Knowledge-based Programming by Demonstration using semantic action models for industrial assembly

Junsheng Ding*, Haifan Zhang*, Weihang Li*, Liangwei Zhou*, Alexander Perzylo*

***Abstract*— In this paper, we introduce a knowledge-based Programming by Demonstration (kb-PbD) paradigm to facilitate robot programming in small and medium-sized enterprises (SMEs). PbD in production scenarios requires the recognition of product-specific actions but faces challenges in the lack of suitable and comprehensive datasets, due to the large variety of involved hand actions across different production scenarios. To address this issue, we utilize standardized grasp types as the fundamental feature to recognize basic hand movements, where a Long Short-Term Memory (LSTM) network is employed to recognize grasp types from hand landmarks. The product-specific actions, aggregated from the basic hand movements, are formally modeled in a semantic description language based on the Web Ontology Language (OWL). Description Logic (DL) is used to define the actions with their characteristic properties, which enables the efficient classification of new action instances by an OWL reasoner.**

The semantic models of hand actions, robot tasks, and workcell resources are interconnected and stored in a Knowledge Base (KB), which enables the efficient pair-wise translation between hand actions and robot tasks. For the reproduction of human assembly processes, actions are converted to robot tasks via skill descriptions, while reusing the action parameters of involved objects to ensure product integrity. We showcase and evaluate our method in an industrial production setting for control cabinet assembly.

I. Introduction

Small and medium-sized enterprises (SMEs) often need to adapt their products to changing customer demands. However, the traditional industrial robot programming methods are designed for large-scale production and require expert knowledge in the automation domain. They are thus less suitable for small-lot production in SMEs, where the robots need to be frequently re-programmed to reflect product changes or variants. Current approaches address this problem with the introduction of intuitive graphical user interfaces (GUIs), such as Rafcon [1] or our own solution from previous work [2]. Such GUI-based methods allow the user to compose a sequence of robot tasks for product reconfiguration. Nevertheless, the manual process of constructing task sequences can be time-consuming and requires knowledge in working with such systems.

Programming by Demonstration (PbD) with passive observation stands out as the most intuitive approach in robot programming, which enables the operator to perform the demonstration in a high Degree of Freedom (DOF) using their body and requires almost no extra training [3].

*J. Ding, H. Zhang, W. Li, A. Perzylo, L. Zhou are with fortiss, Research Institute of the Free State of Bavaria associated with Technical University of Munich, Guerickestraße 25, 80805 München, Germany.
Please direct correspondence to ding@fortiss.org

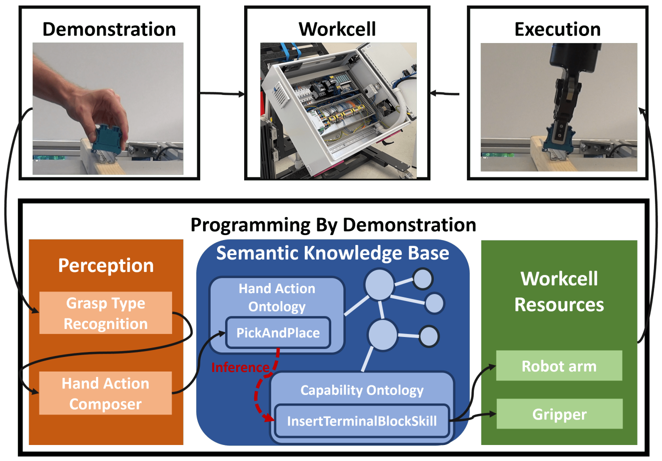

Fig. 1: Concept of our kb-PbD paradigm with the industrial use case of control cabinet assembly. For the correct reproduction with robots, the product-specific actions have to be recognized and linked to the robot skills.

Different approaches have been introduced in recent years for manufacturing tasks such as Peg-in-hole or Slide-in-the-groove [4]. However, these methods focus on teaching kinematics or policies for individual assembly operations, whereas product reconfiguration at SMEs requires the generation of robot task sequences.

In flexible manufacturing systems, skills are defined as parameterizable, executable function blocks that provide specific functionalities [5]. For example, an *InsertTerminalBlock* skill offers the product-specific functionality of inserting terminal blocks onto DIN rails in a control cabinet assembly process as depicted in Fig 1. Such skills possess distinct characteristics across different manufacturing scenarios, and their parameterization is critical for ensuring safe and correct production that satisfies the product configuration, which in the case of the *InsertTerminalBlock* skill, are the specific type of terminal block and its installation positions on the DIN rail. Notably, humans can also perform actions analogous to the *InsertTerminalBlock* skill with equivalent functionality in production. Consequently, in this work, we extend the skill definition to encompass the human action domain. Within this concept, we address the challenges in PbD for SMEs by categorizing the product-specific human actions as skills, and the correct parameterization for robot tasks to fulfill product specifications.

Current vision-based action recognition methods require training on dedicated video datasets [6]. However, existing datasets, e.g., presented in [7], only involve everyday activities and are not directly applicable to industrial contexts, such as recognizing an *InsertTerminalBlock* action. A comprehensive dataset for hand actions in manufacturing settings is still missing, due to the large variety of product-specific hand actions across different production scenarios. In addition, SMEs often lack the expertise required for data collection and annotation in their production environments, which further limits the applicability of current methods. Despite the manifold hand action types in production, the human grasps exhibit recurring shapes that can be categorized within a grasp taxonomy [8], facilitating the generalization of the vision-based action recognition across different production scenarios. Furthermore, the characteristics of the human grasps described in the grasp taxonomy are useful for robot task parameterization.

