J Intell Robot Syst (2015) 80 (Suppl 1):S149–S163
DOI 10.1007/s10846-015-0219-x

Gesture-Based Extraction of Robot Skill Parameters for Intuitive Robot Programming

Mikkel Rath Pedersen · Volker Krüger

Received: 30 May 2014 / Accepted: 11 February 2015 / Published online: 24 February 2015

Abstract Despite a lot of research in the field, only very little experience exists with Teaching by Demonstration (TbD) in actual industrial use cases. In the factory of the future, it is necessary to rapidly reprogram flexible mobile manipulators to perform new tasks, when the need arises, for which a working system capable of TbD would be ideal. Contrary to current TbD approaches, that generally aim to recognize both action and where it is applied, we propose a division of labor, where the operator manually specifies the action the robot should perform, while gestures are used for specifying the relevant action parameter (e.g. on which object to apply the action). Using this two-step method has the advantages that there is no uncertainty of which action the robot will perform, it takes into account that the environment changes, so objects do not need to be at predefined locations, and the parameter specification is possible even for inexperienced users. Experiments with 24 people in 3 different environments verify that it is indeed intuitive, even for a robotics novice, to program a mobile manipulator using this method.

M. R. Pedersen (✉) · V. Krüger
Robotics, Vision and Machine Intelligence (RVMI) Lab, Department of Mechanical and Manufacturing Engineering, Aalborg University Copenhagen, AC Meyers Vaenge 15, 2450 Copenhagen, Denmark
e-mail: mrp@rvmi.aau.dk

V. Krüger
e-mail: vok@rvmi.aau.dk

Keywords Industrial robots · Gesture recognition · Teaching by demonstration · Robot skills

1 Introduction

Traditionally, industrial robots are only economically feasible when factories produce large quantities of parts with low variation. However, customers are demanding more and more configuration options for products. In order to maintain production volumes, while addressing this need from the customers, a shift from traditional mass production to mass customization and lotsize-1 production, which deals with efficient production of very small batch sizes, is needed. This industrial demand is the starting point of several projects funded by the European Commission (EC) [1–3].

In order for manufacturing companies to offer mass customization and lotsize-1 production, *transformable* production systems are needed [17–19]. This means production systems that are able to adapt to new short-term and long-term tasks. Currently, industrial robots are lacking the transformability to be a part of these production systems [17–19]. In the robotics context, transformability means that the robot should be able to complete previously unknown tasks, within the constraints of the manufacturing environment. This is a generalization of the term *flexibility*, where the robot should be able to handle a previously defined variation within a limited set of tasks. Robots with some

degree of flexibility are already present in the industry today. However, transformable robots can be expected to appear in the factories of the future. Transformable robots are typically characterized as robots that are capable of solving a large variety of tasks, by incorporating a certain level of cognitive capabilities. In many cases, but not all, this entails mobile manipulator systems.

Mobile manipulators are already present in industrial scenarios today [22]. However, for mobile manipulators to be truly transformable in factories, a key issue is that it must be possible to program them to perform new tasks on the fly. This means that it should be both *fast* and *intuitive* to program a transformable robot. Furthermore, the factory should not need to employ a large group of robot specialists to program the robots, when it is needed. Instead, it must be possible for shop-floor workers to program and interact with the robot, so the robot fleet essentially becomes a tool at the workers disposal. Here, intuitiveness is key – it must be so intuitive to program a flexible robot, that a short introduction, or very little training, is sufficient. One approach to intuitive robot programming is Teaching by Demonstration.

Teaching by Demonstration (TbD) generally aims to program robots by recognizing a complex human action, or *task*, and emulate this task on a robot [31]. Both the type of action and parameters of the action, i.e. where or on what the action is applied, needs to be extracted. In many approaches, the focus is to divide the continuous action into fundamental motions of a human, and map these to similar fundamental motions supplied to the robot a priori, where most research focus is on modeling the robot motion. These fundamental motions are often called *motion primitives*, *motor primitives*, *action primitives* or similar [31], and are usually modeled on a low level, such as kinematic states of joints or limbs of a robot.

