incorrect demonstrations are one of the frequent causes for failures. This is why the authors investigate what kind of feedback helps the teachers most to improve their teaching, and find that showing the teacher a visual

representation of what the robot has learned can improve the teaching efficiency up to 180%. This result is also supported by [CT14], where the effect of providing teaching guidance is investigated. This shows that giving the teachers feedback about which consequences their teaching has on the robot's understanding of the task, speeds up the teaching and helps them to give better demonstrations and to build a better mental model of the state of mind of the robot.
The necessity of aligning this mental model, so the teacher's expectation of what the robot has learned, to the robot's real knowledge, also referred to as state of mind, is discussed in detail in [HB18]. There, the authors point out that for the human to be able to understand the robot's behaviour, and therefore to be able to comprehend why the robot acted the way it did as well as to be able to predict what the robot will do next, the human's assumptions about the robot's abilities must be correct. That a mismatch hinders the teaching process is shown in [KTM10]. There, robots were taught new behaviours with the help of pre-programmed skills, that could be chosen from a program. The experiment tested two different cases: in the first one, the teacher was allowed to see the robot during the teaching process and could so directly see the execution of the skills. In the other case, the teachers only obtained feedback through the program, telling them when the skills were executed.
The result showed that the group with limited access to the robot performed better, as they had to rely on the information given to them, while the other group could rely on their own interpretation of the robot's movements. This lead to teachers commanding the performance of a new skill before the last one was finished, as they misjudged when the robot was done performing the skill. This shows why communicating the robot's state of mind to the teacher is so important, as well as to choose the communication to be as easily understandable as possible. Therefore, the last advantage of presenting the robot's knowledge to the human is discrepancy resolution, so closing the gap between the teacher's mental model and the robot's state of mind.
But while the necessity of presenting the robot's knowledge to the user is recognized and the advantages from doing so are clearly visible, hardly any of the before mentioned approaches try to use the skill segmentation for that purpose. As skills are usually describing the robot's knowledge in words that humans would also use, they are able to communicate the state of mind of the robot to the user. Therefore, by showing the human teacher which skills the robot is going to execute, the teacher can trust that the robot understood the goal of the task. Furthermore, in case of wrong segmentation or skill recognition, the teacher is able to notice the mistake before the execution of the program, thus being ableto correct them before a failure occurs, giving a higher ease of debugging. Additionally, the teacher also sees which demonstrations lead to which skill recognition, enabling him to provide clearer demonstrations in case of repeated wrong recognition. And if these do not improve the recognition, it is also easier to find out if the blame lies with the user still providing poor demonstrations, or with the recognition program, thus improving accountability.

Yet only the method used by [SNS19] aims to provide a dialog for the user, by using the obtained skill segmentation as an input for task level programming, so to create a program that uses pre-programmed routines to perform the detected skills. These are then shown to the user with the help of a GUI, of which an example can be seen in Fig. 2.9. The GUI can be used to edit the order of the skills and their pa-

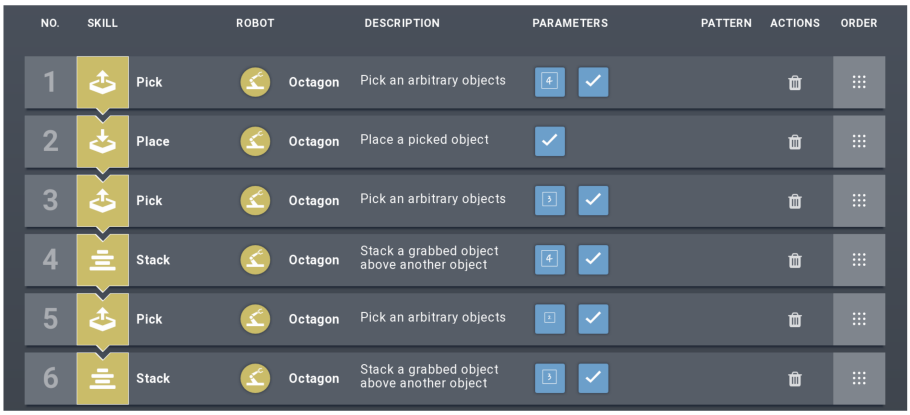


Figure 2.9: The GUI used in[SNS19]. It shows the skill sequence for an exemplary task by giving the skills names and descriptions.

rameters. But while this design fulfills all of the above mentioned criteria, it limits the robot's skill execution to linear programs. In the approach on which our work is based, the robot is able to choose from multiple behaviours to react more flexibly to environmental circumstances. Therefore, the moments where such decisions can be made as well as their consequences must be represented as well.
Therefore, this work aims to extent the work done in [WEL20] to allow an intuitive representation of the robot's knowledge. This is expected to build trust in the robot's capabilities, to give accountability, to improve the ease of debugging and the user's teaching capabilities and to help close the gap between the teacher's mental model and the robot's state of mind. The approach with which this is tried to be achieved is introduced in the following.

Chapter 3

Skill Based Task-Graph

The first step in sharing the robot's knowledge with the user is to extract this knowledge from the user's demonstration. The knowledge transfer used in our framework can be seen in Fig. 3.1. First, the user provides a demonstration of the new task

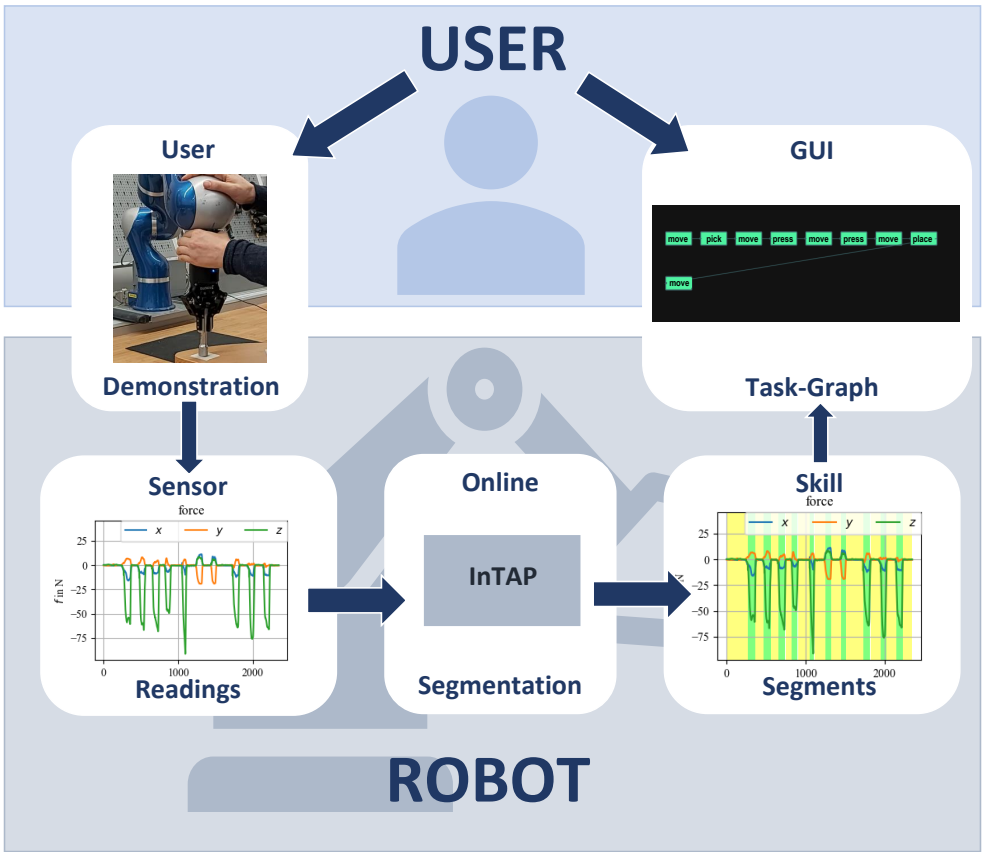


Figure 3.1: An overview of the information flow in our framework. The user can refine the robot's knowledge either by giving new demonstrations, which are then segmented into a task-graph, or by manipulating this task-graph directly in the GUI.

with the help of kinestetic teaching. The user's demonstration is recorded and then down-sampled to 50Hz. The down-sampled sequence of force, torque, gripper status, gripper opening, position and orientation are then used to extract the performed skills, which are then presented to the user as a task-graph via a GUI. This task-graph describes the sequence of skills the robot has learned as the necessary steps to perform the task, thus giving the user a feedback about what the robot learned from the demonstration. To enable an immediate feedback, the skill extraction is performed online.
This is achieved by merging a data-driven and a semantic segmentation approach into a combined segmentation method, inheriting the advantages of both approaches. Our approach performs the online segmentation in a two-step manner: In the first step, the result from the semantic segmentation is used to create a preliminary online segmentation. In the second step, the preliminary segmentation result is used to split the demonstration into smaller parts, which then can be refined in batches by the combination of the data-driven segmentation and the semantic segmentation. This leads to an evolving task-graph that is constantly updated during the user's demonstration.
Once the user has finished demonstrating the task, the whole demonstration is segmented by the combined segmentation method. The thus obtained task-graph can then be used to further refine the robot's knowledge. The details regarding the user's possible interactions with the task-graph are given in Chap. 4, describing the Human-Robot-Interaction concept used in our framework.
In the following, the segmentation algorithm used to obtain the task-graph is introduced.

3.1 Skill Segmentation Algorithm

To extract a skill level description of the task demonstrated by the user, this demonstration must first be segmented into skills. In this framework, the focus lies on identifying a larger variation of so called *contact skills*, skills performed while being in contact with parts of the environment. As discussed in Chap. 2, only few other methods try to solve the problem of detecting skills which are similar in their contact sequence, but different in their purpose. Instead, most focus on extracting only a few, easily distinguishable skills. But as the use-case for our framework is an assembly line setting, where the robot needs to be able to learn multiple ways of manipulating objects, a larger variation of teachable skills is necessary.
Therefore, our framework detects apart from the commonly used skills such as *Pick*, *Place* and *Move*, the following *contact skills*: *Press, Slide, Contour, User, Peg-in-hole*. A short description of each skill can be found in Tab. 3.1. The detection of the *contact skills* is achieved by using a combination of semantic and data-driven skill segmentation. While most semantic approaches are unable to detect dynamically different *contact skills*, as their characteristics are hard to describe with pre- and

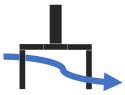
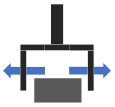
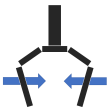
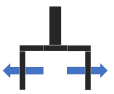
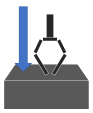
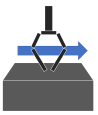
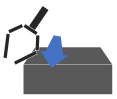
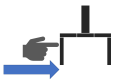
Skill Name	Description	Image
Move	The robot moves without any interaction with the environment	
Pick	The robot takes an object	
Place	The robot releases an object from its grasp	
Close Gripper	The robot closes its gripper and does not hold an object afterwards	
Open Gripper	The robot does not hold an object and opens its gripper	
Press	The robot presses against a surface without additional movement	
Slide	The robot presses against a surface while sliding along it	
Contour	The robot presses against the side of a surface and follows its outline	
Peg-in-hole	The robot inserts an object into a hole	
User Interaction	The user pushes against the robot's gripper	

Table 3.1: Table describing the different skills. The blue arrows mark the movement direction of the robot.

postconditions, data-driven approaches can learn to distinguish these skills by their force and torque profile. Yet the drawback of data-driven segmentation is a tendency to over-segment, which then requires expensive post processing steps. That is why most data-driven approaches are not suited for online segmentation. By combining both semantic and data-driven segmentation, their drawbacks can be compensated. Data-driven segmentation is able to distinguish the *contact skills*, and semantic segmentation gives reliable segmentation lines while being able to run online. Thus, an online segmentation algorithm that robustly detects a variation of *contact skills* can be created. The details of the semantic and data-driven approaches used in this work are introduced next, followed by a detailed explanation of their combined application in our proposed online segmentation.

3.1.1 Semantic Segmentation

The semantic approach of this work uses pre- and postconditions to recognize skills during the demonstration. The preconditions describe which conditions need to be fulfilled to mark the beginning of a skill, while the postconditions describe which conditions need to be fulfilled to indicate that a skill is done. Similar to the approach used in [SNS19], we also use over-all conditions. These mark conditions which need to be fulfilled during the whole execution of a skill.
The conditions are defined by hand by the expert designing the skills. In our work, the conditions are computed from the movement of the robot's gripper fingers, as well as the readings from the FT-sensor installed between gripper and robot. As these measurements are not enough to distinguish between the different *contact skills*, the semantic approach is limited to detecting a general *contact skill* class instead. Overall, the semantic segmentation can detect Pick, Place, Open-Gripper, Close-Gripper, Move and Contact. An overview over the skills detected by the semantic segmentation, as well as which conditions are used to detect them, is shown in Tab. 3.2
For each time step, the measurements recorded during the demonstration are used to evaluate which flags are set, describing the conditions currently met. The flags used are *OpenGripper, HoldingObject, GripperMoving* and *Contact*. The *OpenGripper* flag is computed from the distance between the robot's gripper fingers. The distance is normalized to the range $[0, 1]$, where 0 indicates a completely closed gripper and 1 a fully open gripper. The threshold to indicate *OpenGripper* has been chosen as 0.6, to allow the gripper to hold larger objects. The *HoldingObject* flag is found from the gripper status, which is read from the gripper. The gripper status has the value -1 if the gripper does not hold an object and 1 if the gripper is holding an object. Furthermore, it has the value 0 if the gripper is moving. Therefore, the *HoldingObject* flag can directly be taken from the gripper status when the gripper is not moving. If the gripper is moving, the last set *HoldingObject* flag is repeated until the gripper stopped moving again. The information about the gripper's movement is also used in addition to the gripper finger distance to set the *GripperMoving* flag.

