Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2023) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

5th International Conference on Industry 4.0 and Smart Manufacturing

Unifying Skill-Based Programming and Programming by Demonstration through Ontologies

Thomas Eiband[a,*], Florian Lay[a], Korbinian Nottensteiner[a], Dongheui Lee[a,b]

[a]*Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Münchener Str. 20, 82234 Wessling, Germany*
[b]*Autonomous Systems, Technische Universität Wien (TU Wien), Gußhausstraße 27-29/384, 1040 Vienna, Austria*

Abstract

Smart manufacturing requires easily reconfigurable robotic systems to increase the flexibility in presence of market uncertainties by reducing the set-up times for new tasks. One enabler of fast reconfigurability is given by intuitive robot programming methods. On the one hand, offline skill-based programming (OSP) allows the definition of new tasks by sequencing pre-defined, parameterizable building blocks termed as skills in a graphical user interface. On the other hand, programming by demonstration (PbD) is a well known technique that uses kinesthetic teaching for intuitive robot programming, where this work presents an approach to automatically recognize skills from the human demonstration and parameterize them using the recorded data. The approach further unifies both programming modes of OSP and PbD with the help of an ontological knowledge base and empowers the end user to choose the preferred mode for each phase of the task. In the experiments, we evaluate two scenarios with different sequences of programming modes being selected by the user to define a task. In each scenario, skills are recognized by a data-driven classifier and automatically parameterized from the recorded data. The fully defined tasks consist of both manually added and automatically recognized skills and are executed in the context of a realistic industrial assembly environment.

Keywords: intuitive programming; smart factory; skill recognition; robot skills; Programming by Demonstration; Learning from Demonstration; user interface; kinesthetic teaching; ontology; knowledge base;

1. Introduction

Robotic production sites should be reconfigurable as fast as possible towards manufacturing of new products to deal with uncertain market conditions or problems in the supply chain. A key enabler of fast changeovers are robotic systems that are intuitively programmable by the end user without the need for robot experts or time-consuming setup procedures. On the one hand, offline programming frameworks allow to program new tasks without blocking a

[*] Corresponding author.
E-mail address: thomas.eiband@dlr.de

production system that is currently in operation. An intuitive programming workflow can be achieved by arranging so-called skills as building blocks for a robot program in a Graphical User Interface (GUI), which we term as offline, skill-based programming (OSP), with an example provided by [32]. On the other hand, programming by demonstration (PbD) frameworks enable a fast and intuitive way of programming when the robotic system is directly accessible, often relying on kinesthetic teaching [8, 31, 9].

Ideally, future robotic systems will have a large foundation of knowledge and will be able to solve major parts of new problems autonomously by task and motion planning, which aims to find an appropriate skill sequence and skill parameters as well as collision free paths in the workspace. Although a large degree of autonomy is desired to overcome manual programming efforts, it is rarely achieved due to uncertainties and unforeseen edge-cases that arise in new tasks [27]. Therefore, existing knowledge about the concepts in the world should be efficiently combined with information provided by the end user. This work presents an approach that unifies the concepts of PbD and OSP with the help of an ontological knowledge base, such that the end user only needs to take care of missing task knowledge and the system exploits existing knowledge wherever possible. For instance, an end user could use OSP for well known, easy to describe parts of a tasks and PbD for parts that are more intuitive to be demonstrated and difficult to model.

OSP allows the user to manually add skills in the task definition process. In this work, this is achieved via the so-called Human Factory Interface (HFI) [29, 4], which is a GUI that visualizes an editor for task definition. Such editor offers a library of robot skills, that each can be manually parameterized according to the task needs. The PbD pipeline presented in this work lets the end user intuitively set up new robotic tasks by demonstration, using skill recognition, automatic skill parameterization and task execution as a sequence of parameterized skills. The automatic skill parameterization process extracts parameter values from data instead of manually defining them. This has the advantage that the user achieves two goals at once with a single demonstration, which are 1) defining the skill type; and 2) automatically parameterizing it.

Furthermore, this framework is connected to an ontological knowledge base that represents conceptual knowledge of the system as well as geometric and semantic information about the world. The underlying Factory of the Future (FoF) ontology was developed for the automation and manufacturing domain [29] and uses the IEEE 1872-2015 standard [15] at its core.

The main contributions of this work are: 1) an intuitive programming paradigm by unifying PbD with OSP; 2) an automated skill parameterization approach by combining demonstrated with semantic knowledge represented in the ontology; 3) integration of skill recognition, automatic parameterization, and execution in one framework. The document is outlined as follows. Section 2 discusses related work, Sec. 3 introduces the framework that unifies PbD and OSP, Sec. 4 describes the experimental evaluation, and Sec. 6 concludes this work and points out future research avenues.

2. Related Work

In the following, related works are discussed that introduce the concept of skills in the robotic manufacturing domain, how they are identified, and how they are either manually or automatically selected to perform a task. Finally, works are examined that present automatic skill parameterization by means of PbD.

The concept of robot skills has emerged already long time back, such as in [13]. From the cognitive perspective, skills can be described as actions that are executed based on a mental model of the performing agent [34]. More specifically, the skill embeds knowledge such that it can be applied in different situations with varying parameters that adapt its behavior. A major goal of skills is to reduce the implementation effort, increase autonomy of the system and increase reusability of the implemented behaviors in different situations. Skills are already widely applied in the research domain of service robotics [22] and become also a well known concept in the industrial manufacturing domain [30, 26], with recent approaches surveyed in [21, 12]. A clear conceptual model about an industrial robot skill is provided in [23], which was validated in a realistic industrial environment on different hardware platforms. Accordingly, hardware independent skills are a key enabler to deploy the developed skills on various hardware systems [3], for instance based on the Open Platform Communication Unified Architecture (OPC-UA) [26].