In this work, we propose a knowledge-based Programming by Demonstration (kb-Pbd) paradigm to facilitate robot programming in SMEs. We showcase the usability of the proposed method within an industrial assembly setting for control cabinet production. Our main contributions in this work can be summarized as: a) A hand action recognition method is proposed and implemented utilizing features of grasp type and hand velocity, saving the effort for collection of the dedicated datasets for product-specific hand actions; b) We semantically modeled hand actions in Web Ontology Language (OWL) incorporating task-level action properties. The product-specific actions are defined in logical expressions with restrictions on action properties, allowing the automatic classification with OWL reasoner; c) Robot executable tasks are efficiently constructed and parameterized from the product-specific hand actions with their functionality description.

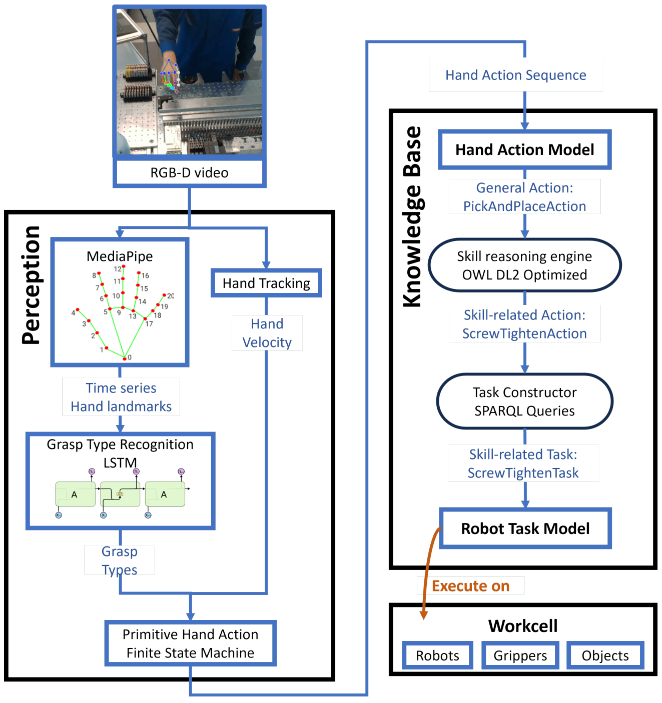
RGB-D video
Perception
MediaPipe
Hand Tracking
Hand Velocity
Time series Hand landmarks
Grasp Type Recognition LSTM
Grasp Types
Primitive Hand Action Finite State Machine
Hand Action Sequence
Knowledge Base
Hand Action Model
General Action: PickAndPlaceAction
Skill reasoning engine OWL DL2 Optimized
Skill-related Action: ScrewTightenAction
Task Constructor SPARQL Queries
Skill-related Task: ScrewTightenTask
Robot Task Model
Execute on
Workcell
Robots
Grippers
Objects

Fig. 2: Overview of the kb-PbD paradigm with a perception pipeline for hand action recognition from RGB-D images (left). The KB semantically encodes and stores the hand action instances, within which a reasoning engine classifies the skill-based actions via logical inference (top right). Robot tasks are constructed from hand actions and executed in the workcell (bottom right).

II. Related Work

Grasp taxonomies [8] offer a comprehensive framework for describing and categorizing grasp types, which has been widely researched for PbD methods. [9] and [10] focused on reproducing robotic grasps including the generation of approach trajectories from grasps in human demonstration. In [11] and [12], the utility of grasp taxonomies have been investigated in a Virtual Reality (VR) environment, enabling the accurate recording of human grasps with detailed information on the contact points and associated normals, which facilitates the planning of pregrasp trajectories of robots.

However, instead of generating new robot grasp policies from human demonstration, PbD for industrial use cases often requires safe and robust reproduction using available robot skills, which involves the recognition of product-specific actions and the reproduction at a robot-centric task level. [13] introduced a task-level PbD (TLPbD) with the kinesthetic teaching of a robot, resulting in significant efficiency gains compared to a conventional GUI-based programming method. The skill recognition is enabled by utilizing the Planning Domain Definition Language (PDDL) for skill descriptions, which reasons under the closed-world assumption.

Web Ontology Language (OWL) is often used for knowledge representation, but reasons under the open-world assumption. Ramirez et al [14] utilized semantic logical expression in Description Logics (DL) to define human activities, enabling the automatic classification of task-related actions by an OWL reasoner. In their following work [15], the OWL rule-based method is validated for teaching robots new skills for industrial tasks such as sorting fruits. Within their works, only action features regarding hand velocities and distances to objects are used to implement a compact perception pipeline. In the scope of this paper, we aim to integrate grasp recognition to expand hand action properties. In addition, OWL has been investigated in various approaches for knowledge representation in industrial assembly scenarios [2], [16]. In this paper, we further investigate the semantic modeling of production-related hand actions and approaches to convert them into robot tasks for reproduction using OWL.

Vision-based action recognition has advanced rapidly in recent years [17]. However, recognizing product-specific actions (skills) in industrial assembly is still challenging due to the lack of comprehensive datasets. For example, [18] collected and annotated a dataset with long action sequences from the assembly processes of toy trucks. Although they proved the generalization of this method to new toy types, the dataset may not be directly used for other industrial types of products. In contrast, the hand action dataset from [7] included hand poses alongside the RGB-D videos, showing the usability of grasp types for action recognition.