Skill Name	Preconditions	Over-All Conditions	Postconditions
Pick	NoHoldingObject OpenGripper GripperMoving		HoldingObject ClosedGripper NoGripperMoving
Place	HoldingObject ClosedGripper GripperMoving		NoHoldingObject OpenGripper NoGripperMoving
Close Gripper	NoHoldingObject OpenGripper GripperMoving		NoHoldingObject ClosedGripper NoGripperMoving
Open Gripper	NoHoldingObject ClosedGripper GripperMoving		NoHoldingObject OpenGripper NoGripperMoving
Move		NoGripperMoving NoContact	GripperMoving Contact
Contact Skill	Contact	Contact NoGripper- Moving	NoContact

Table 3.2: The Table showing the conditions used in the semantic segmentation to recognize skills. The over-all conditions need to be fulfilled during the whole execution of the skill.

The *Contact* flag is found from the measurements of force and torque. The norm over the force in all three dimensions is taken and compared to the force threshold, which has been chosen as 5N. The same is done for the torque, with a threshold of 2Nm. If either of the two crosses their respective threshold, the *Contact* flag is set. With the help of the flags the skills are detected in the demonstration, by monitoring which of their conditions are fulfilled. The algorithm for the semantic segmentation can be seen in Algorithm 1. All skills start with their status set to IDLE, indicating that no skill has been recognized. Once the preconditions of a skill are met, as the required flags are set, the skill is set to ACTIVE. The skill continues to be ACTIVE, until either their postconditions are met or their over-all conditions are not fulfilled anymore. If their over-all conditions are not met anymore and their postconditions are not met yet, the skills are reset to IDLE. When their postconditions have been met, their status is set to DONE. Then the skill is added to the list of detected skills together with their start- and end time. Furthermore, all skills are reset to IDLE after a skill has been detected. This means that no more than one skill can be found

Algorithm 1 Semantic Segmentation

Input: Measurements m, Semantic skills $skills$

```
while m do
    flags ← extract_flags(m)
    for skill ∈ skills do
        skill ← update_status(flags)
        if skill.status == DONE then
            segments[skill.start:skill.end] ← skill.name
            for s ∈ skills do
                s ← reset_to_idle
            end for
            break
        end if
    end for
end while
```

for a segment.
An example of a semantic segmentation result can be seen in Fig. 3.2. There, the robot has been taught to perform a sequence of skills consisting of Move, Peg-in-hole, Place, Close-gripper, Press, Slide and User. The contact skills are marked in black. From the image can be seen that the semantic segmentation is detecting the start- and endpoints of the skills reliably, but is unable to recognize which *contact skill* is performed. Furthermore, the semantic segmentation is unable to detect that the *contact skill* segment after the Close-gripper skill consists of two *contact skills*, namely Press and Slide, as they are performed without breaking contact in between.

3.1.2 Data Driven Segmentation

The data-driven segmentation proposed in this work uses Support Vector Machines (SVMs) with a sliding window approach similar to [WJX+18]. For each time-step, features of the measurements in the surrounding time window are extracted and fed into the SVMs, which then predict which skill is performed. The features used in this approach consist of two parts. The first are features extracted with the publicly available library tsfresh [CBNKL18] and the second are features designed to allow a better distinction between the *contact skills*.
The tsfresh library was chosen as it allows a large number of possible features to be extracted from time-series. By mostly relying on a generic library to extract the features, instead of creating a complete set of specifically designed features, the framework avoids becoming overly focused on the *contact skills* chosen now, and allows the easy addition of more *contact skills* in the future. Furthermore, the tsfresh library groups its features according to their computational costs, thus allowing a flexible choice of features depending on the computational power of the set-up cho-

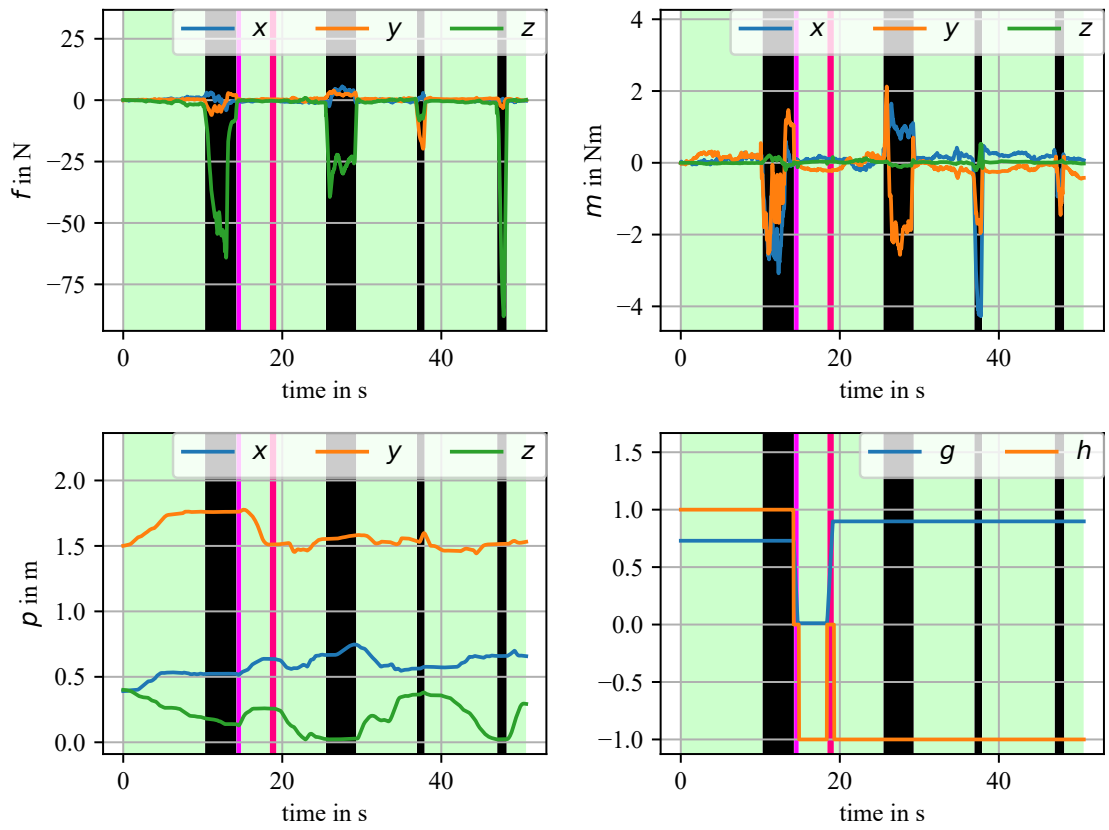


Figure 3.2: Exemplary segmentation result for the semantic segmentation. The semantic segmentation is overlaid on the end-effector forces (upper left), the torques (upper right), the cartesian end-effector position (lower left) and the gripper movements (lower right). The skill sequence performed is Move, Peg-in-hole, Place, Move, Close-gripper, Move, Press, Slide, Move, User, Move. The Move skill is shown in light green, Place in pink, Close-gripper in red and the general Contact skill in black.

sen for the framework to run on. In this work, the minimal feature selection is used. For the tsfresh features, the measurements are not used directly to extract the features, but are preprocessed first. This is done by computing the velocity from the cartesian position of the end effector, and the angular velocity from the rotation of the end effector frame. Additionally, as the skills should be detectable independent of the direction they are performed in, the norm over the x-,y- and z-axis is computed for the velocity, the angular velocity, the rotation in Euler angles, the forces and the torques. The tsfresh features are then computed. Finally, after combining the tsfesh and the *contact skill* features, the features are normalized to mean $\mu = 0$ and standard deviation $\sigma = 1$.
As mentioned before, the *contact skill* features have been added to improve the recognition of the contact based skills. This is done by introducing knowledge about

the physical meaning of the measurements, which the tsfresh library is unable to do. Thus features such as the mean power can be introduced, describing the dynamic behaviour of the contact. An overview over the *contact skill* features can be found in Tab. 3.3. As the strength of data-driven segmentation lies in detecting skills by

Feature Name	Meaning
mean_power_f	The mean power computed from force and velocity.
mean_power_t	The mean power computed from torque and angular velocity.
slope_to_max_force	The slope of the line connecting the force at the moment of first contact to the force at the moment of the maximum force in the force domain.
slope_from_max_force	The slope of the line connecting the force at the moment of the maximum force to the force at the last moment of contact in the force domain.
length_contact	The temporal length of the contact.
movement	The total displacement between beginning and end of the skill.
cov_torque_force	The correlation factor between torque and force.

Table 3.3: Table describing the additional *contact skill* features.

their force and torque profile, the SVMs are trained to classify the skills Move and the *contact skills* Press, Slide, User, Peg-in-hole and Contour. The skills based on gripper actions, such as Pick and Place, are not learned data-driven, as they can be robustly and efficiently extracted by the semantic segmentation. Therefore, any attempts to learn them from data are likely to reduce the recognition performance instead of improving it.
The SVMs used in this approach use a Radial Basis Function (RBF) kernel. The regularization parameter C and the kernel coefficient *gamma* are found through a grid search during the training. The SVMs are trained on examples of the skills both consisting of the whole skill as well as parts of the skills as large as the sliding window. Thus, the skills can be recognized when they are only partially in the sliding window or when they are too long to fit completely inside.
Furthermore, to allow the detection of the *contact skills* independently of the tool they are performed with, the information about the tool's weight and center-of-mass is used to compensate the effect of the tool on the skill performance. This also implies that our framework assumes that it always knows which tools are used during the demonstrations.

An example of a data-driven segmentation result can be seen in Fig. 3.3. The de-

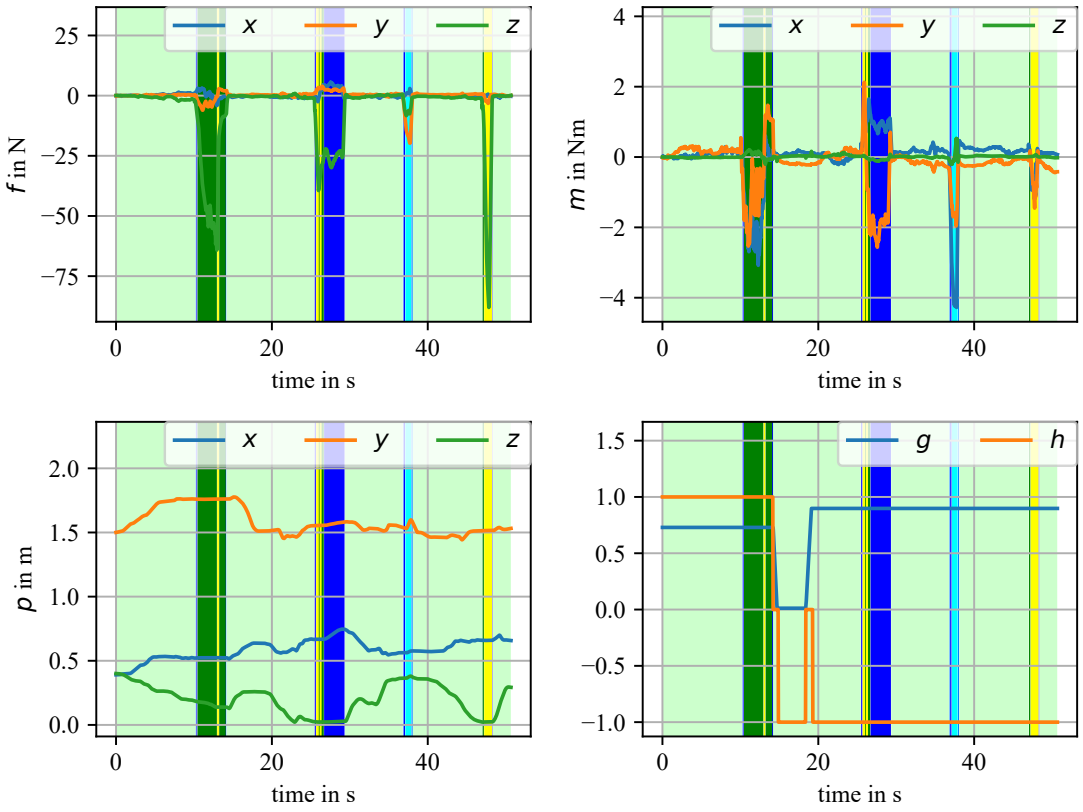


Figure 3.3: Exemplary segmentation result for the data-driven segmentation. The data-driven segmentation is overlaid on the end-effector forces (upper left), the torques (upper right), the cartesian end-effector position (lower left) and the gripper movements (lower right). The skill sequence performed is Move, Peg-in-hole, Place, Move, Close-gripper, Move, Press, Slide, Move, User, Move, Press, Move. The colours are: Move (light green), Peg-in-hole (dark green), Press (yellow), Slide (dark blue), User (cyan).

picted demonstration is the same skill sequence as shown in Fig. 3.2. From the image can be seen that the data-driven segmentation is able to detect which *contact skill* is performed during the demonstration. Yet it over-segments, incorrectly detecting multiple skills during a single skill demonstration. This occurs most often at the beginning of the skills. Furthermore, the data-driven skill segmentation does not recognize skills based on gripper actions.