Robot skills shall also be able to represent the individual actions that the system will perform and with which parameters. To make a skill understandable to the end user, meaningful skills must be identified, which are able to solve a variety of tasks in a specific domain [7]. This is also needed when mapping production skills to a given process and available resources [25]. For instance, an assembly robot needs to have skills to manipulate objects, such as pick and place, and to mate components, such as insertion strategies or inserting and driving screws.

Considering PbD as programming mode, so-called skill recognition can be used, which infers an appropriate skill type given a human demonstration [7, 8]. The motivation of skill recognition is that the human has rather an understanding about a declarative description of "what" to achieve with the task. Whereas the robot requires an imperative representation of "how" to solve it [2]. In consequence, the human demonstrates "what" to do and the skill knows "how" to do it using its internal model that is parameterized from the demonstration. Supervised skill recognition approaches use methods such as semantic knowledge models [1, 31], data-driven classification [5, 7], or a combination of these [8]. Unsupervised approaches operate without a predefined set of skills [18], however, are not able to express the meaning of a recognized action, due to the missing action annotation.

Once a skill is recognized from a human demonstration, it can be parameterized, where the parameters can be extracted from the demonstration data such that the skill becomes ready for execution. In comparison, OSP using a graphical user interface requires the manual parameterization of skills [32, 14]. Examples of automatic parameter extraction can be found for hybrid motion and force control in force-based skills [20], for task-parameterization techniques by using a demonstrated target frame as task parameter [6] or for visually extracted task parameters [24, 33], for task parameters extracted from trajectory data [17], and for semantically extracted skills [31].

The proposed framework in this work builds on our previously developed approaches. First, a user study extracted a library of legible skills [7]. By using this library, skill recognition approaches either detect individual [7] or sequences of skills [8]. The FoF ontology [29] allows to combine the available semantic and demonstrated knowledge in a unified programming framework. Finally, automatic skill parameterization extracts meaningful parameters for the recognized skill from the human demonstration data [20]. The proposed framework integrates the above mentioned approaches to form an automated process that connects the human demonstration phase, skill recognition, skill parameterization, and complete task execution.

3. Unified Programming Framework

The unified programming framework allows to define a new task either by PbD(blue enclosed area) with the demonstration data $\boldsymbol{X}$ or graphically by OSP (green enclosed area) with the manually selected actions $\boldsymbol{A}$ (Fig. 1). The

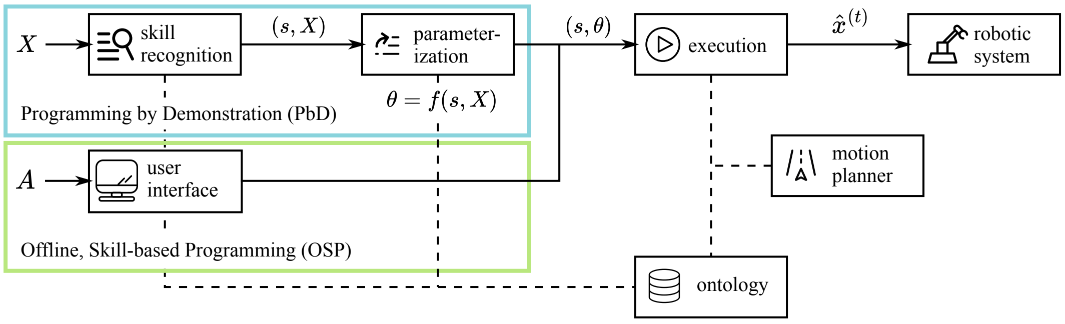


Fig. 1: Unified programming framework. Solid lines show the process flow, dashed lines show the data exchange with the ontological knowledge base (ontology).

execution stage connects to the real robotic system and executes the actual skills. A skill instance, which is created from a fully parameterized skill, commands the robot via the associated robot hardware interface, manipulates the physical world, and updates the representations of the objects in the knowledge base. A motion planner provides collision free paths in the environment of the workcell and is interfaced with both the execution stage and a central knowledge base, represented as ontology. The knowledge base serves geometric data about the environment, abstract data about objects, systems and their atomic functions, and conceptual data, such as the relationship between knowledge entities and their properties.

Due to the combination with an ontology, the skill recognition can make use of a set of known skills that can be recognized according to the current context. The connection to the ontology also simplifies the skill parameterization because knowledge about the environment exists apart from the demonstration data. Hence, the skill execution can be improved in terms of adaptability to changes in the world, for instance the pick pose of an object could be selected from multiple options that are stored as alternatives in the knowledge base.

3.1. *Skill Recognition and Parameterization*

The blue enclosed area in Fig. 1 refers to the programming mode of PbD and displays the system modules in the automated programming pipeline. The skill recognition methodology relies on a previous work [7] that mainly considers skills with physical contact, such as *touch*, *press*, or *insert*. It was later extended to manipulation tasks such as *pick*, *place*, or *move* [8]. The skill recognition stage accepts a human demonstration time-series $\boldsymbol{X}$ and outputs the tuple $(s, \boldsymbol{X})$ with the predicted skill s, which is displayed to the user. Next, the parameterization stage automatically derives a parameter set $\boldsymbol{\theta}$ by means of the mapping function $\boldsymbol{\theta} = f(\boldsymbol{X})$, providing the tuple $(s, \boldsymbol{\theta})$ to the execution stage.