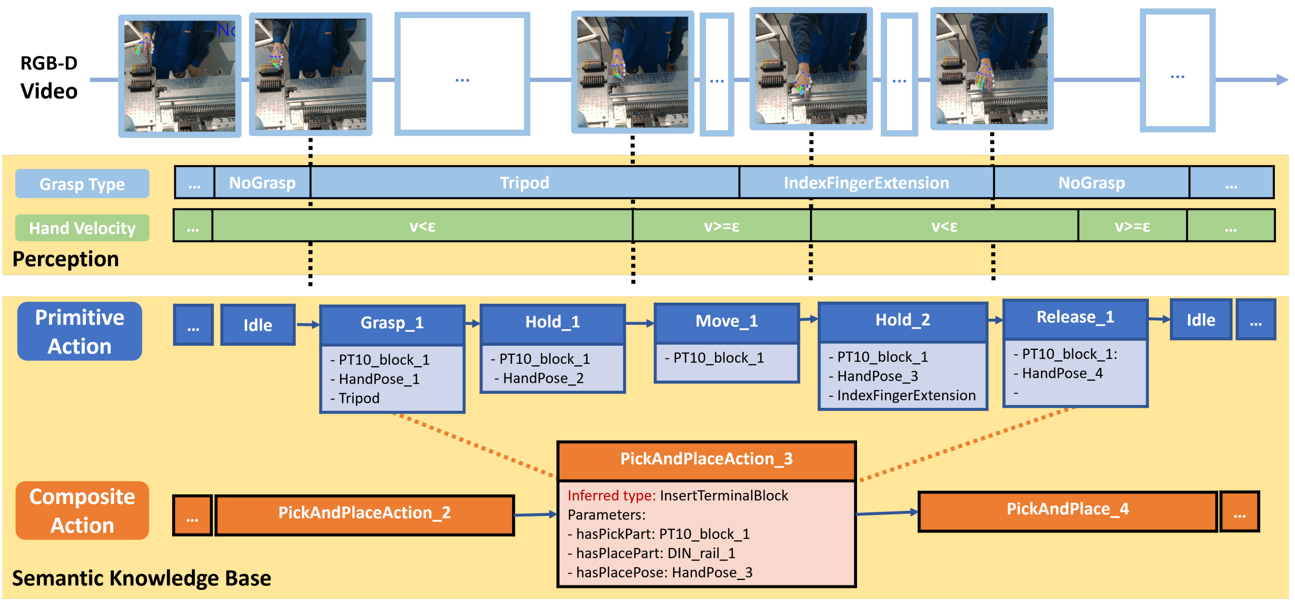
RGB-D Video
Grasp Type
... NoGrasp Tripod IndexFingerExtension NoGrasp ...
Hand Velocity
... v<ε v>=ε v<ε v>=ε ...
Perception
Primitive Action
... Idle
Grasp_1 - PT10_block_1 - HandPose_1 - Tripod
Hold_1 - PT10_block_1 - HandPose_2
Move_1 - PT10_block_1
Hold_2 - PT10_block_1 - HandPose_3 - IndexFingerExtension
Release_1 - PT10_block_1: - HandPose_4 -
Idle ...
Composite Action
... PickAndPlaceAction_2
PickAndPlaceAction_3
Inferred type: InsertTerminalBlock
Parameters:
- hasPickPart: PT10_block_1
- hasPlacePart: DIN_rail_1
- hasPlacePose: HandPose_3
PickAndPlace_4 ...
Semantic Knowledge Base

Fig. 3: An example for action recognition from the RGB-D video. We utilize grasp types and hand velocities (with a threshold ϵ set to 0.3 m/s) as criteria to instantiate the *PrimitiveActions* in the KB. A *PickAndPlaceAction* is composed upon the primitive ones and inferred on the product-specific type defined in DL expressed in 2 on action parameters.

Furthermore, they introduced grasp type recognition using LSTM that processes the hand poses from time series data. Similarly, [19] deployed MediaPipe [20] to generate hand landmarks in an autonomous annotation pipeline for various human activities in the kitchen, which offers valuable insights on action recognition that may be transferred to industrial assembly.

In our earlier concept paper [21], we introduced the concept of grasp type recognition utilizing hand landmarks generated with MediaPipe from RGB images only. In this work, we extend our concept paper with the implementation and evaluation of the kb-PbD paradigm, where a perception pipeline with grasp recognition is combined with a semantic action model for the efficient recognition of product-specific actions. The extended approach is further applied to the industrial use case of equipping wiring cabinets in a prototypical workcell.

III. Concept

An overview of our kb-PbD paradigm is depicted in Fig. 2. Within the perception component (see Sec. III-A), two modalities are used to monitor hand action properties: a Long Short-Term Memory (LSTM) network for grasp type recognition using hand landmarks generated from Mediapipe [20], and a hand tracking module to calculate hand pose and hand velocity from an RGB-D video. Changes in grasp types and hand velocity trigger the generation of new actions through a Finite State Machine (FSM). A Knowledge Base (KB), using a GraphDB[1] repository, persistently stores the semantic knowledge encoded in the Web Ontology Language (OWL). This includes the domains of the hand actions (see Sec. III-B), as well as the workcell, and the robot tasks introduced in our previous work [22]. During the perception phase, the KB is populated with new action instances and their associated properties, such as hand pose, grasp types, and interaction objects. Within the semantic model of hand actions, product-specific actions are defined with restrictions on action properties, enabling the automatic classification of new action instances by an OWL reasoner. As our KB follows a full materialization approach, derived facts are again persistently stored in the semantic repository and are made available for subsequent querying. For robot-based reproduction of actions (see Sec. III-C), robot tasks are pair-wise translated from the human actions using SPARQL queries (SPARQL Protocol and RDF Query Language). A semantic skill model establishes the semantic link between the robot tasks and the human actions that share the same functionality in production, which further enables task parameterization through the reuse of action parameters to satisfy product configurations. The converted processes can then be executed by the semantic manufacturing execution system (sMES) [22] utilizing the resources in a given robot workcell.

A. Perception

This section introduces how basic hand actions are recognized from an RGB-D video and how the high-level actions are composed upon them.

Within the perception pipeline, the grasp type recognition serves as the foundation for tackling the challenge of recognizing diverse hand actions across various production scenarios. We consider the grasp types within hand actions as time series data and train an LSTM on the dataset of hand landmarks for grasp recognition. Mediapipe [20] is used to generate 2.5D hand landmarks with 21 joints as shown in Fig. 2.