3.1.3 Combined Segmentation

The combination of the data-driven segmentation and the semantic segmentation has the advantage of detecting skills based on the gripper finger movement, like Pick

and Place, on a semantic level, and *contact skills* by combining the results of the semantic and data-driven segmentation. The approach introduced in the following performs semantic and data-driven segmentation in parallel, to find all possible skill segments that can be found in the demonstration. Then, it finds the best possible combination of the skill segments, as can be seen in Fig. 3.4. A parallel instead of a

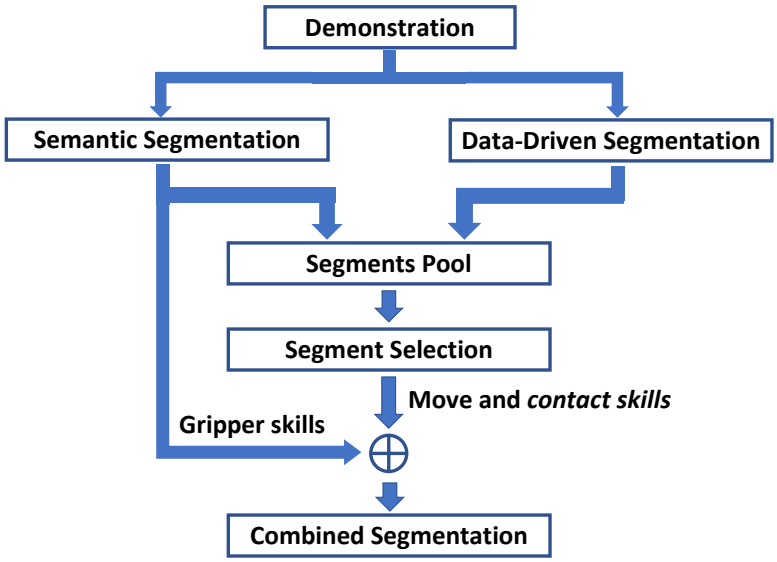


Figure 3.4: The flow chart describing how the semantic and the data-driven segmentation are combined. In the first step, the results of the semantic segmentation and the data-driven segmentation are combined in a segments pool. From that pool, a segmentation of the demonstration into Move and the *contact skills* is build. In the final step, the gripper based skills are directly taken from the semantic segmentation and merged into the previous result.

hierarchical approach was chosen as it allows to segment tasks where contact skills are performed continuously, without breaking contact in between the skills. An approach that first uses semantic segmentation and then a classifier on top would not be able to achieve this.

In a first step, the demonstration is segmented both semantically as well as data-driven, as can be seen in the first image (step 1) of Fig. 3.5. From the two segmentation results, two sets of possible segmentation lines are found. These mark the points in the demonstration where the segmentation results change from one skill to a new one. These segmentation lines will be referred to as keyframes, and an example can be seen in step 2 of Fig. 3.5. The keyframes also hold information about which skill caused the segmentation line, in the form of a skill label. From these keyframes, a pool of possible skill segments is built, as shown in step 3. The algorithm that creates this pool can be seen in Alg. 2.

This pool is created to find all possible combinations of the two segmentation approaches. By creating skill segments not only from the two approaches separately,

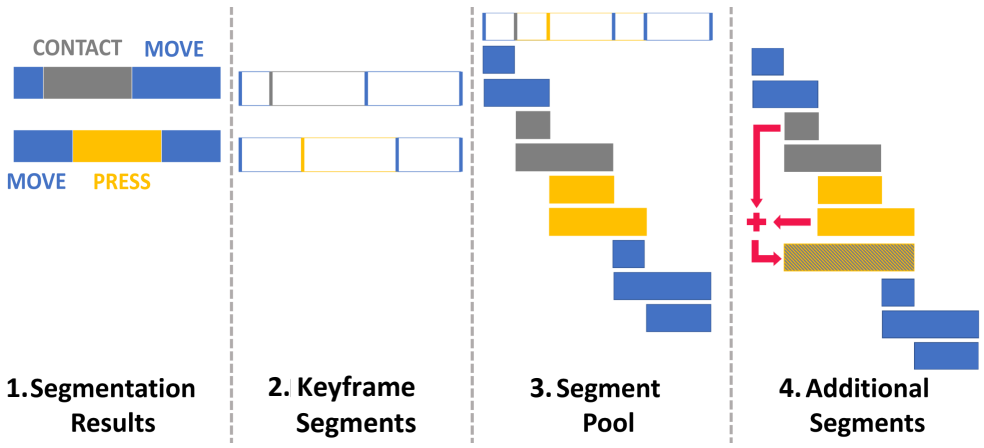


Figure 3.5: The process of building the pool of possible segments. In step 1, exemplary segmentation results for the semantic segmentation (top) and the data-driven segmentation (bottom) can be seen. In step 2, the keyframes k_s (top) and k_d (bottom) are shown. Step 3 shows how the keyframe segments are found from k_s and k_d. Step 4 shows how keyframe segments belonging to the same skill class are combined to extend the pool of possible segments.

but also from the combination of their segmentation results, the possibility of the correct segmentation lying in between the two results is also considered. This pool of segments is created by first iterating over the keyframes $k_{s_i} \in k_s$ from the semantic segmentation, taking them as the start point of the segments. The endpoint of each segment is found as the next keyframe $k_{s_{i+1}}$ of the semantic segmentation, as well as all keyframes of the data-driven segmentation $k_{d_j} \in [k_{s_i} < k_d < k_{s_{i+1}}]$ that lie between k_{s_i} and $k_{s_{i+1}}$. The segment is assigned the same skill label as the starting keyframe. This process is then repeated in the same way for the data-driven segmentation.
In the next step, the skill labels are used to find the skill class for each segment in the pool. The skill class differs between skills that are *contact skills*, and skills that are not. When two skills of the same skill class follow one another, an additional segment is added to the pool that spans over both existing segments. This is done to create segments for cases in which the keyframes of the semantic and data-driven segmentation are slightly shifted, thus creating the possibility that the starting keyframe of the one segmentation and the ending keyframe of the other segmentation is correct. An example for this case can be seen in step four of Fig. 3.5.
From the pool of possible segments, the final segmentation is found by a greedy search. Each segment obtains a score for each skill, using the results of the SVM's decision function. The SVMs used are identical to the SVMs used in the data-driven segmentation, except for having solely been trained on complete examples of the skills. As in this step of the segmentation process only whole skill demonstra-

Algorithm 2 Segment Pool

Input: Results for semantic segmentation S_s and results for data-driven segmentation S_d

```
segments = [ ]
keyframes_s ⟵ find_segmentation_points(S_s)
keyframes_d ⟵ find_segmentation_points(S_d)
for k_i ∈ keyframes_s do
    endpoints ⟵ all points in keyframes_d that lie between k_i and k_{i+1}
    for e_j ∈ endpoints do
        segments ⟵ add s = [k_i, e_j]
    end for
    segments ⟵ add s = [k_i, k_{i+1}]
end for
for k_i ∈ keyframes_d do
    endpoints ⟵ all points in keyframes_s that lie between k_i and k_{i+1}
    for e_j ∈ endpoints do
        segments ⟵ add s = [k_i, e_j]
    end for
    segments ⟵ add s = [k_i, k_{i+1}]
end for
for s_i ∈ segments do
    if skill_class(s_i) == skill_class(s_{i-1}) then
        segments ⟵ add s = s_i ∪ s_{i-1}
    end if
end for
```

tions are considered, there is no need to consider skill fragments.
From the pool of scored segments, the segment with the highest score is picked. The part of the demonstration belonging to that segment is then assigned the skill label that lead to this highest score. All other segments that overlap this segment are removed from the pool. This is repeated until the whole demonstration is segmented. Thus, the Move skills and the *contact skills* are found. In the next step, the Pick, Place, Open-gripper and Close-gripper skills are added. They are directly taken from the semantic segmentation and merged into the previous segmentation result, by overwriting what has been found in their place before. In the last step, all segments that are below a predefined minimum length are merged into the segments before and after, to reduce over-segmentation.
An example of the segmentation result of the combined segmentation can be seen in Fig. 3.6. The figure shows that the approach is able to recognize all skills correctly, even the Press and Slide skills performed right after one another.
A comparison of the segmentation results of the semantic segmentation, the data-driven segmentation and the combined segmentation can be seen in Fig. 3.7.

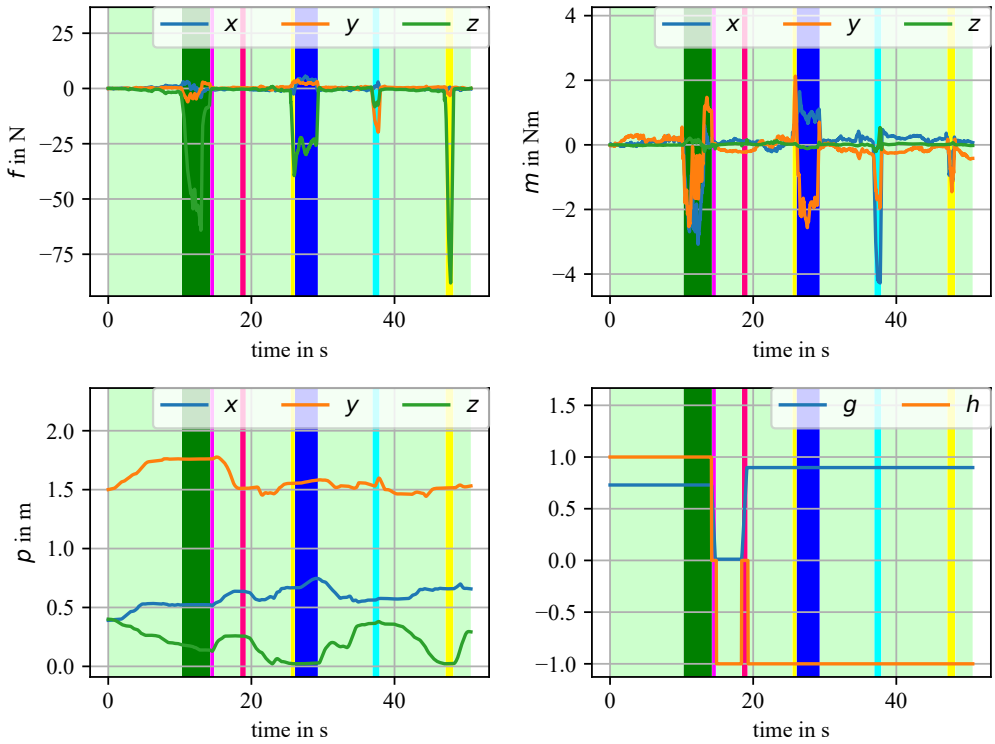


Figure 3.6: Exemplary segmentation result for the combined segmentation. For the skill sequence please refer to Fig. 3.3. The colours are: Move (light green), Peg-in-hole (dark green), Press (yellow), Slide (dark blue), User (cyan), Pick (pink), Close-gripper (red).

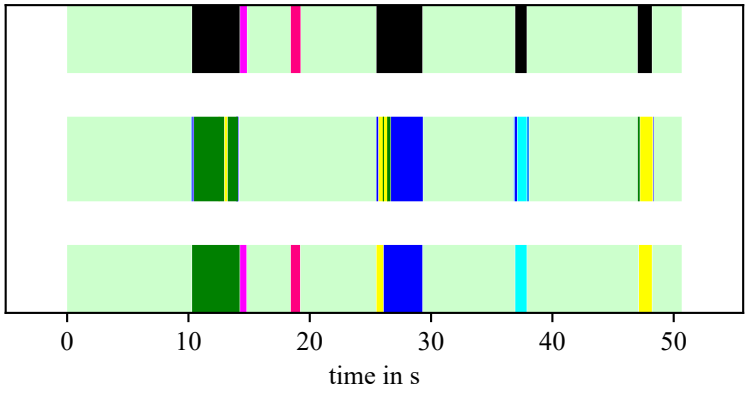


Figure 3.7: A comparison of the segmentation results for the semantic segmentation (top), the data-driven segmentation (middle) and the combined segmentation (bottom). For the meaning of the colours please refer to Fig. 3.6. Black indicates a general *contact skill*.

It shows how the combined segmentation is able to make use of the strength of both the semantic and the data-driven segmentation, by using the gripper based skills from the semantic segmentation and the *contact skill* recognition from the data-driven segmentation, without the issue of over-segmentation.

3.1.4 Adaption for Online Segmentation

As the previously described method of combining the semantic and data-driven segmentation requires the demonstration to be completed before it is run, the online segmentation is done in a two step manner. The flow chart of the online segmentation can be seen in Fig. 3.8.

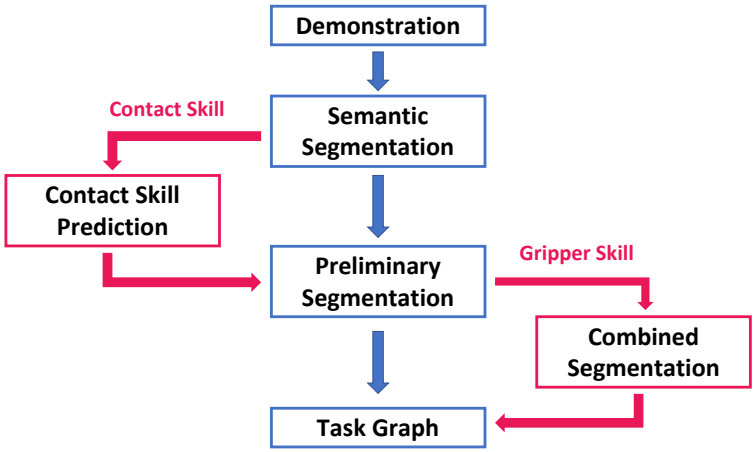


Figure 3.8: How the task-graph is built by the online segmentation. First, the semantic segmentation is used to find a preliminary segmentation result, that is shown as a task-graph. When the semantic segmentation detects a *contact skill*, the SVMs are used to predict the label of the *contact skill*. After a gripper based skill is detected, the combined segmentation is performed up until that point in the demonstration. As these two steps are conditional, they are shown in red.