3.2. *Offline, Skill-based Programming*

The green enclosed area in Fig. 1 refers to the programming mode of OSP and uses the HFI as programming interface. The user actions are denoted as $\boldsymbol{A}$. The first user action is to pick a predefined skill from a skill library. The second user action is to define the parameter values of the skill. Here, the HFI guides the user through the parameterization process where only parameter values that are valid in the context are displayed. For example, the user can only select objects that are available in the current workcell and that are not already bound as a resource elsewhere. Analogous to PbD, the OSP stage output is a tuple $(s, \boldsymbol{\theta})$ that is fed to the execution stage.

3.3. *Task Execution*

The execution stage runs a skill and configures its underlying primitives during the runtime. A skill is assembled by so-called primitives, noted as < ... >, which are reusable building blocks for atomic functions of the robot that are modeled in the FoF ontology. Examples for primitives are < planned_motion >, < open_gripper >, or < cartesian_path >. This work only uses sequential arrangements of primitives within a skill, while other structures such as decision trees or state machines are possible. During the execution phase of a skill, each of its primitives is configured given the previously extracted skill parameters $\boldsymbol{\theta}$. The mapping function from $\boldsymbol{\theta}$ to a parameter $\boldsymbol{\theta}_p$ of primitive p is given as $\boldsymbol{\theta}_p = f_p(\boldsymbol{\theta})$ for each primitive $p \in [1..P]$. A simple example for a mapping function is about parameterizing a < *planned_motion* > primitive serving as approach path of an insertion skill. The demonstrated path until the contact on the target object can be inferred by thresholding the force and torque values to find the first contact point. The end-point of the approach path can then be stored relative to the object and used to plan future motions towards new target object locations.

While skill parameters are task specific and fixed for the same task goals, the primitive parameters are context specific and may vary according to the location of each object. Finally, each skill is executed by orchestrating its underlying primitives and the controllers that they implement, resulting in the control input $\hat{\boldsymbol{x}}^{(t)}$ at each time step t.

3.4. *Ontological Knowledge Base*

Recent development in robotic domain ontologies represent the abilities and skills of robotic agents in manufacturing situations, as well as concepts related to autonomy. In assembly processes, ontologies have been proposed to represent task-specific knowledge that describes geometric features and part connections, which aid in solving assembly planning problems. The IEEE 1872-2015 standard outlines a core ontology that specifies the most general concepts, relations, and axioms of robotics and automation [15, 11]. The FoF ontology introduced in [29] extended the standardized concepts to allow the modeling of object manipulation properties, such as pick poses, approach and departure paths, and entities such as `Parameter`, `Skill`, `Task`, `Place`, and `StorageDevice`. Each entity is addressed by a unique identifier, called Internationalized Resource Identifier (IRI), with the ultimate aim to make this information publicly accessible and allow to reuse conceptual knowledge in the robotics domain. Figure 2 shows the composition of ontologies used in this work.

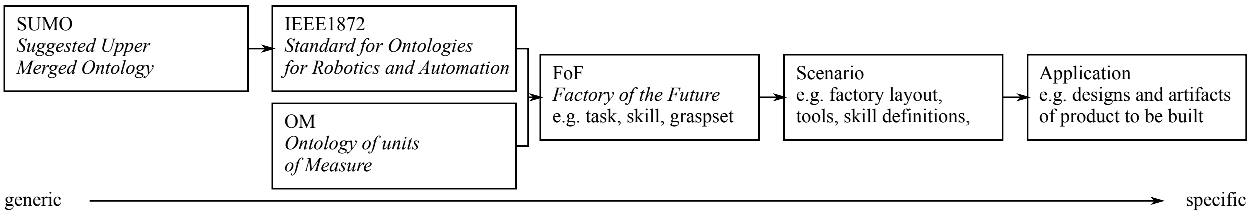

SUMO
Suggested Upper Merged Ontology
IEEE1872
Standard for Ontologies for Robotics and Automation
OM
Ontology of units of Measure
FoF
Factory of the Future
e.g. task, skill, graspset
Scenario
e.g. factory layout, tools, skill definitions,
Application
e.g. designs and artifacts of product to be built
generic
specific

Fig. 2: Composition of ontologies (SUMO [19], IEEE1872 [15], OM [28], FoF [29]), which are building on top of each other from left to right. Image adapted from [29].

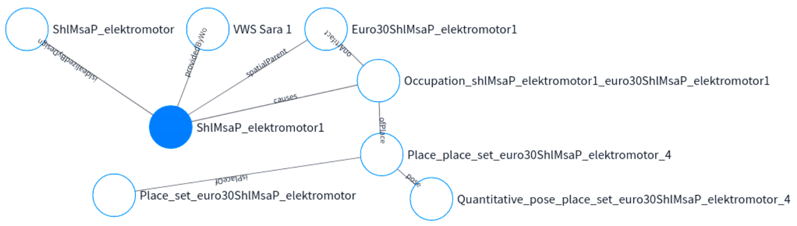
All physical objects, which are termed `Artifact`, have transformations assigned with respect to a spatial parent. For instance, an object that is located in a storage has a transformation associated between the object frame and each place frame that is part of the storage. A `Design` describes the type of an object and exists only once. It has a so-called `GraspSet`, which defines at least one relative grasp pose as well as an approach and departure path. This allows to define this information only once and share it among all `Artifacts` of this `Design`. With that concept, transformation chains between the robot and the grasp pose of any object can be computed. A similar concept applies for object place poses, which is described in the complete ontology standard [29].