[1]https://www.ontotext.com/products/graphdb/

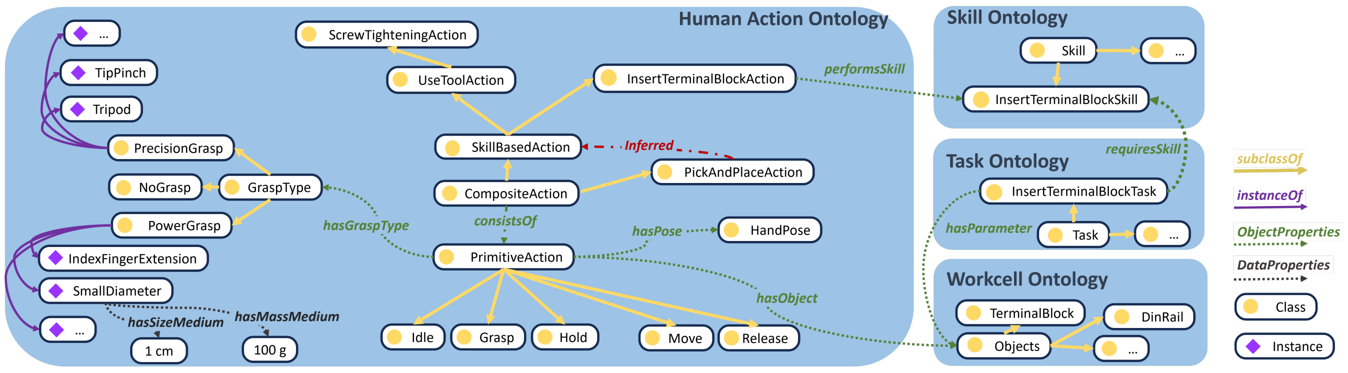

Fig. 4: Overview of partial ontologies within our semantic KB. The action ontology contains the semantic model of *PrimitiveActions* and *CompositeActions*. The product-specific actions are defined as *SkillBasedActions* using Description Logic (DL) and can be automatically classified from general *PickAndPlaceAction*. The skill ontology interconnects the action and the tasks for pair-wise task construct, where the objects inside the workcell are used for parameterization.

During the dataset collection of hand landmarks, the operator performs a single grasp type while holding product parts or tools in hand for 2000 frames in front of the camera. For example, the dataset for grasp type "*SmallDiameter*", as depicted in Fig. 9b, is recorded with a screwdriver held in hand. Since each dataset only contains a single grasp type, the manual annotation on each frame is not needed. The dataset is then trained on the standard LSTM model from TensorFlow[2] to classify the grasp types with a time series length of 30 frames. The compact yet robust 8-layer LSTM model achieved a 96.7% recognition rate on the test set and an average Frames per Second (FPS) of 15, deployed on a computer equipped with an Intel i7-9850H CPU and an Nvidia Quadro T2000 GPU.

A hand tracking module locates the hand position in the RGB-D image and calculates the hand orientation from the normalized hand landmarks. The hand pose in the world coordinate system is further used for the calculation of the hand velocities.

The grasp types and the hand velocity serve as the key properties to identify the basic hand actions, defined as *PrimitiveActions* in this work. Partial processing result of an RGB-D video during assembly is shown in Fig. 3 for illustration. A hand action finite state machine (FSM) is used to generate new *PrimitiveActions* of *Grasp*, *Move*, *Hold*, *Release* and *Idle*, each defined as states within the FSM. For example, a *Grasp* is initiated when the grasp type changes from *NoGrasp* to *Tripod*, while a *Move* action is generated when the hand velocity exceeds a predefined threshold ε of 0.3m/s. Interacted objects maintain crucial contextual information on specific hand actions and are also detected by calculating the nearest object to hand. While object detection does not remain the focus of the work, the objects are placed in a calibrated tray with their positions semantically stored in the workcell description.

A sequence of *PrimitiveActions* further aggregates to form a *PickAndPlace* action, where the *Grasp* and *Release* actions represent the start and the end of it as shown in the second layer of action in Fig.3. The generated hand action sequence, along with associated action-related properties, e.g. grasp types, hand poses, and interacted objects, is then inputted into the semantic model within the Knowledge Base for instantiation. Until this step, only general actions without production context are recognized and will be further classified within the semantic action model.

[2]`https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM`

B. *Semantic hand action model*

This section introduces the semantic hand action model within our Knowledge Base and the inference of the product-specific actions from the general actions using the OWL reasoner.

The perceived actions are stored in the KB as new OWL instances within the hand action model, categorized under two classes as shown in Figure 4: a *PrimitiveAction* class containing subclasses of *Grasp*, *Move*, *Hold* and *Release*, and a *CompositeAction* class containing subclasses of a general *PickAndPlace* action and other *SkillBasedActions*.

The *PrimitiveActions* are instantiated in the ontology and directly linked to the action-level parameters of:

- **Grasp types**, that are semantically modeled as instances under 3 enumerated classes (power, intermediate, and precision), while the properties within the grasp taxonomy [8] are preserved, e.g. the width, handling weight, etc.
- **Hand poses** defined by the hand position $[x, y, z]$ and orientation $[\alpha, \beta, \gamma]$ in the world coordinate system.
- **Interacted objects** including the parts or tools used for production that are stored as OWL instances in the semantic workcell model.

The characteristic properties of each grasp type may offer complementary context for the robot task parameterization during the reproduction. For example, the grasp types like *SmallDiameter* or *Tripod*, both can be performed in a *ScrewTighten* action but indicates different torque values applied to the screw.