In the first step, the semantic segmentation is used online to detect which skill is currently performed. When it is detecting a *contact skill*, the SVMs are used to resolve which *contact skill* it is. The detected skills are used to build the task-graph online. Once the semantic skill segmentation detects the end of a gripper based skill such as Pick or Place, an intermediate fixed segmentation line is found, as this line will not change during the rest of the segmentation process. Therefore, the part of the demonstration that ends with this line can already be treated as a complete demonstration, and thus be segmented with the combined segmentation algorithm. The results of the combined segmentation then replace the preliminary segmentation in the task-graph.

This leads to a constantly evolving task-graph, where the earlier skills of the demonstrations are already finalized while the most recent skills are the preliminary result. This continues until the demonstration is over. Finally, the demonstration can be segmented with the combined segmentation method as a whole.

3.2 Segmentation Results

The evaluation of the segmentation algorithm on its own is given in the following. The effectiveness of the skill segmentation as a means to share the robot's knowledge with the human teacher is investigated in the user study introduced in Chap. 5.
The segmentation algorithm was trained and evaluated on a data-set provided by three people. Each skill was demonstrated approximately 100 times. The segmentation 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 graphs depicting the measurements of the recorded demonstrations with the demonstrated skills.
The window length for the data-driven segmentation as well as the minimum length for a skill is chosen as 0.4s. The evaluation is based on two commonly used metrics [LKK16], 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. A high accuracy in temporal tolerance not only indicates a successful segmentation, but also measures how easily the segmentation result can be incorporated in a larger framework. When the start- and endpoints of segments are found reliably, the influence of miss-classification of skills becomes less drastic. In a framework as ours, the user is simply able to choose the correct skill label instead, without having to change the segmentation result any further.
The temporal tolerance is found by comparing the segmentation lines obtained from the combined segmentation algorithm to the segmentation lines of the ground truth. All algorithmic segmentation lines that lie in a region of $\pm t_{\text{err}}$ around the ground truth are considered True Positives (TP), while all algorithmic segmentation lines that lie outside of such a region are considered False Positive (FP). If a ground truth segmentation line has no corresponding algorithmic segmentation line within its region, it is considered a False Negative (FN). The region margin $\pm t_{\text{err}}$ was chosen as 0.2s in this evaluation. The overall accuracy is then computed as the F1-score as follows

$$F1 = \frac{2 * TP}{2 * TP + FN + FP} \ . \tag{3.1}$$

Our framework has an F1 score of 0.93. The number of cases of FN and FP are approximately equal. The FP mostly occurred at the beginnings and ends of *contact skills*, when the user started or finished the demonstration of the skill with more energy then she performed the rest of the demonstration with. This lead to the detection of Press or User skills before or after the correct skill detection, depending

on whether increased force or torque was measured.
The FN on the other hand had two causes. The first is a slightly delayed detection of the *contact skills*, as a certain level of force and torque needs to be crossed before the skills are detected. The second cause is incomplete compensation of the tool weight, leading to small torque measurements when the tool is not held perpendicular to the workspace. This caused the algorithm to occasionally fail to detect Move skills in between two Contour skills.
The second metric, classification by data point label, evaluates how many of the data point labels l have been chosen correctly. From this, a confusion matrix can be calculated showing what percentage of ground truth label l_{GT} have been labeled with which label l_{A} by the algorithm

$$C_{g,a} = \sum_{i,j} \frac{I(l_{\mathrm{GT}_j} = g, l_{\mathrm{A}_i} = a)}{I(l_{\mathrm{GT}_j} = g)} \,, \tag{3.2}$$

were g is the ground truth skill label and a is the algorithmic skill label. The indicator function $I(\cdot)$ outputs 1 if the argument is true, otherwise zero. The confusion matrix of our approach is shown in Fig. 3.9. The skills that are most often confused

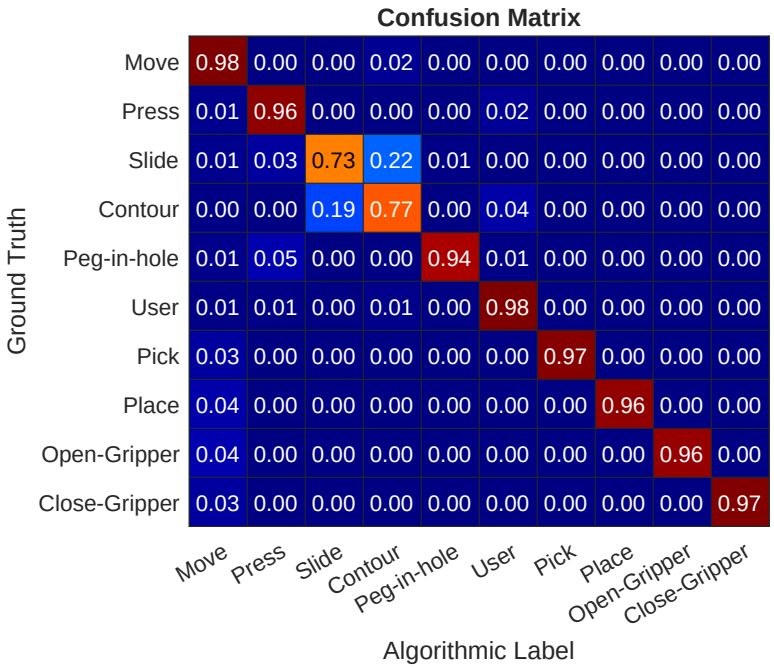


Figure 3.9: The confusion matrix for our approach.

are Slide and Contour. The Contouring skill is used to follow the outline of an object, while the Slide skill is used to slide over the planar surface of an object. Depending on how the gripper is positioned while performing these skills, the torque readings during the demonstration become almost identical, leading to the confusion.

Furthermore, the semantic segmentation detects the start of gripper based skills slightly too late and the end slightly too early, as the *GripperMoving* flag is set after the gripper fingers have already started moving.

Chapter 4

Interactive Task Programming Framework

As explained in Sec. 2.3, an effective PbD framework does not only need a reliable way to encode the robot's knowledge, but also a good communication channel between robot and human to share the robot's knowledge with the human. This ensures that the human teacher correctly understands what the robot has learned from the demonstration. Thus she can comprehend how her teaching influences the robot's knowledge, improving her ability to be a good teacher. Furthermore, by providing an immediate feedback about what the robot is learning during the teaching, she can detect potentially harmful errors in the robot's knowledge before they lead to critical damage. And even in minor cases of incorrectly learned tasks, an understandable knowledge representation helps the user detect errors faster and with less effort.
Therefore, the focus of our framework lies on providing the user with a suitable representation of the robot's knowledge, to improve the teaching experience for the human and to support the user in the challenge of detecting errors in the robot's knowledge. Thus, the skill based task-graph introduced in the last chapter is the center piece of our PbD framework, called Interactive Task Programming Framework (InTAP). The graph is used on the one hand to inform the user about the robot's state of mind, and on the other hand to give the human the option to correct the robot's knowledge if necessary.
InTAP consits of a backend, running the necessary processes for PbD and skill segmentation, as well as a GUI, designed to allow the user to interact with the task-graph, creating the interface between the human teacher and the robot's knowledge. The GUI runs on a tablet, which lies next to the user during the teaching process to allow easy access at all times. Through this GUI, the user is guided through the whole teaching process, by using different modes for the different stages of the teaching process. An image of the GUI at the beginning of the teaching process can be seen in Fig. 4.1. The different modes of the teaching process are introduced in more detail in the following.

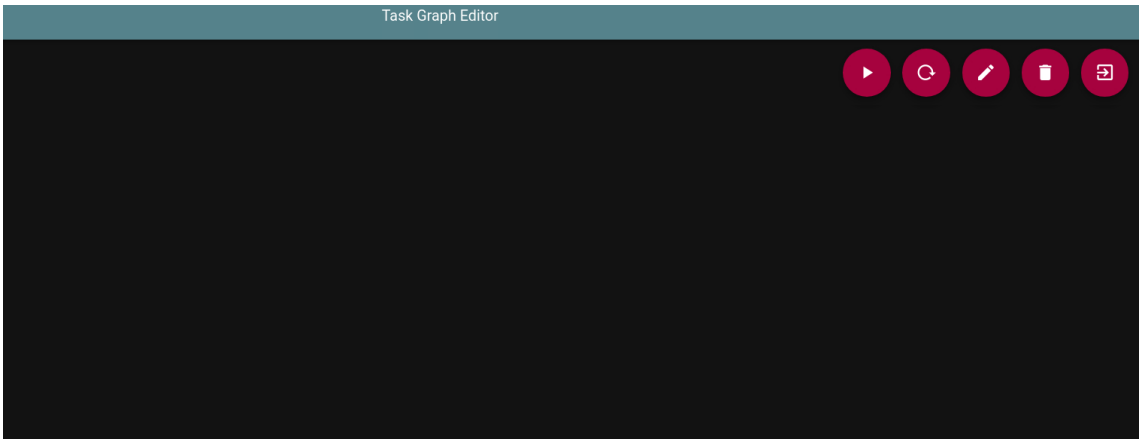


Figure 4.1: The GUI at the beginning of the teaching process. The buttons in the top right corner start the different modes of the teaching process. The PLAY-button starts the execution mode, the RELOAD-button reloads the graph from the backend, the PENCIL-button starts the editing mode, the DELETE-button deletes the whole graph and the EXIT-button quits the application.

4.1 Teaching Mode

The teaching mode allows the user to demonstrate new tasks to the robot. In the beginning of the teaching session, when no demonstration has been given yet, the PLAY button triggers the teaching mode directly. Once the user started the teaching mode, she can use the pedals placed next to the robot to start the gravity compensation. The set-up of the workspace is shown in Fig. 4.2. In gravity compensation, the user is able to guide the robot along the trajectory required to perform the task. To open or close the robot's gripper, the user can trigger the right-most pedal. During the demonstration, the robot's cartesian position as well as the forces and torques measured with a FT-sensor above the robotic gripper are recorded. These are then forwarded to the online segmentation algorithm. The thus build task-graph is shown to the user on the GUI.
The different stages of the online algorithm are represented by the colour coding of the skills in the task-graph. This is done to give the user a feedback regarding the reliability of the presented segmentation result. The first step of the online segmentation, the preliminary segmentation result taken from the semantic segmentation, is shown in green. This indicates that while the result is likely to stay the same, it might still change. When the online segmentation detects a *contact skill*, the skill is coded in yellow while it is demonstrated, to indicate that the predicting result is likely going to change while it is demonstrated. Once the user has moved on to demonstrating the next skill and the prediction result of the *contact skill* will not change anymore during the preliminary segmentation, it changes to green.
Once the preliminary segmentation detects a gripper based skill, the combined segmentation is performed on the demonstration up to the end of the gripper based

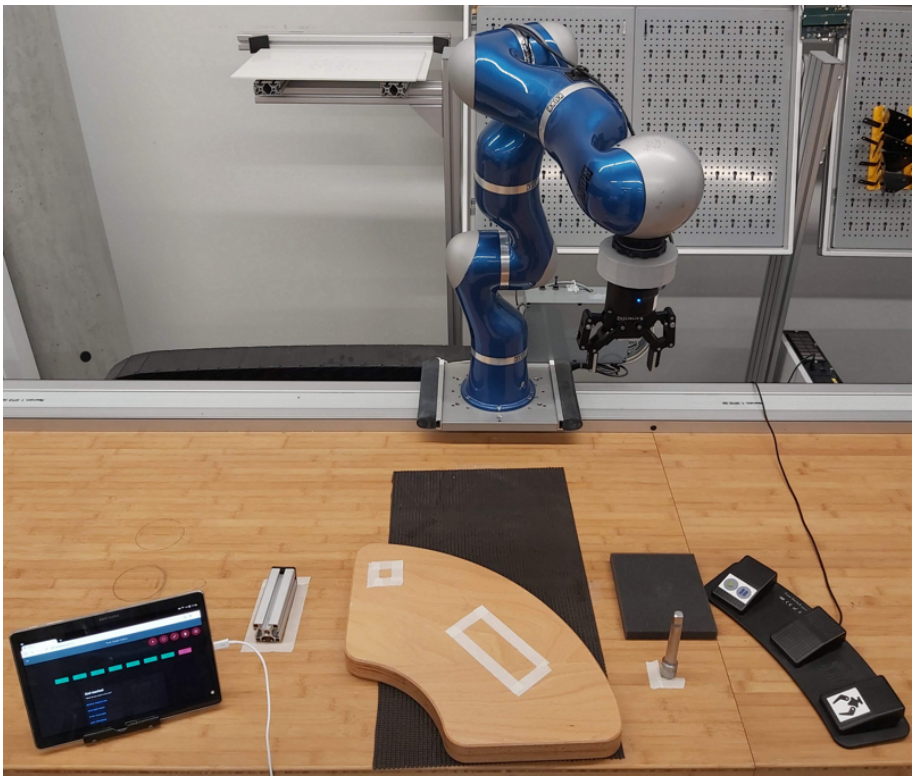


Figure 4.2: The workspace. The GUI is running on the tablet to the left of the robot. The pedals to start the gravity compensation and to open and close the gripper are to the right of the robot.

skill. The result of the combined segmentation will not change anymore during the rest of the teaching, therefore it is coded in the final colour turquoise. Examples for the colour changes in the task-graph can be seen in Fig. 4.3, where the three stages *currently predicting contact skill*, *preliminary segmentation result* and *combined segmentation result* can be seen.