Starting from the perspective of an object, all objects of the same `Design` that reside in the same `StorageDevice` can seamlessly be queried. This is later exploited in Sec. 4.1 to find a pattern in the arrangement of objects. Another example is the placement of objects on a predefined `Place`, which again can make use of previously modeled approach paths and create a so-called `Occupation` on the respective `Place`. Since the relative transformation between `Design` and `Place` is known, autonomous placement of objects becomes possible. Furthermore, previously learned behaviors from PbD can be easily reused by transforming them to new target frames of a new `Place`, as exploited in Sec. 4.2. Fig. 3a shows an excerpt of the modeled knowledge and Fig. 3b shows the geometric world representation of a variable workstation.

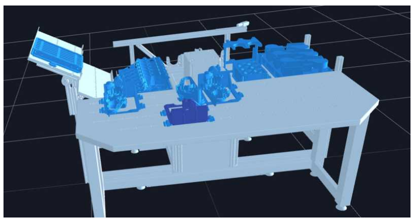
4. Experiments

All experimental scenarios were implemented and tested with a DLR SARA lightweight robot [16] on a variable workstation of the flexible production network in our lab setup, as similarly used in [29, 4]. A video shows the programming workflow for adding the skills in both scenarios[1].

[1] https://doi.org/10.5281/zenodo.8383257


ShIMsaP_elektromotor
VWS Sara 1
Euro30ShIMsaP_elektromotor1
Occupation_shIMsaP_elektromotor1_euro30ShIMsaP_elektromotor1
ShIMsaP_elektromotor1
Place_place_set_euro30ShIMsaP_elektromotor_4
Place_set_euro30ShIMsaP_elektromotor
Quantitative_pose_place_set_euro30ShIMsaP_elektromotor_4
idealizedByDesign
providedByWo
spatialParent
causes
ofPlace
pose

(a)



(b)

Fig. 3: Graphical representations of ontological knowledge. (a) shows entities (circles) and their properties (lines) that are modeled in the ontology. The example shows a physical object marked as blue dot (`Artifact`) of a specific type (→*idealizedByDesign* →`Design`) and how it is geometrically (→*spatialParent* →`Storage`) and semantically (→*causes* →`Occupation` →*ofPlace* →`Place`) referenced in the ontology. (b) Visualization of the variable workstation.

4.1. Object Identification by Touch

In a shared workspace, objects can either be manipulated by human or robot. In consequence, both agents alter the physical world state. The robot uses skills to update both the physical state and the modeled world state that is represented in the knowledge base. Instead, the user might change only the physical state without changing the modeled state since it is very likely that a system cannot track all induced changes.

Reasons for a miss-match between the physical state and modeled state can be 1) a shared assembly processes where parts are either manipulated by user or robot; 2) the removal of parts by the user, e.g. due to quality checks or sorting out defective parts; or 3) an incompletely filled storage arrangements (e.g. as shown in Fig. 4c). In this use-case, a *touch* skill shall be employed, which is able to haptically explore the environment until it finds an object. Here, it is used to identify the next object in an object storage that is potentially incomplete or its content does not match with the modeled state due to aforementioned reasons.

Implementation. The implementation follows the idea presented in [10] to identify a linear exploration path that is used to detect object locations in the workspace. In [10], the skill extracted all necessary information by multiple user demonstrations. The main difference is now that the skill also exploits the information in the ontology to infer the exploration path itself. This enables the system to fully parameterize the skill from a single demonstration, which reduces teaching effort, allows collision free movements due to the usage of a world model, and improves the inference about the exploration path because possible object locations are already modeled in the ontology.

In detail, the internal skill parameterization constructs multiple primitives from the initial demonstration (Fig. 4b). The first primitive (< cartesian_path >) approaches the exploration region. The second primitive (< cartesian_move >) explores the next object by using a linear motion. This motion is defined by querying all possible storage places in a storage container. It then computes a singular value decomposition (SVD) on the storage places to identify the best-fitting line. Note that the eigenvector shown in Fig. 4b might be anti-parallel to the desired exploration direction depending on the outcome of the SVD. This line is translated to intersect the contact point from the demonstration and becomes the actual exploration path. A visual representation of the parameterization can be found in the experimental results (Fig. 4b).

The skill execution sequences the previously constructed primitives and transitions from one to the next primitive by predefined conditions. The most important condition is the expected contact during execution of the exploration path. Once the contact is detected, a transition is made to the next primitive that is responsible for the departure.

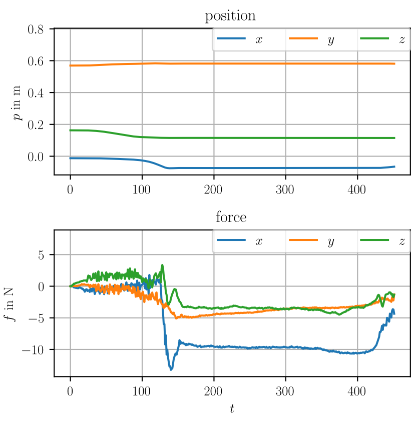
Experiment. The procedure is as follows:

1. User creates a new empty task via the HFI (the whole task is shown in Fig. 4d).
2. User manually adds an *equip_device* skill with parameter: `tool: robotiq-140`, attaching it as tool.
3. User provides demonstration of touching an object in the workspace (Fig. 4a and 4c).
4. Robot automatically recognizes a *touch* skill and adds it to the skill sequence
5. User manually adds a *pick* skill with parameter: `object: housing`, defining the type of object to be picked.
6. User starts the fully defined task.
7. Robot executes task with identification of new object location, and picks the next object of same type (Fig. 6).