The *CompositeActions* are modeled on a higher layer and consist of a sequence of the *PrimitiveActions*. The *PickAndPlace* action, considered as the general type of

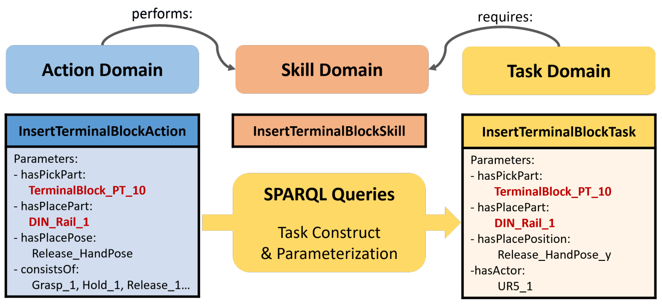

Fig. 5: An example of pair-wise translation from a hand action to a robot task that is semantically connected to the skill domain. The construction and the parameterization of robot tasks are enabled by a SPARQL query (Listing 6), which efficiently reuses the action parameters (marked in red) to ensure the correct reproduction.

CompositeActions, is defined using the Description Logic (DL) in (1) without explicit restrictions on action parameters. The parameters of the *PrimitiveActions* are further extracted and reused for parameterization of the *PickAndPlace*. For example, the parameters of *hasPickObject* and *hasPlaceObject* properties are inherited from the parameters of *performsOn* in the *Grasp* and *Release* actions as shown in Fig 3.

$$\begin{aligned} &PickAndPlace : - \\ &\quad \exists hasPickObject(GeneralObject) \\ &\quad \wedge \exists hasPlaceObject(GeneralObject) \end{aligned} \tag{1}$$

Different from general *PickAndPlace* actions, product-specific actions categorized under *SkillBasedAction* class are distinguished by their unique action parameters, allowing them to be modeled as OWL defined class. In OWL, defined classes can be modeled with necessary and sufficient conditions with restrictions on properties. Instances that fulfill the conditions will be classified to the defined class by the OWL reasoning engine. In our system, product-specific actions are modeled as defined classes under the general *PickAndPlace* action with restrictive conditions on the action properties. For our use case in control cabinet assembly, the product-specific action *InsertTerminalBlock* is modeled as a defined class with restrictive conditions on the types of the interacted objects, as expressed in DL (2). Consequently, when a human picks up a terminal block and places it on the DIN rail, this *PickAndPlace* action is automatically classified as a *InsertTerminalBlock* action by the OWL reasoner, i.e. OWL2 DL Optimized [23] in our system.

$$\begin{aligned} &InsertTerminalBlock : - \\ &\quad \exists hasSuperClass(PickAndPlace) \\ &\quad \wedge \exists hasPickObject(TerminalBlock) \\ &\quad \wedge \exists hasPlaceObject(DinRail) \end{aligned} \tag{2}$$

C. Robot reproduction

This section introduces the process of converting a hand action sequence into a semantic sequence of robot tasks that can be executed on robots for production.

```
INSERT {
    #construct and parameterize a new robot task
    $newRobotTask rdf:type ?TaskType;
            ?hasTaskParameter ?skillRelatedParameters .
} WHERE {
    # find the skill type performed by the given action
    $handAction rdf:type ?actionType.
    ?actionType rdfs:subClassOf action:skillBasedAction;
                skill:performsSkill ?skillType.
    # find the task type requiring the same skill
    ?taskType skill:requiresSkill ?skillType.
    # find action parameters related to the skill
    $handAction ?hasActionParameters ?actionParameters .
    ?actionParameters rdf:type skill:hasObjectParameters .
    # find template task parameters
    ?taskType ?hasTaskParameters ?taskTemplateParameters.
    # Pairing of parameters of action and task
    ?actionParameters rdf:type ?parameterTypes;
    ?taskTemplateParameters rdf:type ?parameterTypes. }
```

Fig. 6: A pseudo SPARQL query for construction and parameterization of a robot task from a given human action. The task type is determined by skill-level matchmaking with the action. The task reuses the action parameters for correct reproduction.

Skills, as defined in [2], are parameterizable program blocks that fulfill certain functionalities on the production. For the efficient reuse of skills, robot tasks are semantically modeled to represent individual assembly steps within the production process. Each task requires the corresponding skill implementation on robots for execution. The tasks and the skills are semantically connected with the OWL properties of *requiresSkills*, which allows a semantic Manufacture Execution System (sMES) [24] to dynamically execute the tasks on the workcell components that offer the required skills. For example, a *GraspTask* can be executed on any robot grippers within the workcell that possess the Grasp-Skill, which is a Robotiq 2f-85 Gripper in the given workcell as shown in Fig.7. Furthermore, high-level tasks, such as the general *PickAndPlace* task or product-specific tasks, can be composed from a sequence of basic ones.

Product-specific hand actions, which share the same functionalities as the robot skills, can be semantically linked through an OWL property of *performsSkills*. This property further enables the pair-wise match between hand actions and robot tasks as shown in Fig.5. Thus, the translation of the hand action sequence to a robot task sequence is enabled.