Once the user is done giving the demonstration, she can press the gravity compensation pedal to end the recording. Then the robot learns how to perform the task from the given demonstration. The learning mechanism is taken from [WEL20], which first replays the users demonstration to record the robot's execution of the learned task. Then it uses GMM to encode the user's demonstration and the robot's demonstration together. The means of the GMMs are then used to execute the task. This learning mechanism is used to learn how to execute each skill separately.

After the robot has learned how to perform the task, the user is given the choice to either watch the robot execute the learned program, to quit the whole teaching session or to end the teaching mode in order to interact with the task-graph.

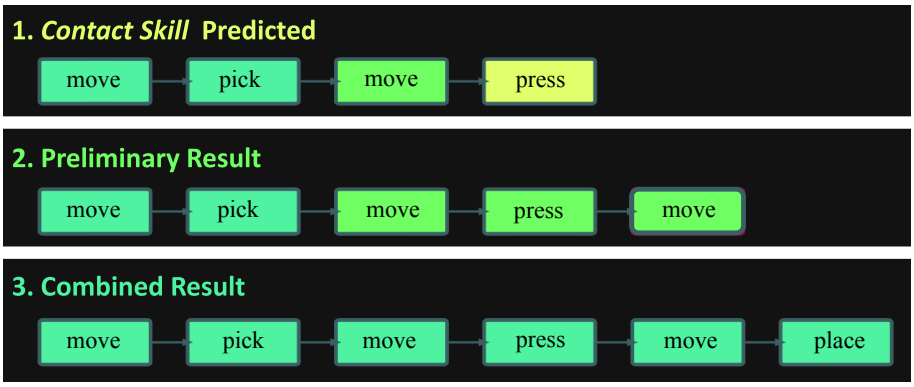


Figure 4.3: An example for the evolving task-graph. The different colours indicate which step of the online segmentation a skill has already passed. Yellow: SVMs currently predicting the label for the *contact skill*, green: the preliminary segmentation is done, turquoise: the combined segmentation is done.

4.2 Execution Mode

Once the robot has learned a task, the user can let it execute the task by pressing the PLAY-button. The robot then performs the skills in the task graph based on what it learned during the demonstrations. The skill currently executed by the robot is marked in red in the task-graph, showing the user how the task-graph relates to the actual execution of the task.
Instead of starting the execution of the task from the beginning, the user also has the option to start the execution from any arbitrary skill in the task-graph. This is done by first choosing the skill by clicking on it, and then pressing the PLAY-button. As there is currently no check for the necessity of objects for the execution of skills, it is left to the user to start the program only from skills that don't require the robot to hold an object.
After the robot finished executing the last skill in the task-graph, the robot remains in the position the skill ended in. From there, the user is able to give a new demonstration to extend the task-graph, which starts the teaching mode from the current point in the task-graph. Alternatively, the user can stop the execution mode and move back into the main menu.

4.3 Editing Mode

The editing mode is started when the user presses the PENCIL-button in the main menu. The task-graph can then be edited, either by changing skill labels or by deleting parts of the task-graph. The selecion of a skill, the changing of its skill label and the deleting dialog can be seen in Fig. 4.4. As the knowledge of the skill's label and the skill's execution are so far kept separately in our framework, the change of a skill's label has no effect on how the skill is executed. The delete option in the

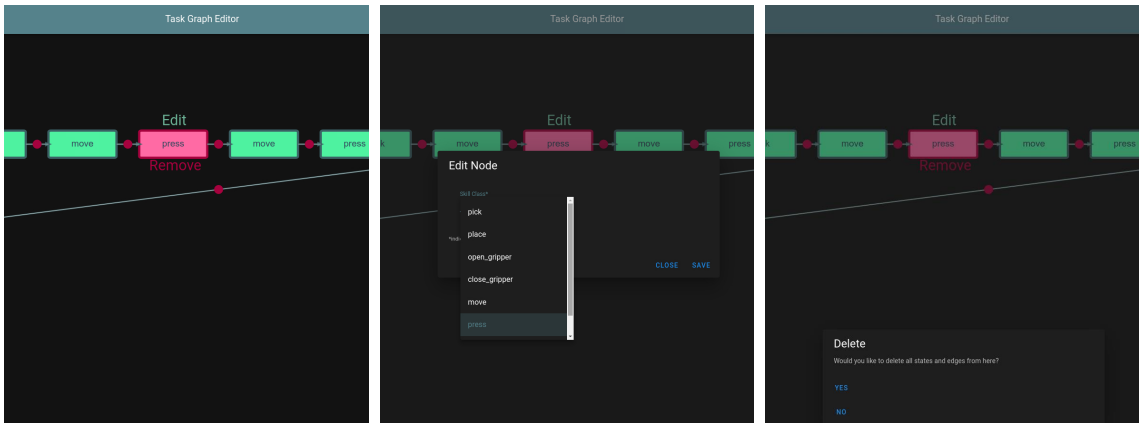


Figure 4.4: The editing mode. The image to the left shows a skill selected in the editing mode. The image in the middle shows the change-skill-label dialog. The image to the right shows the delete-partial-graph dialog.

editing mode allows the partial deletion of the task-graph. The user can choose to delete the end of the task-graph, including the selected skill. This allows the user to keep the beginning of a task, and only re-demonstrate the end of the task, starting from the first incorrect skill. The deletion of the task-graph is intended for cases where the demonstration can not be correct by a reassignment of skill labels, for example when the teacher made a mistake during the teaching process.

4.4 Reload Graph

The RELOAD-button has two functions: The first is to reload the graph from the backend, for example when the user refreshed the frontend. The second is to give the user the option to reduce over-segmentation. When a skill is recognized during the teaching process, that the user had not intended to demonstrate, for example a Press skill before a Place skill, the RELOAD-button can be used to remove the surplus skill. This is done by first changing the label of the surplus skill to that of the skill it was intended to be part of. Then, the user can press the reload button to merge the two skills together. This merging process goes over the whole task-graph and searches for parts where two skills with the same label are performed consecutively. If it finds two such skills, it merges them into one continuous execution of the skill. The additional step of using the RELOAD-button to merge skills was chosen to avoid a scenario where the user mistakenly changes a skill label, so that two skills of the same label follow one another, and thus merges them by mistake.

4.5 Delete Graph

The DELETE-button can be used to delete the whole task-graph. This can be necessary when errors during the teaching occurred early in the demonstration. Then it can be easier for the user to demonstrate the whole task from the beginning, rather than to find the best part to partially delete the task-graph.

4.6 Quit Application

The EXIT-button ends the teaching session and stops the frontend. The program taught during the teaching session is saved, allowing the user to rerun the execution or to continue teaching anytime.

Chapter 5

User Study

To evaluate the intuitiveness and suitability of our suggested task-graph for PbD settings, we conducted a user study. It consists of a laboratory study, where 6 participants programmed a robot with the help of our framework, as well as an online study where 26 users were asked to evaluate the comprehensibility of our approach. To allow a combined evaluation of the laboratory study and the online study, both were designed to be as similar as possible.

5.1 Compared Methods

To validate whether the skill based task-graph description of our approach outperforms simplified action-based task-graphs, we compare the representation of our approach to the time-line representation used in [IØRS17]. It 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-graph describes the learned task through a time-line, showing the detected points as well as gripper movements. To allow a fair comparison, we also use the segmentation found in our approach as basis for the time-line based task-graph. Thereby, the segmentation result is shown without skill annotation and only the openings and closings of the gripper are displayed. The same task-graph in both representations can be seen in Fig. 5.1. The time-line based task-graph simplifies the gripper based skills to the actions of open and close, thus removing the additional knowledge used in the skill recognition to differ between Pick/Place and Close-gripper/Open-gripper. The *contact skills* used in the experiments were limited to Press and Slide. This was done to decouple the classification results from the evaluation of the task-representations.

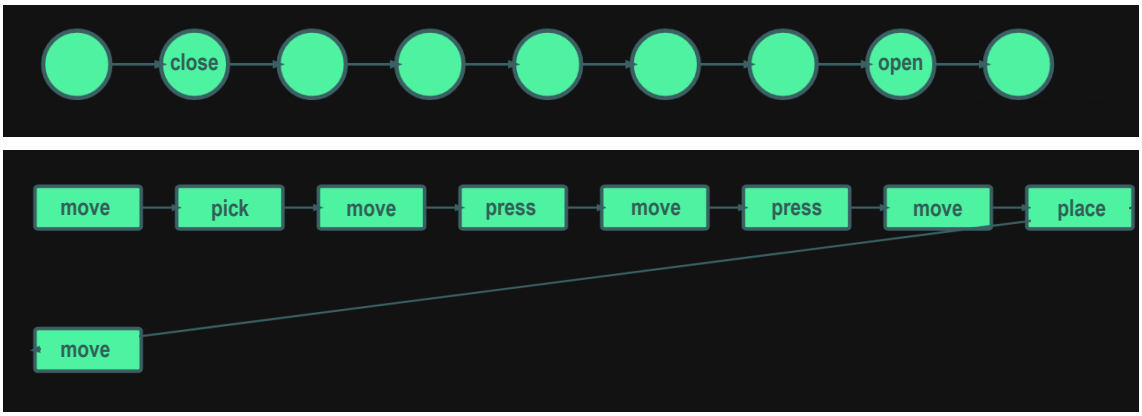


Figure 5.1: The upper part of the image shows the stamping task in the time-line representation, the lower part shows the same task in the skill based representation.

5.2 Hypotheses

The user study investigates the effect of robot knowledge representations on the teaching process, with focus on the human teacher. To evaluate how suited the task-graph representations are to serve as a communication of the robot's knowledge to the human teacher, the user study investigates different aspects of knowledge representations that contribute to their effectiveness. The aspects chosen are: the workload when teaching with the task-graph representation, match of the user's mental model to the robot's actual state of mind and the user's trust in the robot. These categories are chosen based on the challenges a successful knowledge representation needs to overcome, explained in Chap. 4. For humans to be willing to use a PbD framework, the workload must be small and the humans must trust that the robot learns from it correctly. Furthermore, the users must have a correct mental model of the robot's state of mind, for them to be able to be good and effective teachers. Thus we aim to prove the following hypotheses with the user study:

H1) The skill based task-graph representation makes it easier for the human teacher to see at first glance if the robot has learned the demonstrated task correctly, thus significantly reducing the teacher's workload during the teaching process compared to the time-line based task-graph representation.

H2) The skill based task-graph representation gives the user more feedback and control over the robot's knowledge, thus significantly increasing the human's trust in the robot, compared to the time-line based task-graph representation.

H3) The skill based task-graph representation explains the robot's knowledge intuitively understandable to the human teacher, thus helping the user to perform better as a teacher, compared to teaching with the time-line based task-graph representation.

5.3 Task descriptions

To be able to evaluate the hypothesises, the participants were asked to solve different problems based on two different tasks. The first task is a *stamping task*, shown in Fig. 5.2. There, the user takes a stamp, presses it into an ink pad and then stamps

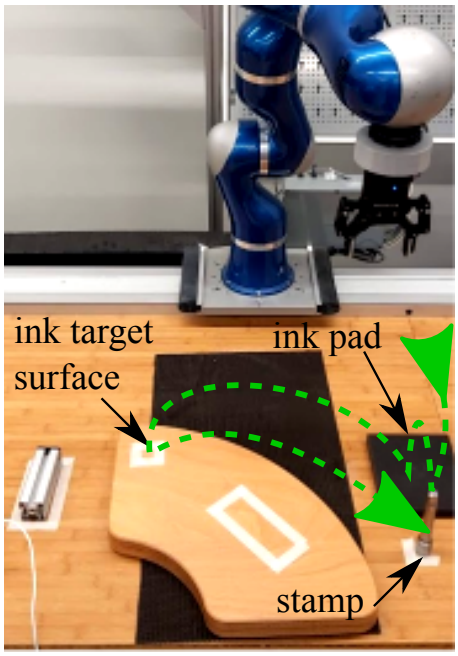


Figure 5.2: The stamping task used in the user study.

in a marked area before placing the stamp back. Furthermore, an incorrect version of this task was used as well, where the stamp is replaced immediately after pressing it in the ink pad, without stamping the marked area.
The second task is an assembly task, shown in Fig. 5.3. There, the user takes a glue-stick and applies it in a marked area by sliding it over the surface. Then she takes a second object, places it on top of the marked area and attaches it by pressing down on it. The assembly task also comes in an incorrect version, where the second object is not taken and placed on the marked area, so that the robot presses down on the surface instead of the second object.
To avoid accidents, the ink and glue were substituted by stand-ins.