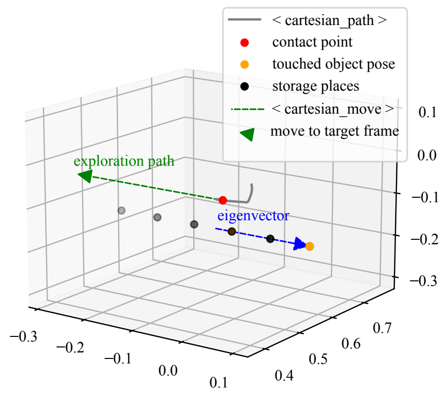
This experiment shows that the user can demonstrate a part of the task which is recognized as *touch* skill and continue the task definition process by adding the *pick* skill manually. The demonstration time for the *touch* skill was 4.52 s and the GUI programming time of the *pick* skill was approximately 7 s, requiring 5 clicks. Since the *pick* skill is already implemented and can be executed autonomously, it only has to be parameterized without requiring a user demonstration. Hence, only parts of the task are demonstrated where the knowledge base has either missing information or the user simply favors the programming mode of PbD over OSP.

4.2. Object Insertion

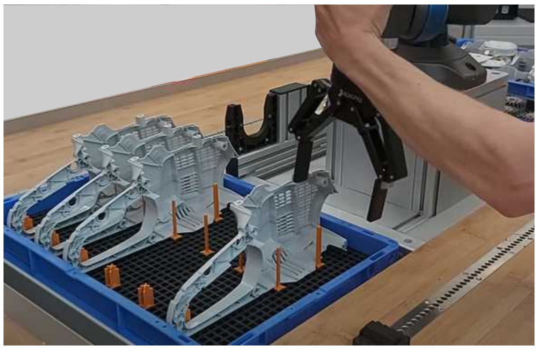
Implementation. The insert skill makes use of the knowledge represented in the ontology to infer if it requires a user demonstration to solve the task. Due to the access to the knowledge base, it is also able to query the target coordinate



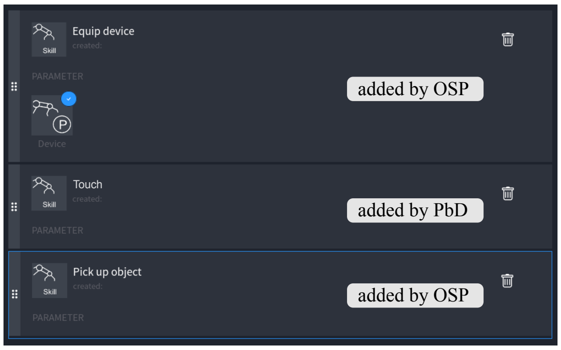
(a) Demonstration data.



(b) Derived parameters and primitives denoted as < ... >.



(c) Demonstrating how to touch an object in a storage.



(d) Task Representation as shown to the user in the HFI

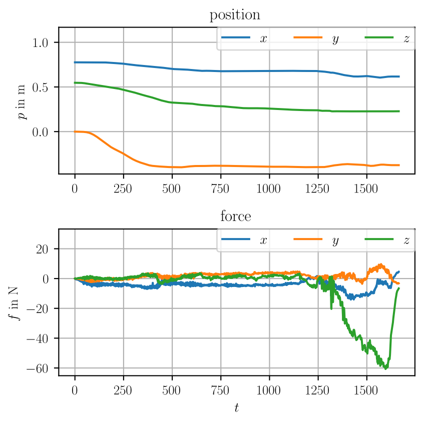
Fig. 4: Demonstration of touching an object in a storage.

frame of the insertion, hence being able to transform the learned strategy and reproduce it in new target coordinate frames.

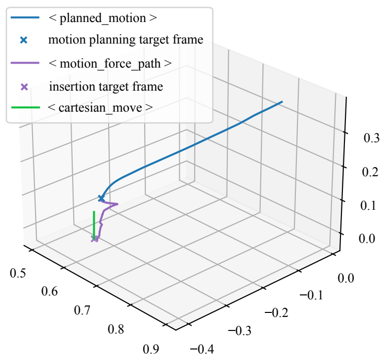
The skill parameterization constructs four primitives out of the demonstration (Fig. 5b). The first primitive (< planned_motion >) is constructed by extracting the first physical contact point of the demonstration until the object touches the environment, which is used as the motion planning target frame. The second primitive (< motion_force_path >) is shaped by the demonstration and is in charge to reproduce the demonstrated trajectory and wrench using an impedance controller with wrench overlay and predefined stiffness values. The third primitive (< open_gripper >) control the gripper after the insertion took place. The fourth primitive (< cartesian_move >) departs linearly from the insertion location. The skill execution sequences the constructed primitives, while their transition conditions are simply the termination of each primitive.

Experiment. The procedure is as follows:

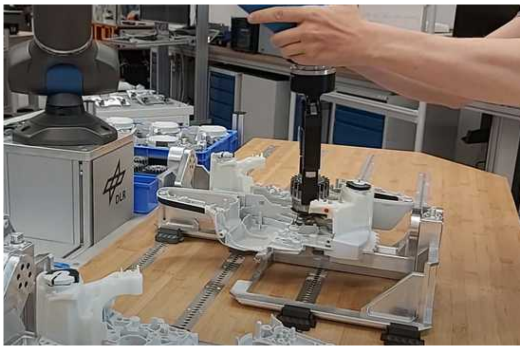
1. User creates a new empty task via the HFI (the whole task is shown in Fig. 5d).
2. User manually adds an *equip_device* skill with parameter: `tool: robotiq-140`, attaching it as tool.
3. User manually adds a *pick* skill with parameter: `object: motor`, defining the type of object to be picked.
4. User provides demonstration of inserting a motor into the housing (Fig. 5a and 5c).
5. Robot automatically recognizes an *insert* skill and adds it to the skill sequence.
6. User starts the fully defined task.
7. Robot executes *pick* skill and picks motor.
8. Robot executes *insert* skill and inserts motor into housing.
9. Robot executes task of inserting the motor into a new target housing by reusing the demonstrated strategy (Fig. 7).