The reproduction that satisfies the product integrity requires the generation of robot tasks with the correct task types and the correct parameterization. During the action recognition, the action parameters, such as interacted objects, are also stored as OWL instances in our KB, allowing them to be efficiently reused for the parameterization of robot tasks. The parameters essential for the product configuration are defined under the skill domain in the ontology to distinguish them from the general ones. For example of the *InsertTerminalBlock*, parameters like the specific type of terminal blocks and the DIN rail to be installed, are most essential for configuring a specific type of control cabinet. The construction and parameterization of robot tasks are enabled through a SPARQL Query (Listing 6) that iteratively

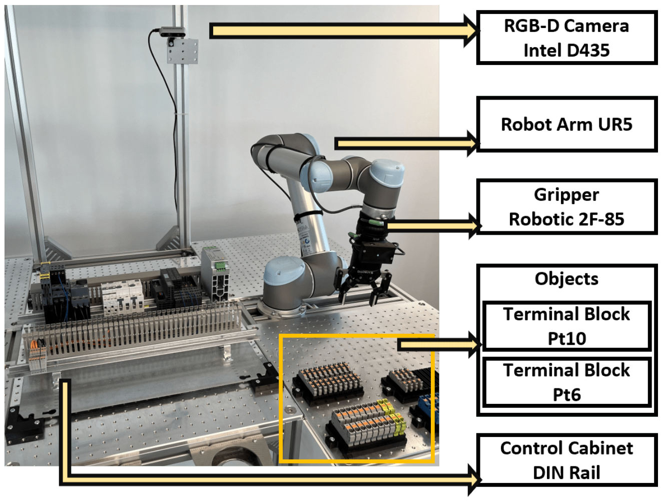

Fig. 7: Workcell setup for evaluation with the industrial use case of control cabinet assembly.

performs the following steps: (1) takes an instance of the hand action each time and identifies the skill performed by the human on current step; (2) determine the corresponding robot task type associated with the skill type; (3) create a new task instance from the task template of this type; (4) identifies the reusable action parameters that are relevant for product configuration and replace the task template parameters with them.

IV. Experiments & Dicussion

The reproduction within our proposed kb-PbD requires the semantic model of robot tasks and the implementation of the product-specific robot skills, which makes a direct comparison with other PbD methods difficult. However, the robot tasks are executed with the skills that yield deterministic results with the correct parameterization, which further depends on the correct recognition of the product-specific hand action. Thus, we focus on the evaluation of the usability of our knowledge-based action recognition with two experiments. In the first experiment (see IV-A), we compare our knowledge-based action recognition with a baseline method using a cnn+lstm network proposed by [25] in an industrial setting, as shown in Fig 7, where the action recognition accuracies of both methods are compared, and the generalization for other production scenarios is discussed. Subsequently, in the second experiment (see IV-B), we evaluate the modeling capability of our DL rule-based action model on a public dataset [7] with a large variety of hand actions. In addition, a preliminary comparison of the programming efficiency between the kb-PbD and a conventional GUI-based programming approach is conducted in Sec. IV-C.

A. Baseline comparison

1) Experiment setup: A robot workcell, depicted in Fig. 7, is set up to match a real production scenario for electrical control cabinets. As part of the workcell, a UR5 robot is equipped with a Robotiq 2F-85 parallel gripper and offers the primitive skills of *MoveArm* or *Open/Close Gripper*, as

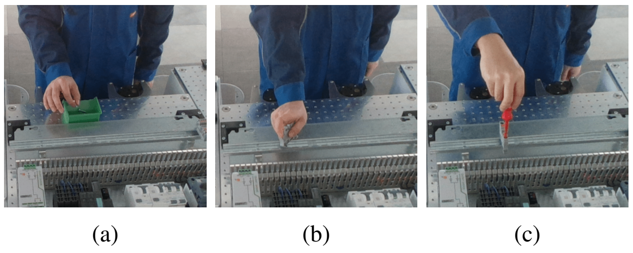

Fig. 8: Overview of required skills for the control cabinet assembly process (from left to right *PickAndPlace*, *InsertTerminalBlock* and *ScrewTighten*).

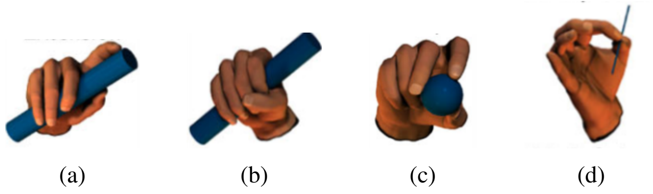

Fig. 9: Overview of involved grasp types within the hand actions from Fig. 8 (from left to right: *IndexFingerExtension*, *SmallDiameter*,*Tripod* and *TipPinch*).

well as the composite skills of *PickAndPlace* and *InsertTerminalBlock*. On a DIN mounting rail, different types of terminal blocks can be installed at arbitrary positions. The objects in the workcell are placed in the calibrated trays for precise robot manipulation. An Intel D415 camera perceives the RGB-D image of the human demonstrations from an overhead perspective. In the manual production of a control cabinet, three distinct types of human actions are performed: a general *PickAndPlace* action 8a and 2 product-specific actions, *InsertTerminalBlock* 8b and *ScrewTighten* 8c.

Due to the lack of hand action datasets that covers the above-mentioned hand actions, we collected our own datasets for training our LSTM grasp recognition network within our kb-PbD paradigm, as well as for the training of a baseline method using a cnn-lstm network [25]. For training the LSTM network, a dataset of hand landmarks generated from Mediapipe is collected, which contains 4 representative grasp types involved in this production as in Fig. 9, i.e., *IndexFingerExtension* 9a, *SmallDiameter* 9b, *Tripod* 9c, *Tippinch* 9d, along with a *NoGrasp* type. For training the cnn-lstm network from [25], an RGB-D video dataset of human action sequences in assembly is recorded including 80 video sequences for the 3 required actions as in Fig.8 plus an *Idle* action. Subsequently, both methods are evaluated within the workcell, where an operator consequently performs an assembly process of a control cabinet including 15 actions. The results of action recognition are compared to manually annotated labels of each frame in the RGB-D video.

TABLE I: Comparison of our method with a baseline method [25] on recognition accuracy and dataset properties.