5.4 Laboratory Study

In the laboratory study, 6 participants from the German Aerospace Center tested the InTAP framework and programmed a robot both with the skill based task-graph (SBTG) representation and the time-line based task-graph representation (TBTG). All participants had previous experience in working with robots. The robot used in

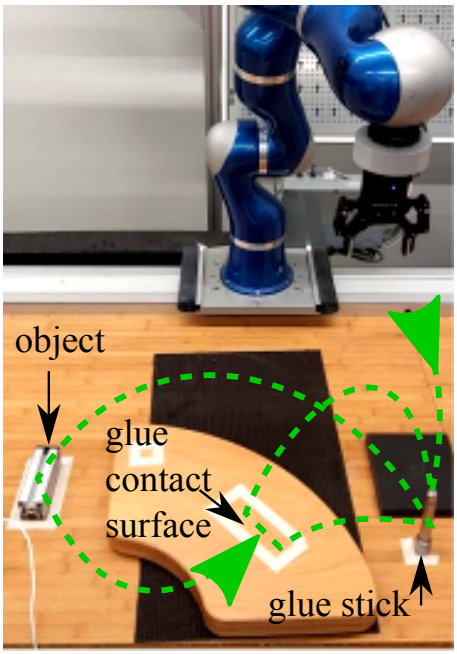


Figure 5.3: The assembly task used in the user study.

the study was a DLR LWR IV [ASHO+07] robot, which was mounted on a linear axis for wider reach. It was equipped with a "Robotiq 85" 2-finger gripper that was attached to a wrist-mounted FT-sensor, measuring the forces and torques that were acting on the gripper. The user could physically guide the robot in gravity compensation mode to teach the tasks, and use a button to open and close the gripper.
The experiments were conducted with a within-subject design, so that half of the participants first used the SBTG and the other half the TBTG. At the beginning of the study, the users were informed about the goal of the study and the procedure of the experiment. Then, they were given the chance to familiarize themselves with PbD, by allowing them to guide the robot in gravity compensation. At this point they were not shown the task-graphs yet. After they were satisfied with their ability to guide the robot, they were shown a short video explaining how the framework could be used with the respective task-graph representation they were first using. Afterwards, they could ask questions regarding the use of the framework.
Then the first block of the experiments began, in which the participants were asked to teach the robot new tasks from scratch. Therefore, they were shown a video explaining the task they were asked to program, before they taught the robot the new tasks. The first task the users were asked to program was the *stamping task*, the second the *assembly task*.
After the participants were satisfied with their results, the second phase of the experiments began. There, the participants were told colleges had taught the robot the same tasks, and the users were asked to verify if they had done so correctly. First, the participants were asked to base their opinion about the correctness of the

program solely on the task-graph showing the task in the currently used representation. Then they were allowed to watch the robot execute the program as well. If they found the program to be incorrect, they were asked to correct it. The participants were shown two examples of programmes, one correct and one incorrect. The order of the two examples was chosen randomly.
After the debug tasks, the users were asked to fill out questionnaires, rating their experience with the respective knowledge representation. The NASA-TLX was used to measure the subjective workload of teaching the robot with the knowledge representation and the ISO 9241-110 questionnaire for Dialog Principles was used to evaluate the information transfer from robot to human. To further investigate the explainability of the knowledge representations, based on [She17], 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 questions can be found in the Appendix.
After the participant filled out the questionnaires, the experiments were repeated with the other knowledge representation.

5.5 Online Study

The online study was designed as an extension of the laboratory study. To reduce the required time for answering the online study, two identical studies were created, one for each of the task-graph representations. The participants were then split in two groups, each only seeing one of the task-graph representations. Each group consisted of 13 participants, of whom 75% had no background in robotics.
In the studies, the participants were first informed about the goal and the procedure of the online study. Then they were shown a short video explaining how the framework uses the respective task-graph representation. As the participants would not be programming a robot, the details of how the task-graph can be manipulated were omitted.
Since the subjects could not program a real robot, the first set of experiments was changed. First, the participants were given the beginning of the *assembly task* as a step-by-step image series, the part where the glue is applied to the surface. Then they were shown a video of a robot correctly executing this task, together with the GUI depicting the task-graph. Then they were asked if they understood how the task-graph describes what the robot has learned. For the second task of the first set, the participants were given the *stamping task* and two graphs, one showing the correct *stamping task*, the second the incorrect *assembly task*. Then they were asked to choose which of the two graphs belonged to the *stamping task*. Furthermore, they were asked if they found it easy to choose their answer, as well as what they based their decision on.
The second set of the experiment also consisted of a debugging task. The participants were given the description of the *assembly task*, and the task-graph showing

the incorrect *assembly task*. Then they were asked if the task-graph performed the *assembly task* correctly. The participants could choose between *Yes*, *No* and *Not sure*. Then they could watch the robot execute the program, the same way the participants of the laboratory study could. Then the participants were asked to choose the point in the task-graph, from which on they would delete the graph in order to correct the program.
Finally, to evaluate how suited the task-graph representations are for the tasks, the participants were given the the ISO 9241-110 questionnaire for Dialog Principles and the questions regarding the explainability of the task-graph.

5.6 Results

The results of the participants' answers to the questionnairs are presented in the following. As the NASA-TLX questionnaire could only be applied in the laboratory study setting, the results of the NASA-TLX questionnaire contain only the answers of the laboratory study participants. For the ISO 9241-110 questionnaire and the general questions, the answers from both the laboratory study and the online study are combined.
The total result of the NASA-TLX questionnaire can be seen in Fig. 5.4. The NASA-

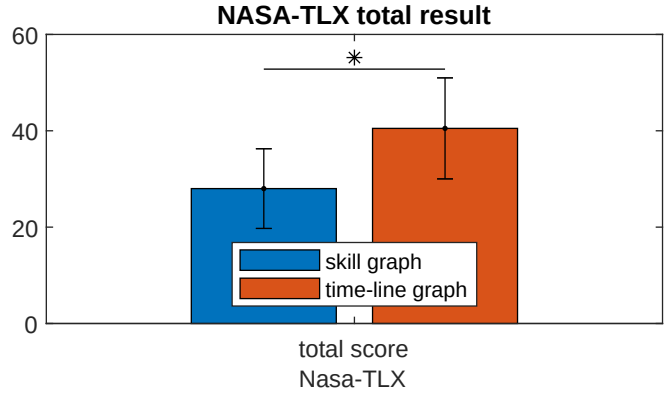


Figure 5.4: The total result of the NASA-TLX questionnaire answered by the participants of the laboratory study. The users score the categories on a scale from 1 (best) to 20 (worst). The scores of the individual categories are then added up to obtain the total score. The plot shows the total score for the SBTG used in our approach (blue) and the TBTG (red). The $\star$ marks that a p-value of $p < 0.05$ was found with a Wilcoxon signed rank test.

TLX measures the workload which the users experience when working with the two knowledge representations. To measure this workload, the NASA-TLX splits the workload in different categories, as shown in Fig. 5.5. Both the total result of the NASA-TLX as well as the single categories show a clear reduction in workload for the SBTG compared to the time-line task-graph. The significance of this reduction

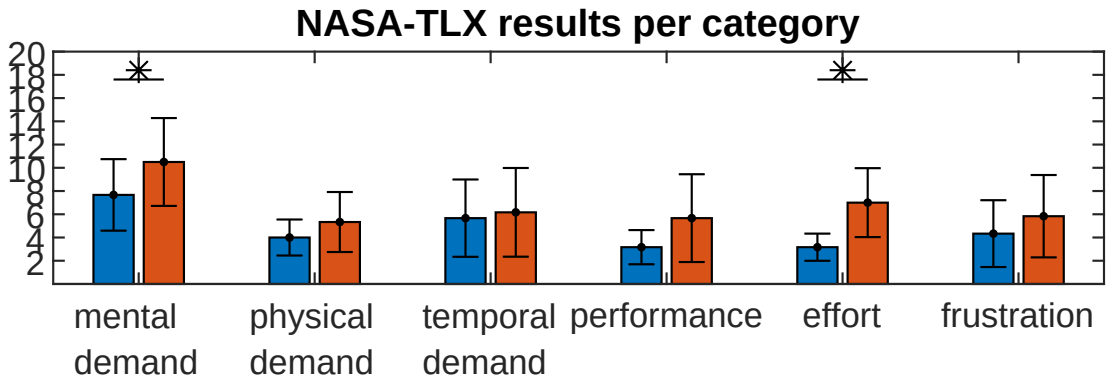


Figure 5.5: The individual results for each of the categories of the NASA-TLX questionnaire answered by the participants of the laboratory study. The users score the categories on a scale from 1 (best) to 20 (worst). The plots show the mean of the scores given for the individual categories. For details about the $\star$ and colours please refer to Fig. 5.4.

is supported by a Wilcoxon signed rank test for the total test result, as well as the categories *mental demand* and *effort*. The reduction of the perceived mental workload is 27% for the SBTG compared to the time-line based task-graph, and the reduction in effort 54%. In the categories *physical demand* and *temporal demand*, the difference between the two representations is the smallest, as the PbD framework used for both is the same. In the categories *performance* and *frustration*, the SBTG also outperforms the TBTG, but the Wilcoxon signed rank test could not support the significance of these results. Yet an overall trend towards the reduction of the experienced workload for the SBTG representation is clearly visible in all categories. This trend is also reflected in the results of the online study. There, the participants were asked after the first task set if they found it easy to find the task-graph belonging to the given task. The answers can be seen in Fig. 5.6. It shows that for

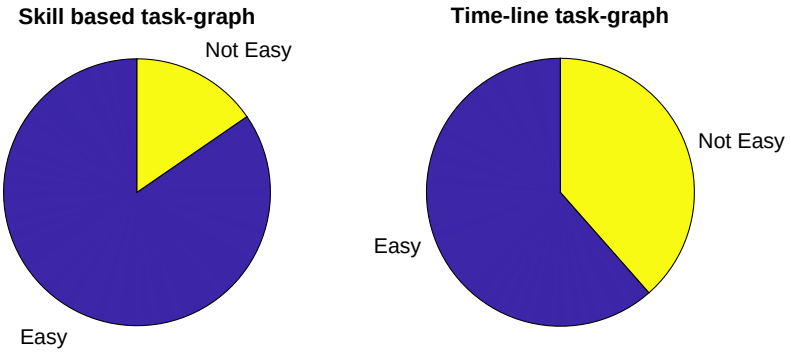


Figure 5.6: The answers of the participants of the online study regarding if they considered it easy to find the matching task-graph to a given task description.

the SBTG, 85% of the participants found it easy to find the matching task-graph, while for the TBTG it is only 62% of the participants.
The intuitiveness and usability of the representations are compared in Fig. 5.7, where the results of the ISO 9241-110 questionnaire can be seen. On average, the

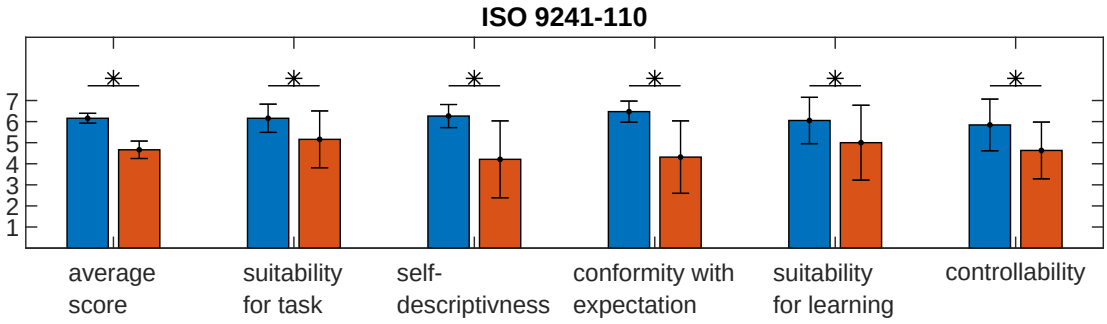


Figure 5.7: The combined results of both the online and the laboratory study for the ISO questionnaire. The scores rate how much the users agree to the category from 1 (Strongly Disagree) to 7 (Strongly agree). The higher the result, the better. For details about the $\star$ and colours please refer to Fig. 5.4

users rated the usability of the SBTG representation 32% better than the time-line representation. Especially in the categories evaluating the intuitiveness of the representation, the SBTG drastically outperforms the TBTG: The self-descriptiveness of the SBTG was rated 48% better and the conformity with the user's expectation 50% better. In the remaining categories, the SBTG representation also outperforms the TBTG significantly, as shown with a Wilcoxon signed rank test.
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. 5.8 are grouped in the aforementioned categories used to evaluate the explainability of the task-graph representations. In the debugging category, the users were asked to rate their ability to detected errors in the robot's learned program with the help of the task-graph. From the results, a clear preference for the SBTG can be seen. The users' self-evaluation is also confirmed by the number of the times the users were able to predict if a program would perform a task as desired, for which the results can be seen in Fig. 5.9. It shows that 84% of the participants were able to correctly identify a task as incorrect with the help of the SBTG, whereas only 64% could do so with the TBTG.
In the categories *match of mental model* and *explainability* of the explainability questions, the SBTG also out-performed the TBTG drastically.
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 knowledge representation.
In a final oral interview in the laboratory study, the participants showed a clear preference for the SBTG. When asked for their first impressions, half of the participants found the time-line representation easier to understand when seeing it for the

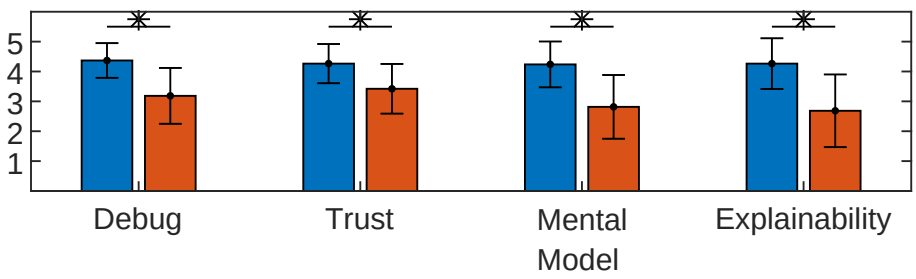


Figure 5.8: The combined results of both the online and the laboratory study for the explainability questions. For details about the ⋆ and colours please refer to Fig. 5.4. The questions were grouped in the categories 'Ease of debugging', 'Trust in the robot', 'Match of the user's mental model to the robot's real knowledge' and 'Explainability of the knowledge 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.