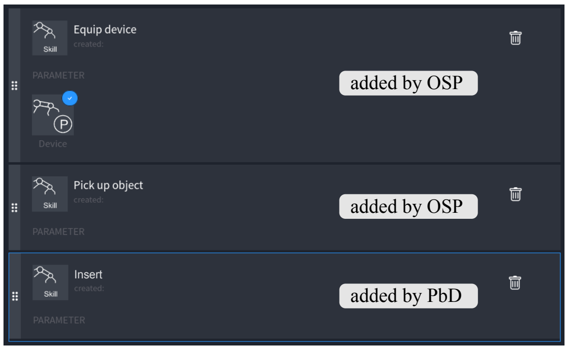
(a) Demonstration data.



(b) Derived parameters and primitives.



(c) Demonstrating how to insert the motor into the housing.



(d) Task Representation as shown to the user in the HFI

Fig. 5: Demonstration of inserting an object into a target object.

Similarly as in the previous experiment, the user can either demonstrate parts of the task which are recognized as skills or define parts of the task manually by adding skills to the HFI. The GUI programming time of the *pick* skill was approximately 7 *s*, requiring 5 clicks. The demonstration time for the *insert* skill was 16.65 *s*. Again, only parts of the task are demonstrated where the knowledge base has either missing information or the user chooses the programming mode of PbD instead of OSP due to personal preferences.

5. Discussion

The proposed framework aims to simplify robot programming by providing multiple intuitive input channels. The experimental results are discussed with respect to the following factors. 1) Selection of an input method that speeds up the programming phase; 2) Suitability to the user's strengths and experiences; 3) Feasibility to define the part of the task; and 4) Reusability of existing knowledge in the system.

Regarding (1), the user could freely choose an input method in case the interface allowed to fully define the skill. Using PbD is expected to be more efficient than OSP if the parameters are complex to describe, such as the motion and force trajectory that defines the strategy of the *insert* skill. For instance, a *pick* skill can be parameterized both via OSP or PbD. However, in the experiments, the OSP interface for the *touch* skill using the HFI did not allow to manually define an exploration path. This design choice was made because it is assumed that the exploration path can be more efficiently extracted from a single human demonstration. Regarding (2), each user might have different strengths and experiences. While one user might feel comfortable with hand-guiding the robot, the other would prefer to exclusively use a graphical user interface. Evaluating the programming times (1) and the user's mental demand (2) would require a user study to obtain quantitative results. Point (3), the feasibility of defining the part of the task is mainly governed by technical constraints. Some skills cannot be demonstrated by kinesthetic teaching at all, which is

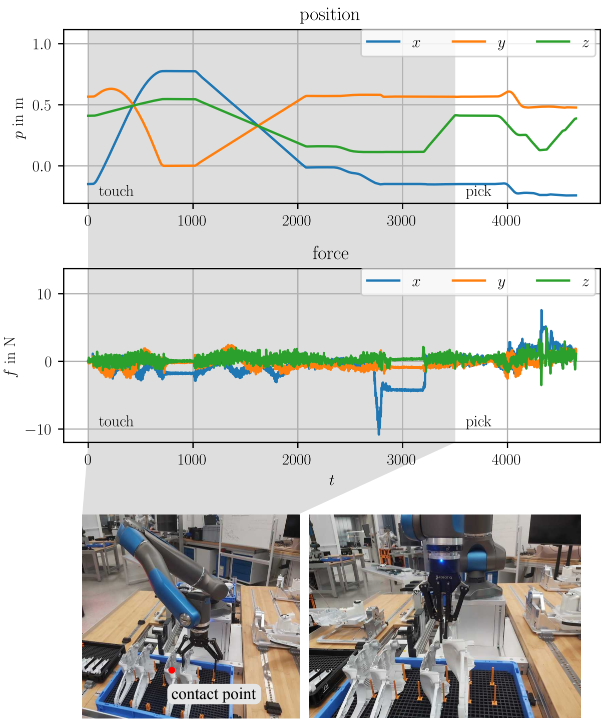


Fig. 6: Sequenced execution of *touch* and *pick*.

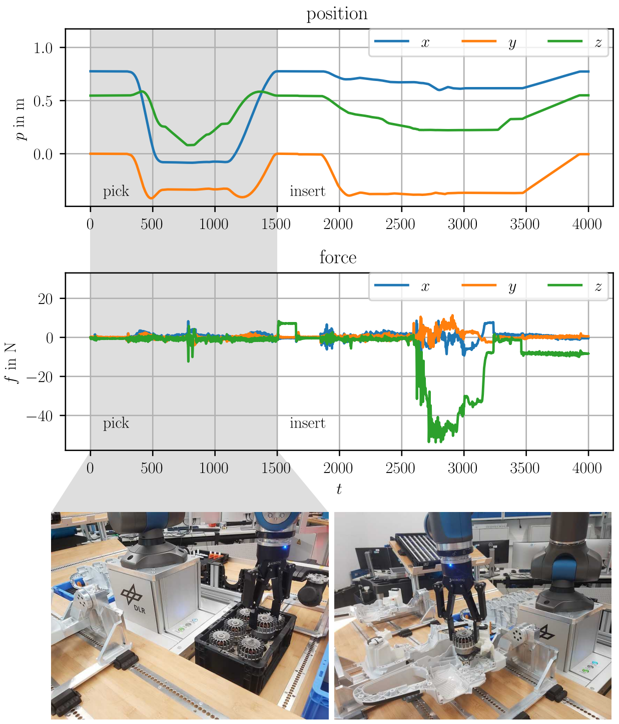


Fig. 7: Sequenced execution of *pick* and *insert*.

in the above experiments the *equip_device* skill responsible for the tool change. It requires a rotation of the last joint while the robot flange is inside the tool holder. However, the flange is not accessible by the human hand during this operation. Although parts of the task can only be programmed with one of the programming methods, the system can reuse knowledge about existing skills concerning point (4). For instance, after demonstrating the insertion skill for the motor, it can be reused anywhere else in the workspace by transforming the insertion path into a new frame. Similarly, as the tool change strategy is already part of the system's knowledge, it is used for any tool change in the robotic workcell by parameterizing it with the name of the desired tool to be attached.