Method	Action recognition rate			Dataset	
	Pick AndPlace	Insert TerminalBlock	Screw Tighten	Size (frames)	Reusability
kb-pbd	87.50%	90.13%	92.22%	8000	high
cnn-lstm	75.60%	76.74%	90.48%	7801	low

2) Result: Table I provides a comparative analysis on the recognition accuracy and dataset properties of both methods. Ours could recognize all 3 actions with an average accuracy of around 90%. The recognition accuracy is calculated on whether the *PrimitiveActions* are generated and parameterized on the correct RGB-D frames, since the *CompositeActions* are not recognized on each frame but are classified by the OWL reasoner upon the composition of new *PickAndPlace* actions. Within this experiment, all 15 *CompositeActions* within the assembly process are recognized with the correct parameterization. The cnn-lstm method has similar recognition accuracy for the *ScrewTighten* action compared to ours, but is less accurate on the *PickAndPlace* and *InsertTerminalBlock* action. This could be caused by the similar features of both actions in the RGB-D videos.

TABLE II: Result of grasp type recognition in experiment IV-A.2. Frames with low confidence from MediaPipe are abandoned.

Total actions	Total Frame	Correct	Abandoned	Accuracy
15	3029	2130	121	73.24%

A detailed result on the grasp recognition within our kb-PbD paradigm using Mediapipe and LSTM is presented as in table II. During the evaluation, the hand-camera distance was increased compared to the distance during dataset collection, resulting in unreliable hand landmarks generated from Mediapipe. To ensure the correct hand action recognition, the hand landmarks with confidence below 50% are abandoned, yielding an overall accuracy in grasp recognition of 73.23%. Our action recognition method shows robustness (around 90% in TableI) against false grasp recognition for two reasons: (1) only changes in grasp type from or to *Idle* would instantiate new hand actions, where false recognized grasp types within an action has no effect and (2) the grasp type *Idle*, indicating a fully open hand, is more distinctive to other grasp types with object in hand.

To conclude, our method resulted in higher accuracy in recognizing product-specific actions than the cnn-lstm method. In addition, the cnn-lstm only recognized action types but not the action parameters, while our method also recorded the action parameters in the KB during the recognition, facilitating the efficient reuse in parameterization of robot tasks. The dataset size reflects the implementation effort of the method, which is similar for both methods in this single production scenario. However, the grasp dataset can be easily expanded and reused for other production scenarios, whereas the cnn-lstm method requires the recording of new production-specific datasets and thus, is less applicable for SMEs. The robot reproduction of the human assembly process of the control cabinet can be found in the complementary video.

B. Generalization of action recognition

Our method requires manually engineered logical expressions in DL to define product-specific hand actions, which are critical for action recognition with the OWL reasoner. In this experiment, we evaluate whether our semantic hand action model would satisfy the need for modeling a large variety of hand actions. Due to the lack of a comprehensive description for product-specific hand actions in production, we hence modeled the action types from an everyday action dataset [7] using DL.

TABLE III: Example of everyday actions that can be successfully modeled in our semantic model.

Skill	Rescrictive properties	
	hasPickObject	hasPlaceObject
PutTeabag	Teabag	Mug
CloseMilk	MilkLid	MilkBottel
	hasPickObject	hasGraspType
SqueezeSponge	Sponge	SmallDiameter
ScratchSponge	Sponge	AdductedThumb

In a total of 44 everyday actions with interacted objects (exclude *HighFive*), 31 can be distinctively modeled as defined OWL classes in the semantic model. 20 actions can be modeled with their unique interacted objects, such as *PutTeabag* or *Open/CloseMilk*. 11 actions may have overlapping object properties but can be further distinguished with performed grasp types. For example, the *SmallDiameter* and the *AdductedThumb* grasps are differently performed in the *scratch/squeeze sponge* actions. Some representative action classes that are successfully defined are given in Table III.

Among the 13 actions that we failed to model, 8 actions could only be distinguished by their characteristic fine hand movements that are not perceivable from our current perception modalities. For example, either *Sprinkle*, *ScoopSpoon* or *Stir* is performed with the *Stick* grasp on the spoon handle that can only be grouped under a *UseTool(Spoon)* action in our model. For the same reason, *DrinkingMug* is still considered a failed model since the tilt action during drinking differs itself from a general *PickAndPlace(Mug)* action. 5 other failed actions share the duplicate action properties but result in changes in the object's shape or pose, such as the *FlipSponge* caused the rotation of a sponge, or the *TearPaper* and *OpenEnvelope* result in object shape change. The underlying cause for these failures is the lack of perception modalities. However, our framework remains adaptable to other perception modalities and is capable of encoding their results in our KB for recognition of the new action properties.

C. Preliminary efficiency evaluation for PbD

To offer a preliminary overview of the efficiency of our PbD robot programming method, we conducted a comparative experiment against a conventional task-level programming approach using a self-implemented GUI similar to RAZER [26]. The operator is instructed to program four predefined robot assembly processes of a control cabinet with both methods. Each assembly process contains five *InsertTerminalBlock* tasks including two types of terminal blocks. In the first two processes, the installation positions of the terminal blocks on the rail were not explicitly defined, while in the last two, an insertion on a precise position was required, imitating the production scenarios where the position of the terminal block on the rail is critical for the product configuration. Within the PbD process, the installation positions were read from the hand poses of the *Release* actions after the insertion. The GUI-based approach offered a slider element corresponding to rail positions. This design allowed us to have a first look at programming efficiency with and without precision considerations. A preliminary result indicates that half of the time required for the GUI-based programming can be saved with a PbD-based approach. However, when precise parameterization of a manufacturing process is considered, GUI-based programming may be beneficial. A more comprehensive experiment with more participants is to be conducted in the near future.