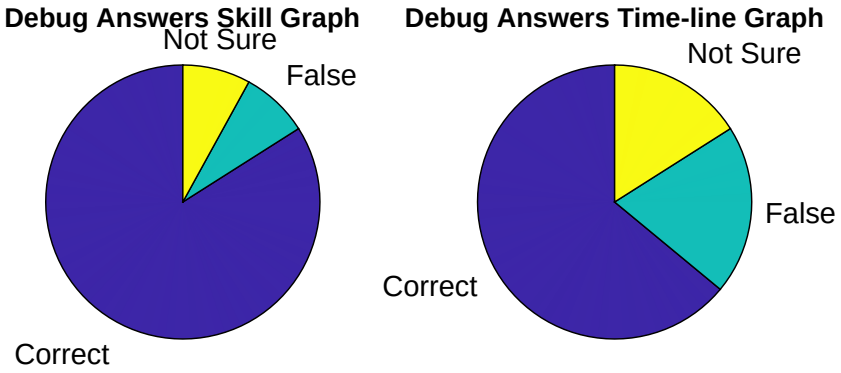


Figure 5.9: The results of the debug tasks. The participants were asked whether a program would execute the task correctly. The diagrams show the percentage of correct, false and undecided answers for the SBTG (left) and the TBTG (right).

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 they found easier to work with.

5.7 Discussion User Study

The results presented in the last chapter show the participants' clear preference for the SBTG. How the results relate to the hypotheses introduced in Sec. 5.2 is explained in the following.

5.7.1 Workload

For users to be willing to work with a PbD framework, the workload of using this framework must be low, especially in a framework designed for non-expert users. Thus, hypothesis H1 states that the use of a SBTG representation achieves this necessary low workload compared to the TBTG. The results shown in Sec. 5.6 prove this hypothesis. For the user to be able to intuitively interact with the task-graph, it needs to be self-descriptive, thus reducing the mental workload needed to interpret it. This is achieved in the SBTG representation by breaking the task down into steps that are self-descriptive, the skills. The meaning of the skills can be inferred from the use of their names in everyday life, thus only little effort is necessary to understand what the SBTG describes. As the steps depicted in the TBTG are unrelated to the actions the robot performs, it is more difficult for the user to understand how the task-graph describes the learned task. This is mirrored in higher results for self-descriptiveness for the skill based representation, as well as a reduced mental workload, compared to the time-line based representation.
This also leads to a reduced effort for teaching the robot new skills, as shown in the *effort* category of the NASA-TLX, as well as in the comments from the online study. When the participants of the online study were asked how they knew which task-graph to choose in the matching task, most participants using the SBTG could infer the correct answer from the performed skills. As the task for which they were asked to find the task-graphs for was the stamping task, they easily rejected the alternative choice, the assembly task graph, by recognizing that the stamping task does not require a Sliding skill. The participants using the TBTG on the other hand had to go through both task-graphs step by step and find the correct answer from the number of gripper actions. This requires a higher concentration on the task and is more error prone, as shown by the number of incorrect debug answers shown in Fig. 5.9.
The higher error rate leads to a perceived worse performance of the TBTG, as shown by the trend in the *performance* category of the NASA-TLX questionnaire. The combination of a higher mental workload and lower performance leads to frustration, which is also supported by trend in the *frustration* category of the NASA-TLX. As the purpose of a PbD framework is to create an intuitive and enjoyable teaching environment for the human teacher, a SBTG thus clearly outperforms a TBTG, proving hypothesis H1.

5.7.2 Trust

An essential criteria to allow the wide spread use of robots in the industry, is the trust of the human workers in the robots. This trust in the context of PbD mostly depends on the human trusting that the robot learns how to perform the tasks correctly. Without this trust, the human teacher can not rely on the robot to perform its tasks unsupervised, and the desired substitution of robots in highly

physically demanding and repetitive tasks can not take place. Thus hypothesis H2 states that the SBTG creates the base-line to foster this necessary trust, while the TBTG does not.
This is reflected by the results of the user study. The participants of both the laboratory study and the online study rated the suitability of the SBTG to explain the robot's knowledge significantly higher than the suitability of the TBTG. By ensuring that the user feels that the robot's knowledge is communicated adequately, the user is more likely to rely on the robot's feedback. This is reflected by the number of participants that felt unwilling to give an answer in the debugging tasks, as shown in Fig. 5.9. For the SBTG, only 8% of the participants felt insecure about their choice and preferred not to answer, while for the TBTG the number of participants with 16% was twice as high. This shows that the users felt sufficiently informed about the robot's knowledge, when using the SBTG, to base their decisions on the robot's feedback, indicating that they trust that they understood the robot's feedback correctly.
Furthermore, as PbD frameworks are foremost designed to support non-expert users in teaching a robot new tasks, most users have little experience in this field and are forced to learn how to perform their job in an unfamiliar environment. Thus, the first step to build the users' trust in the robot's ability to learn correctly, is for the users to build the trust in themselves to teach correctly. Therefore, a PbD framework should give the user this trust by supporting them in their role as a teacher. Thus, the suitability of a knowledge representation for helping the user learn how to be a good teacher is essential. The participants rated the suitability of the SBTG for learning 21% higher then the TBTG, thus supporting that the SBTG representation is more suited to build the teacher's trust in their own abilities. It achieves this by providing more information about what the robot is learning from a demonstration, thus giving the human teacher an immediate feedback about the consequences of their teaching. Thus, the user is able to learn faster how to reach their desired goal, which builds the confidence in their own teaching ability that in turn is the base-line for trusting the robot.
For humans to trust the robot to learn a new task correctly, they must also feel in control of the robot's knowledge. As the robot learns how to perform the new tasks via a knowledge transfer from human to robot, the human must feel in control of this transfer to be comfortable to let the robot execute the task on its own. Thus, the robot's knowledge must be represented in a way that gives the human the possibility to completely control it. The participants rated the controllability of the SBTG at 26% higher than that of the time-line based task graph. This is due to the larger number of manipulation options given by the SBTG representation. By knowing the exact skill that is recognized in each step of the task, the user can correct the labels of skills and merge surplus skills, without having to give new demonstrations. From the task-graph alone, the user can then see if the desired result is reached. For the TBTG, the user must instead watch the robot execute the task, and estimate with the help of the task-graph if the robot learned everything

correctly. If the users feel that the robot has learned the task incorrectly, they have to give a new demonstration for the part they doubted, solely able to influence the robot's knowledge through the same channel that lead to the wrong result before. Thus, by giving the user multiple options to correct the robot's knowledge, the user can trust that the robot learned the task correctly for the SBTG representation.
That the aforementioned points lead to a stronger trust in the robot, is shown by the participants rating their trust in the robot at 25% higher when working with the skill-based task-graph representation compared to the time-line based representation, proving hypothesis H2 to be correct.

5.7.3 Teaching

As discusses in Sec. 2.3, the representation of the robot's knowledge does not only influence how comfortable the user is during the teaching process, but also which results she can achieve. An intuitively understandable knowledge representation helps to close the gap between what the human assumes the robot learned and what the robot really learned, thus making it easier for the human to notice mistakes in the robot's program and to correct these. This is addressed by hypothesis H3, stating that the SBTG representation helps to close this gap and therefore helps to improve the user's teaching ability compared to the TBTG representation.
Therefore, the robot's knowledge representation should match the user's mental model of how the robot is learning, making it easier for the user to understand the knowledge representation and reducing the possibility of mismatches. Thus, a conformity with the user's expectation is essential. The participants of the user study rated the conformity of the SBTG 50% higher than the conformity of the TBTG. This is due to the SBTG breaking down the robot's knowledge in the same way a human would, by identifying the different skills necessary to perform the task. A TBTG on the other hand uses different methods to encode a task, thus making it harder for the human to follow which steps the robot identified as necessary to perform the task. This influences the user's ability to notice mismatches between what she intended to teach and what the robot learned. In the laboratory study, most participants set objects down forcefully, leading to the detecting of a Press skill before the intended Place skill. While this surplus skill could be seen in the execution of the task in both knowledge representations, participants only chose to correct this behaviour in the SBTG representation. This is due to the users noticing the Press skill while looking through the task-graph, and as this did not match how they expected the robot to perform this task, 50% of the participants chose to remove the surplus skill. In the TBTG representation on the other hand, the users did not notice the surplus skill, as they assumed that the task-graph matched what they taught, without an low effort option to verify this. Thus, by noticing this surplus skill in the skill based knowledge representation, users could correct the robot's knowledge to ensure the indented execution of the task. Furthermore, they could understand how their teaching lead to this unwanted surplus skill, and adjust

their teaching behaviour to become better teachers.
Instead of only relying on the knowledge representation to match the user' mental model, a PbD framework also needs to be able to explain the robot's knowledge to the user. It is not enough to rely on the user to work out the details, but the chosen knowledge representation needs to actively support the user in understanding the robot's knowledge. Otherwise, users can draw incorrect conclusions from the robot's knowledge. These incorrect conclusions are the major reason for the number of incorrect debug answers shown in Fig. 5.9. From the answers of the participants, it became clear that those who incorrectly assumed a program was correct, did so based on the gripper actions. For the TBTG, users assumed that the robot would always lift objects when a *close* gripper action was shown in the task-graph, even though in some cases the gripper closed without picking up an object, to allow a better performance of *contact skills*. The participants using the SBTG on the other hand could use the additional information provided by the difference between the Pick skill and the Close-gripper skill to verify the correctness of the task.
Thus, while for the SBTG only 8% of the participants gave an incorrect answer, for the time-line task graph this number was 20%. This shows that a clear communication of the details of the robot's knowledge helps the users to update their mental model of the robot, making them better teachers. That the SBTG is better at explaining the robot's knowledge becomes clear from the questions aimed at the explainability of the knowledge expectations, where the participants rated the skill based knowledge representation 58% better at explaining the robot's knowledge compared to the TBTG, and the ease of debugging of the SBTG 37% better than that of the TBTG.
Thus, by providing an intuitively understandable knowledge representation that actively helps to prevent misconceptions, the SBTG supports the users in their roles as teachers, while the TBTG allows the users to draw incorrect conclusions from the robot's knowledge and thus reduces their teaching ability. Therefore, hypothesis H3 is shown to be correct.
From the thus proven hypotheses H1-3 it can be concluded that the SBTG is more suited as a robot's knowledge representation than a TBTG, as it reduces the teacher's workload during the teaching process, builds the user's trust in the robot, and supports the user in becoming a better teacher.

Chapter 6

Conclusion

6.1 Summary

In this paper, InTAP was introduced, a framework that supports the user during PbD by providing a skill based task-graph representation of the robot's knowledge. This task-graph is extracted during the user's demonstration by an online skill segmentation algorithm. By combining data-driven skill recognition with a semantic skill recognition, the strength of both approaches are utilized. On the data-driven side, contact-based skills are recognized by their force and torque dynamics, as they have conditions and properties that are hard to describe on a symbolic level. On the semantic side, manipulation skills such as Pick, Place or Gripper Open/Close are recognized by the evaluation of predefined predicates, since their conditions can easily be described on a symbolic level. This combination allows the detection of a wider range of *contact skills*, while still allowing the algorithm to run online.
The results of a user study confirm the intuitive use of InTAP. It was shown that a skill based task-graph representation is superior to a time-line based one in reducing the user's workload when working with the knowledge representation. Additionally, the skill based task-graph lead to a higher trust of the teachers in the robot's abilities to learn new tasks correctly. Furthermore, the skill based task-graph was found to be superior in supporting the user in her role as a teacher. This shows that InTAP is highly suitable for the use in production lines with non-expert robot programmers.

6.2 Limitations

For the data-driven part of our segmentation approach to be able to classify skills regardless of which tools they are performed with, a tool weight compensation is necessary. This is due to the force- and torque readings caused by the weight of the object. Thus our approach requires the information of which tool the robot is using, in order to compensate it. This leads to an overhead, requiring the user to provide this information to the system.

Furthermore, the segmentation performance of our approach is also limited by over-segmentation, as mentioned in Sec. 3.2. This over-segmentation can occur at the beginning and end of *contact skills*, when the contact force in the first instant is high, and lower in the ongoing contact situation, or opposite. Then the algorithm can detect a Press skill before or after the actual *contact skill*. As it is impossible for the algorithm to decide if this Press skill was added intentionally, our framework currently relies on the user to resolve possible cases of over-segmentation by merging this Press skill into the actual *contact skill*.
Additionally, the current implementation of the framework only allows the addition and removal of skills at the end of the task-graph, thus limiting the reuse and correction of existing programs.

6.3 Outlook

To introduce expert knowledge into the framework, a specific controller for each skill can be designed, similar to [AAW19]. By using controllers designed to achieve the skill-specific goals, the robot's execution of the task can be improved, as the potentially sub-optimal user-demonstrations can be enhanced by skill-specific controller strategies. Furthermore, by creating additional controllers designed to transition between the skills, it becomes possible to remove or extend sections of the task-graph freely, as the controllers ensure a smooth transition for cases where the user does not provide a seamless demonstration.
To remove the overhead created by the necessity of knowing which tool the robot is using, a vision sensor could be used to automatically recognize the tool. As the robot is likely using a fixed set of tools, the user could provide the images, weights and center-of-masses of the tools once, and then rely on the vision sensor to recognize the tools during the demonstration.
To prevent incorrect skill detection to go unnoticed by the user, the framework could additionally be expanded to make it easier for the user to notice the need for corrections and to execute them. In the aforementioned case of over-segmentation, the user could automatically be queried to verify if the Press skill was intended, giving her the option to merge it in the *context skill* without additional effort. The same feedback loop could also be used for skills that are prone to be confused, such as Slide and Contour. When the segmentation algorithm predicts similarly good scores for both skills, the user could be queried to resolve the label, ensuring the correct choice.