6. Conclusion

The integration of the full pipeline from user demonstration, skill recognition, skill parameterization, and skill execution enable the following advantages. 1) The user does not need to know or search in the whole set of skills, but simply demonstrates the desired behavior. 2) A recognized skill can be automatically parameterized from the same demonstration data without additional manual effort compared to OSP. 3) The programming modes of PbD and OSP can be combined empowering the end user to choose a suitable programming method. 4) The skill recognition approach can be integrated into sophisticated frameworks that have access to semantic knowledge and thus simplify the parameterization procedure.

Skill recognition and parameterization can be enhanced with ontological knowledge, which in turn reduces additional demonstration effort, as used in approaches that exploit multiple demonstrations to cover distributions about object locations as used in [10]. This is validated in the first experiment that considers a task that sequences a touch skill and a pick skill. First, a touch skill is recognized from demonstration, whose parameters for its exploration path are automatically extracted both from the demonstration and from the connected ontology. The knowledge that is embedded in its implemented behavior is exploited during execution by halting the robot at the explored contact point and by updating the inferred object position. Second, a pick skill is added manually via the user interface, which depends on the knowledge gained by the prior touch skill.

Further, learning from demonstration can be a source of skill parameterization techniques that yield optimized, adaptable robot behaviors. The second experiment highlights this benefit by considering a task that sequences a pick skill and an insertion skill. First, a pick skill is manually added to the task via the user interface. Second, a demonstration leads to the recognition of an insert skill, which is parameterized automatically from the demonstration, using a planned motion phase to approach the insertion and an impedance controlled, force-superimposed insertion phase during execution." The two different programming modes fundamentally differ in their interaction behavior between human and system, where PbD relies on haptic feedback while OSP relies on visual feedback. This empowers end users to choose from different interaction modes that best match with their personal skills.

Limitations of the presented framework are mainly introduced by each of the programming modes independently. With PbD, kinesthetic teaching might interfere with the natural strategy a human demonstrator tries to perform due to the robot's inertia. With OSP, the required knowledge about objects and environment has to be modeled by the system designer. For instance, when programming a pick skill via the GUI, the grasp pose has to be already known to the system. Instead, the grasp pose can be easily shown by a demonstration. The framework does not recommend a specific programming mode to the user, which might lead to the selection of an unsuitable programming mode by non-experts.

The main factors that determine the selection of a programming mode, i.e., if a skill is demonstrated or manually added, are seen in 1) the level of detail in modeling the robotic system and its surroundings; 2) the user-specific preference and competence over the teaching modality; and 3) the task-specific requirements that allow each teaching modality. A future research direction could be analyzing which programming mode is suitable for which kind of problem, followed by designing a decision support system for end users. This analysis should involve a user study to examine the abovementioned factors and define clear performance metrics that allow a better quantitative comparison with existing works.

References

[1] Ahmadzadeh, S.R., Paikan, A., Mastrogiovanni, F., Natale, L., Kormushev, P., Caldwell, D.G., 2015. Learning symbolic representations of actions from human demonstrations, in: IEEE International Conference on Robotics and Automation (ICRA), IEEE. pp. 3801–3808.

[2] Albu-Schäffer, A., Huchler, N., Kessler, I., Lay, F., Perzylo, A., Seidler, M., Steinmetz, F., Weitschat, R., 2023. Soziotechnisches assistenzsystem zur lernförderlichen arbeitsgestaltung in der robotergestützten montage. Gruppe. Interaktion. Organisation. Zeitschrift für Angewandte Organisationspsychologie (GIO) 54, 79–93.

[3] Andersen, R.H., Solund, T., Hallam, J., 2014. Definition and Initial Case-Based Evaluation of Hardware-Independent Robot Skills for Industrial Robotic Co-Workers, in: ISR/Robotik 2014; 41st International Symposium on Robotics, pp. 1–7.

[4] Bachmann, T., Eiberger, O., Eiband, T., Lay, F., Angsuratanawech, P., Rodriguez, I., Lehner, P., Stulp, F., Nottensteiner, K., 2023. Task-specific reconfiguration of variable workstations using automated planning of workcell layouts, in: 56th International Symposium on Robotics, p. accepted.

[5] Berg, J., Reckordt, T., Richter, C., Reinhart, G., 2018. Action recognition in assembly for human-robot-cooperation using hidden markov models. Procedia CIRP 76, 205–210.

[6] Calinon, S., 2016. A tutorial on task-parameterized movement learning and retrieval. Intelligent Service Robotics 9, 1–29.

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

[8] Eiband, T., Liebl, J., Willibald, C., Lee, D., 2023. Online task segmentation by merging symbolic and data-driven skill recognition during kinesthetic teaching. Robotics and Autonomous Systems 162, 104367.

[9] Eiband, T., Saveriano, M., Lee, D., 2019a. Intuitive programming of conditional tasks by demonstration of multiple solutions. IEEE Robotics and Automation Letters 4, 4483–4490.