V. Conclusion

Within this work, we proposed a kb-PbD paradigm with the integration of a grasp taxonomy and semantic models, enabling the efficient recognition of product-specific actions across various industrial assembly processes without the need to train the perception on dedicated datasets. The product-specific actions are defined in DL and related to their functionalities during production, which enables the generation of correct assembly steps, and the reuse of action parameters for product integrity. The reproduction of the human-demonstrated assembly is achieved on a task level. We further evaluated and showcased the system in a control cabinet assembly workcell to prove its usability. Within the current system, the grasp types are recognized for generating new hand basic movements. In future works, this grasp-related context can be utilized for the parameterization of robot tasks, or even for the composition of new skills. Our proposed method is capable of integrating new perception modalities and aggregating the perception results in the KB, allowing the recognition of more complicated actions. We believe the proposed kb-PbD method can be used in other industrial scenarios and also in service robotics for manipulating household objects.

References

[1] S. G. Brunner, F. Steinmetz, R. Belder, and A. Dömel, "Rafcon: A graphical tool for engineering complex, robotic tasks," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 3283–3290.

[2] A. Perzylo, N. Somani, S. Profanter, I. Kessler, M. Rickert, and A. Knoll, "Intuitive instruction of industrial robots: Semantic process descriptions for small lot production," in *2016 ieee/rsj international conference on intelligent robots and systems (iros)*. IEEE, 2016, pp. 2293–2300.

[3] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual review of control, robotics, and autonomous systems*, vol. 3, pp. 297–330, 2020.

[4] Z. Zhu and H. Hu, "Robot learning from demonstration in robotic assembly: A survey," *Robotics*, vol. 7, p. 17, 2018.

[5] A. Perzylo, J. Grothoff, L. Lucio, M. Weser, S. Malakuti, P. Venet, V. Aravantinos, and T. Deppe, "Capability-based semantic interoperability of manufacturing resources: A basys 4.0 perspective," *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 1590–1596, 2019.

[6] H.-B. Zhang, Y.-X. Zhang, B. Zhong, Q. Lei, L. Yang, J.-X. Du, and D.-S. Chen, "A comprehensive survey of vision-based human action recognition methods," *Sensors*, vol. 19, no. 5, p. 1005, 2019.

[7] G. Garcia-Hernando, S. Yuan, S. Baek, and T.-K. Kim, "First-person hand action benchmark with rgb-d videos and 3d hand pose annotations," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 409–419.

[8] T. Feix, J. Romero, H.-B. Schmiedmayer, A. M. Dollar, and D. Kragic, "The grasp taxonomy of human grasp types," *IEEE Transactions on human-machine systems*, vol. 46, no. 1, pp. 66–77, 2015.

[9] S. Ekvall and D. Kragic, "Grasp recognition for programming by demonstration," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 748–753.

[10] ——, "Learning and evaluation of the approach vector for automatic grasp generation and planning," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 4715–4720.

[11] J. Aleotti and S. Caselli, "Grasp recognition in virtual reality for robot pregrasp planning by demonstration," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 2801–2806.

[12] ——, "Grasp programming by demonstration in virtual reality with automatic environment reconstruction," *Virtual Reality*, vol. 16, pp. 87–104, 2012.

[13] F. Steinmetz, V. Nitsch, and F. Stulp, "Intuitive task-level programming by demonstration through semantic skill recognition," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3742–3749, 2019.

[14] K. Ramirez-Amaro, M. Beetz, and G. Cheng, "Transferring skills to humanoid robots by extracting semantic representations from observations of human activities," *Artificial Intelligence*, vol. 247, pp. 95–118, 2017.

[15] K. Ramirez-Amaro, E. Dean-Leon, F. Bergner, and G. Cheng, "A semantic-based method for teaching industrial robots new tasks," *KI-Künstliche Intelligenz*, vol. 33, pp. 117–122, 2019.

[16] D. Beßler, M. Pomarlan, and M. Beetz, "Owl-enabled assembly planning for robotic agents," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, 2018, pp. 1684–1692.

[17] Y. Kong and Y. Fu, "Human action recognition and prediction: A survey," *International Journal of Computer Vision*, vol. 130, no. 5, pp. 1366–1401, 2022.

[18] F. Sener, D. Chatterjee, D. Shelepov, K. He, D. Singhania, R. Wang, and A. Yao, "Assembly101: A large-scale multi-view video dataset for understanding procedural activities," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 21 096–21 106.

[19] N. Elangovan, R. V. Godoy, F. Sanches, K. Wang, T. White, P. Jarvis, and M. Liarokapis, "On human grasping and manipulation in kitchens: Automated annotation, insights, and metrics for effective data collection," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 11 329–11 335.

[20] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann, "Mediapipe hands: On-device real-time hand tracking," *arXiv preprint arXiv:2006.10214*, 2020.

[21] J. Ding, A. Perzylo, and L. Zhou, "A knowledge-augmented concept for programming by demonstration based on hand-object actions," in *International Workshop on Working towards Ontology-based Standards for Robotics and Automation (WOSRA), ICRA*, 2023.

[22] A. Perzylo, I. Kessler, S. Profanter, and M. Rickert, "Toward a knowledge-based data backbone for seamless digital engineering in smart factories," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1. IEEE, 2020, pp. 164–171.

[23] A. Chortaras, D. Trivela, and G. Stamou, "Optimized query rewriting for owl 2 ql," in *Automated Deduction–CADE-23: 23rd International Conference on Automated Deduction, Wrocław, Poland, July 31-August 5, 2011. Proceedings 23.* Springer, 2011, pp. 192–206.

[24] A. Perzylo, M. Rickert, B. Kahl, N. Somani, C. Lehmann, A. Kuss, S. Profanter, A. B. Beck, M. Haage, M. R. Hansen *et al.*, "Smerobotics: Smart robots for flexible manufacturing," *IEEE Robotics & Automation Magazine*, vol. 26, no. 1, pp. 78–90, 2019.

[25] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4694–4702.

[26] F. Steinmetz, A. Wollschläger, and R. Weitschat, "Razer—a hri for visual task-level programming and intuitive skill parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1362–1369, 2018.