Appendix A

User Study Questionnaire

The following document shows the questionnaire used in the user study, containing the NASA-TLX, ISO 9241-110 and the explainability questions. A questionnaire is filled out for each of the task-graph representation.

Questionnaires
Intuitive Knowledge Representation

Demographic Questionnaire

Number: _____

Age: _____

Gender: _____

Professional Activity: _____

Experience with robots: ☐ yes ☐ no

Specify experience (type, duration):

_____

Experience with Intuitive Robot Programming: ☐ yes ☐ no

Specify experience (type, duration):

_____

NASA-TLX for method _____

NASA Task Load Index

Hart and Staveland's NASA Task Load Index (TLX) method assesses work load on five 7-point scales. Increments of high, medium and low estimates for each point result in 21 gradations on the scales.

[redacted]

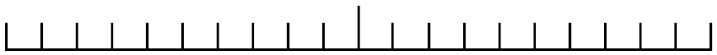
Mental Demand — How mentally demanding was the task?

Very Low — Very High

Physical Demand — How physically demanding was the task?

Very Low — Very High

Temporal Demand — How hurried or rushed was the pace of the task?

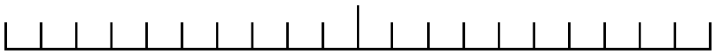


Very Low — Very High

Performance — How successful were you in accomplishing what you were asked to do?

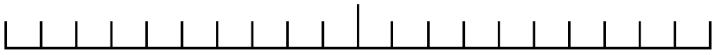
Perfect — Failure

Effort — How hard did you have to work to accomplish your level of performance?



Very Low — Very High

Frustration — How insecure, discouraged, irritated, stressed, and annoyed wereyou?



Very Low — Very High

ISO 9241-110 for method _____

	Strongly Disagree	Disagree	Some-what Disagree	Neutral	Some-what Agree	Agree	Strongly Agree
suitability for the task							
Self-descriptiveness							
conformity with user expectations							
suitability for learning							
controllability							

General Questions for method _____

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The robot understood how to perform the task it was supposed to learn.					
I would feel comfortable to let the robot execute the program unsupervised.					
I could find mistakes in the program with the help of the task graph before watching the robot execute the program.					
I knew where I needed to change the program to correct mistakes.					
I could predict how the robot would execute the task after I saw the task graph.					
My expectation of how the robot would solve this task matched the task graph.					
The task graph explained the robot's knowledge well.					
I liked the way I could interact with the task graph.					

List of Figures

Acronyms and Notations

PbD Programming by Demonstration

HMM Hidden Markov Model

GMM Gaussian Mixture Model

DMP Dynamic Movement Primitive

ZVC Zero Velocity Crossing

SVM Support Vector Machine

DTW Dynamic Time Warping

OAC Object-Action-Complex

RL Reinforcement Learning

SEC Semantic-Event-Chain

GMM Gaussian Mixture Model

SBTG Skill Based Task-Graph

TBTG Time-line Based Task-Graph

Bibliography

[AAD+11] Eren Erdal Aksoy, Alexey Abramov, Johannes Dörr, Kejun Ning, Babette Dellen, and Florentin Wörgötter. Learning the semantics of object–action relations by observation. *The International Journal of Robotics Research*, 30(10):1229–1249, 2011.

[AAW19] Mohamad Javad Aein, Eren Erdal Aksoy, and Florentin Wörgötter. Library of actions: Implementing a generic robot execution framework by using manipulation action semantics. *The International Journal of Robotics Research*, 38(8):910–934, 2019.

[AAWD10] Eren Erdal Aksoy, Alexey Abramov, Florentin Wörgötter, and Babette Dellen. Categorizing object-action relations from semantic scene graphs. In *2010 IEEE International Conference on Robotics and Automation*, pages 398–405, 2010.

[ABB21] ABB. Abb autonomous drilling robot, Last accessed on 31.03.2021. URL: `https://new.abb.com/news/detail/53854/autonomous-drilling-robot-debut`.

[ASHO+07] Alin Albu-Schäffer, Sami Haddadin, Ch Ott, Andreas Stemmer, Thomas Wimböck, and Gerhard Hirzinger. The dlr lightweight robot: design and control concepts for robots in human environments. *Industrial Robot: an international journal*, 2007.

[AZWA16] Eren Erdal Aksoy, You Zhou, Mirko Wächter, and Tamim Asfour. Enriched manipulation action semantics for robot execution of time constrained tasks. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 109–116, 2016.

[BCDS08] Aude Billard, Sylvain Calinon, Ruediger Dillmann, and Stefan Schaal. Survey: Robot programming by demonstration. Technical report, Springrer, 2008.

[BSP+04] Jernej Barbič, Alla Safonova, Jia-Yu Pan, Christos Faloutsos, Jessica K Hodgins, and Nancy S Pollard. Segmenting motion capture data into distinct behaviors. In *Proceedings of Graphics Interface 2004*, pages 185–194. Citeseer, 2004.

[BVL12] Eugen Berlin and Kristof Van Laerhoven. Detecting leisure activities with dense motif discovery. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 250–259, 2012.

[CBNKL18] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W Kempa-Liehr. Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). *Neurocomputing*, 307:72–77, 2018.

[CGB07] Sylvain Calinon, Florent Guenter, and Aude Billard. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2):286–298, 2007.

[CL18] Sylvain Calinon and Dongheui Lee. Learning control. *Journal: Humanoid Robotics: A Reference*, pages 1–52, 2018.

[CT14] Maya Cakmak and Andrea L Thomaz. Eliciting good teaching from humans for machine learners. *Artificial Intelligence*, 217:198–215, 2014.

[EGL+19] Mark Edmonds, Feng Gao, Hangxin Liu, Xu Xie, Siyuan Qi, Brandon Rothrock, Yixin Zhu, Ying Nian Wu, Hongjing Lu, and Song-Chun Zhu. A tale of two explanations: Enhancing human trust by explaining robot behavior. *Science Robotics*, 4(37), 2019.

[FMJ02] Ajo Fod, Maja J Matarić, and Odest Chadwicke Jenkins. Automated derivation of primitives for movement classification. *Autonomous robots*, 12(1):39–54, 2002.

[HB18] Thomas Hellström and Suna Bensch. Understandable robots-what, why, and how. *Paladyn, Journal of Behavioral Robotics*, 9(1):110–123, 2018.

[HKK14] Roshanak Houmanfar, Michelle Karg, and Dana Kulić. Movement analysis of rehabilitation exercises: Distance metrics for measuring patient progress. *IEEE Systems Journal*, 10(3):1014–1025, 2014.

[HSM96] Geir E Hovland, Pavan Sikka, and Brenan J McCarragher. Skill acquisition from human demonstration using a hidden markov model. In *Proceedings of IEEE international conference on robotics and automation*, volume 3, pages 2706–2711, 1996.

[INS01] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Trajectory formation for imitation with nonlinear dynamical systems. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, volume 2, pages 752–757, 2001.

[IØRS17] Iñigo Iturrate, Esben Hallundbæk Østergaard, Martin Rytter, and Thiusius Rajeeth Savarimuthu. Learning and correcting robot trajectory keypoints from a single demonstration. In *2017 3rd International Conference on Control, Automation and Robotics (ICCAR)*, pages 52–59. IEEE, 2017.

[KKGB12] George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew Barto. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*, 31(3):360–375, 2012.

[KOL+12] Dana Kulić, Christian Ott, Dongheui Lee, Junichi Ishikawa, and Yoshihiko Nakamura. Incremental learning of full body motion primitives and their sequencing through human motion observation. *The International Journal of Robotics Research*, 31(3):330–345, 2012.

[KTM10] Nathan Koenig, Leila Takayama, and Maja Matarić. Communication and knowledge sharing in human–robot interaction and learning from demonstration. *Neural Networks*, 23(8-9):1104–1112, 2010.

[LK13] Jonathan Feng-Shun Lin and Dana Kulić. Online segmentation of human motion for automated rehabilitation exercise analysis. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 22(1):168–180, 2013.

[LKK16] Jonathan Feng-Shun Lin, Michelle Karg, and Dana Kulić. Movement primitive segmentation for human motion modeling: A framework for analysis. *IEEE Transactions on Human-Machine Systems*, 46(3):325–339, 2016.

[LO11] Dongheui Lee and Christian Ott. Incremental kinesthetic teaching of motion primitives using the motion refinement tube. *Autonomous Robots*, 31(2):115–131, 2011.

[MKZ+09] Abdullah Mueen, Eamonn Keogh, Qiang Zhu, Sydney Cash, and Brandon Westover. Exact discovery of time series motifs. In *Proceedings of the 2009 SIAM international conference on data mining*, pages 473–484. SIAM, 2009.

[MTS12] Franziska Meier, Evangelos Theodorou, and Stefan Schaal. Movement segmentation and recognition for imitation learning. In *Artificial Intelligence and Statistics*, pages 761–769, 2012.

[NM01] Monica N Nicolescu and Maja J Mataric. Learning and interacting in human-robot domains. *IEEE Transactions on Systems, man, and Cybernetics-part A: Systems and Humans*, 31(5):419–430, 2001.

[NM03] Monica N Nicolescu and Maja J Mataric. Natural methods for robot task learning: Instructive demonstrations, generalization and practice. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 241–248, 2003.

[NOKB12] Scott Niekum, Sarah Osentoski, George Konidaris, and Andrew G Barto. Learning and generalization of complex tasks from unstructured demonstrations. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5239–5246. IEEE, 2012.

[PHAS09] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. In *2009 IEEE International Conference on Robotics and Automation*, pages 763–768, 2009.

[PNA+16] Mikkel Rath Pedersen, Lazaros Nalpantidis, Rasmus Skovgaard Andersen, Casper Schou, Simon Bøgh, Volker Krüger, and Ole Madsen. Robot skills for manufacturing: From concept to industrial deployment. *Robotics and Computer-Integrated Manufacturing*, 37:282–291, 2016.

[QELL20] Zeju Qiu, Thomas Eiband, Shile Li, and Dongheui Lee. Hand pose-based task learning from visual observations with semantic skill extraction. In *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 596–603, 2020.

[Rab89] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[Rob21] RobotWorx. Robotworx sanding robots, Last accessed on 31.03.2021. URL: `https://www.robots.com/articles/sanding-robots-provide-consistent-and-smooth-surfaces`.

[SBS+17] Thiusius Rajeeth Savarimuthu, Anders Glent Buch, Christian Schlette, Nils Wantia, Jürgen Roßmann, David Martínez, Guillem Alenyà, Carme Torras, Ales Ude, Bojan Nemec, et al. Teaching a robot the semantics of assembly tasks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(5):670–692, 2017.

[SC78] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.

[SH20] Aran Sena and Matthew Howard. Quantifying teaching behavior in robot learning from demonstration. *The International Journal of Robotics Research*, 39(1):54–72, 2020.

[She17] Raymond Sheh. "Why did you do that?" Explainable intelligent robots. In *AAAI Workshop-Technical Report*, pages 628–634, 2017.

[SNS19] Franz Steinmetz, Verena Nitsch, and Freek Stulp. Intuitive task-level programming by demonstration through semantic skill recognition. *IEEE Robotics and Automation Letters*, 4(4):3742–3749, 2019.

[VRL+09] Balakrishnan Varadarajan, Carol Reiley, Henry Lin, Sanjeev Khudanpur, and Gregory Hager. Data-derived models for segmentation with application to surgical assessment and training. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 426–434. Springer, 2009.

[WEL20] Christoph Willibald, Thomas Eiband, and Dongheui Lee. Collaborative programming of conditional robot tasks. In *International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[WJX+18] Yue Wang, Yanmei Jiao, Rong Xiong, Hongsheng Yu, Jiafan Zhang, and Yong Liu. Masd: A multimodal assembly skill decoding system for robot programming by demonstration. *IEEE Transactions on Automation Science and Engineering*, 15(4):1722–1734, 2018.

[WSA+13] Mirko Wächter, Sebastian Schulz, Tamim Asfour, Eren Aksoy, Florentin Wörgötter, and Rüdiger Dillmann. Action sequence reproduction based on automatic segmentation and object-action complexes. In *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 189–195, 2013.

[ZAWT17] Fatemeh Ziaeetabar, Eren Erdal Aksoy, Florentin Wörgötter, and Minija Tamosiunaite. Semantic analysis of manipulation actions using spatial relations. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4612–4619, 2017.

[ZH18] Zuyuan Zhu and Huosheng Hu. Robot learning from demonstration in robotic assembly: A survey. *Robotics*, 7(2):17, 2018.

[ZPKD05] R Zoliner, Michael Pardowitz, Steffen Knoop, and Rüdiger Dillmann. Towards cognitive robots: Building hierarchical task representations of manipulations from human demonstration. In *Proceedings of the 2005 IEEE International Conference On Robotics and Automation*, pages 1535–1540, 2005.

[ZYFA15] Konstantinos Zampogiannis, Yezhou Yang, Cornelia Fermüller, and Yiannis Aloimonos. Learning the spatial semantics of manipulation actions through preposition grounding. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 1389–1396. IEEE, 2015.

License