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

[11] Fiorini, S.R., Chibani, A., Haidegger, T., Carbonera, J.L., Schlenoff, C., Malec, J., Prestes, E., Gonçalves, P., Ragavan, S.V., Li, H., 2019. Standard Ontologies and HRI, in: Human–Robot Interaction, Chapman and Hall/CRC. pp. 19–47.

[12] Froschauer, R., Köcher, A., Meixner, K., Schmitt, S., Spitzer, F., 2022. Capabilities and skills in manufacturing: A survey over the last decade of etfa. arXiv preprint arXiv:2204.12908 .

[13] Hasegawa, T., Suehiro, T., Takase, K., 1992. A model-based manipulation system with skill-based execution. IEEE Transactions on Robotics and Automation 8, 535–544.

[14] Heimann, O., Guhl, J., 2020. Industrial Robot Programming Methods: A Scoping Review, in: 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 696–703. doi:10.1109/ETFA46521.2020.9211997. iSSN: 1946-0759.

[15] IEEE Robtics and Automation Society, 2015. IEEE standard ontologies for robotics and automation. IEEE Stan. 1872, 1–60.

[16] Iskandar, M., Ott, C., Eiberger, O., Keppler, M., Albu-Schäffer, A., Dietrich, A., 2020. Joint-Level Control of the DLR Lightweight Robot SARA, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 8903–8910. doi:10.1109/IROS45743.2020.9340700. iSSN: 2153-0866.

[17] Matsubara, T., Hyon, S.H., Morimoto, J., 2011. Learning parametric dynamic movement primitives from multiple demonstrations. Neural Networks 24, 493–500.

[18] Niekum, S., Osentoski, S., Konidaris, G., Chitta, S., Marthi, B., Barto, A.G., 2015. Learning grounded finite-state representations from unstructured demonstrations. The International Journal of Robotics Research 34, 131–157.

[19] Niles, I., Pease, A., 2001. Towards a standard upper ontology, in: Proceedings of the international conference on Formal Ontology in Information Systems - Volume 2001, Association for Computing Machinery, New York, NY, USA. pp. 2–9. URL: https://dl.acm.org/doi/10.1145/505168.505170, doi:10.1145/505168.505170.

[20] Origanti, V.K., Eiband, T., Lee, D., 2022. Automatic parameterization of motion and force controlled robot skills, in: Kim, J., Englot, B., Park, H.W., Choi, H.L., Myung, H., Kim, J., Kim, J.H. (Eds.), Robot Intelligence Technology and Applications 6, Springer International Publishing, Cham. pp. 66–78.

[21] Pantano, M., Eiband, T., Lee, D., 2022. Capability-based frameworks for industrial robot skills: a survey, in: IEEE International Conference on Automation Science and Engineering (CASE), p. accepted.

[22] Paulius, D., Sun, Y., 2019. A survey of knowledge representation in service robotics. Robotics and Autonomous Systems 118, 13–30.

[23] Pedersen, M.R., Nalpantidis, L., Andersen, R.S., Schou, C., Bøgh, S., Krüger, V., Madsen, O., 2016. Robot skills for manufacturing: From concept to industrial deployment. Robotics and Computer-Integrated Manufacturing 37, 282–291.

[24] Pervez, A., Lee, D., 2018. Learning task-parameterized dynamic movement primitives using mixture of gmms. Intelligent Service Robotics 11, 61–78.

[25] Pfrommer, J., Schleipen, M., Beyerer, J., 2013. PPRS: Production skills and their relation to product, process, and resource, in: 2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA), pp. 1–4. doi:10.1109/ETFA.2013.6648114. iSSN: 1946-0759.

[26] Profanter, S., Breitkreuz, A., Rickert, M., Knoll, A., 2019. A hardware-agnostic opc ua skill model for robot manipulators and tools, in: 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), IEEE. pp. 1061–1068.

[27] Righetti, L., Kalakrishnan, M., Pastor, P., Binney, J., Kelly, J., Voorhies, R.C., Sukhatme, G.S., Schaal, S., 2014. An autonomous manipulation system based on force control and optimization. Autonomous Robots 36, 11–30.

[28] Rijgersberg, H., van Assem, M., Top, J., 2013. Ontology of units of measure and related concepts. Semantic Web 4, 3–13.

[29] Schäfer, P.M., Steinmetz, F., Schneyer, S., Bachmann, T., Eiband, T., Lay, F.S., Padalkar, A., Sürig, C., Stulp, F., Nottensteiner, K., 2021. Flexible robotic assembly based on ontological representation of tasks, skills, and resources, in: Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning, pp. 702–706.

[30] Schou, C., Andersen, R.S., Chrysostomou, D., Bøgh, S., Madsen, O., 2018. Skill-based instruction of collaborative robots in industrial settings. Robotics and Computer-Integrated Manufacturing 53, 72–80.

[31] Steinmetz, F., Nitsch, V., Stulp, F., 2019. Intuitive task-level programming by demonstration through semantic skill recognition. IEEE Robotics and Automation Letters 4, 3742–3749.

[32] Steinmetz, F., Wollschläger, A., Weitschat, R., 2018. Razer—a hri for visual task-level programming and intuitive skill parameterization. IEEE Robotics and Automation Letters 3, 1362–1369.

[33] Stulp, F., Raiola, G., Hoarau, A., Ivaldi, S., Sigaud, O., 2013. Learning compact parameterized skills with a single regression, in: 2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids), pp. 417–422. doi:10.1109/HUMANOIDS.2013.7030008. iSSN: 2164-0580.

[34] Zech, P., Renaudo, E., Haller, S., Zhang, X., Piater, J., 2019. Action representations in robotics: A taxonomy and systematic classification. The International Journal of Robotics Research 38, 518–562.