In order to instead communicate explicitly to the robot which actions it should perform, it is useful to look at human language. Like action, it is constructed from smaller components, such as phonemes at the lowest level, and words as higher level semantic blocks [8, 9, 32]. A similar higher level block in the robot action context can be called a robot *skill* [5], which is not merely e.g. joint movements, but instead contains both sensing and action. For identifying a meaningful set of robot skills, it makes sense to relate the human language to robot action, and associate verbs such as “place” or “drive” with robot skills. This enables humans to instruct a robot in the exact same way as they would instruct another human, and should assure a high level of intuitiveness. By implementing skills in this manner on a robot, we can define robot programs, or tasks, simply as parameterized robot skills, where the parameter is e.g. where to place or drive to.

For the robot skills to be intuitive, they must also be sufficiently general, e.g. there should only be one *pick up* skill, that is able to handle every object the robot will encounter in the factory. The skills must also have some cognitive capabilities contained within them, i.e. both sensing capabilities, world knowledge, and context modeling, again in order to handle as many cases as possible with the same skill. When a robot is equipped with even a small number of these skills, they can be concatenated to solve complex tasks in the factory.

One way to use skills to construct full tasks is to divide the programming into two steps: *Sequencing* and *Teaching*. In the sequencing step, the operator specifies the sequence of robot skills, in the right order, that would solve the desired task. This step eliminates the uncertainty in recognizing human action, as the exact skill is manually defined by the operator. It can be done in a very short time, and should assure that the robot is not performing any unexpected actions. When the sequence has been specified, the parameters for each skill need to be supplied. This is the teaching step, and in this paper we propose to specify the parameters using human gestures, e.g. pointing at the object or location to specify which object to pick up and where to place it. In this way, we reduce the general TbD problem to be only recognition of action parameters.

In this paper we present the intuitive programming of a robot system for logistic tasks, leveraging the approach outlined in the previous paragraphs. As an example task we choose an industrial use case, where boxes are regularly filled with parts from an assembly machine. These boxes need to be replaced with

empty ones, whenever they are full. In this case, the involved general skills would be e.g. *drive to* and *place on*, and the parameters simply the assembly station or the warehouse, where full boxes are placed. This use case is a good example of the kind of tasks that are performed by human operators in the industry today – simply because they are too difficult to automate. We restrict the skills to handle this scenario to ensure robust skills, and keep focus on the proposed task programming method.

This paper builds upon our previous work, described in [37], where we in this paper have increased the number of experiments, and included qualitative feedback from all participants. Since it is not feasible to explicitly program reoccurring skill sequences for each box (the robot can only carry one box, and the number of boxes at the station may vary), we also investigate a method for including loops in the presented programming approach.

The contribution of this work is the following:

1. We demonstrate gesture-based teaching of task parameters, where the task is a previously defined sequence of general robot skills. By specifying the skill sequence manually, it is only necessary to recognize the parameter of the action at programming time, and not the action itself.
2. We present a detailed and cohesive description of the suggested approach, and its implementation on a robot system.
3. We have verified the approach with 24 test subjects in 3 different environments, including one industrial environment, and with varying complexity of the available skills.
4. We discuss how this approach extends to other scenarios than the use case programmed in the experiments.

In the following section we will describe the prior work related to this approach. In Section 3 we will supply the reader with an overview of the robot skills that form the basis of the programming system. Section 4 describes the implementation of the system, comprised of a set of skills, a touchpad-based GUI and gesture recognition. The approach is verified in Section 5, where the experiment setup and results are described, and we provide a thorough discussion of the proposed method in Section 6.

2 Related Work

As mentioned already in the introduction, in this work, we are not recognizing action, and replicating it on a robot. We rather supply the robot with basic actions, or *skills*, and recognize a pointing gesture, which specifies where to apply the skill.

The idea of robot skills is not a new one [4, 7, 27, 31], but the use of robot skills for intuitive programming in the industrial context is not gaining much attention from the research community. When it comes to programming of industrial robots, research focus seems to be on offline programming by using CAD models [11, 35, 36] or online programming using augmented reality [6, 14, 24]. Both of these approaches, however, do not address the issue of transformable robots, but rather methods for creating robot programs that perform repetitive motions with limited sensing. Another focus is task planning, often using motion primitives as the planning domain [12, 15, 16]. However, the planning domain is usually some model of motion (or sensing) primitives, where the system tries to plan a complete task from the primitives. In this approach, we supply the robot with a skill abstraction in between the tasks and the primitives.

A use of motion primitives, that somewhat resemble our notion of general robot skills, is the Manipulation Primitives (MPs) described in [28, 29]. This work expands on the Task Frame Formalism (TFF) [10], by enabling the use of any sensor and controller in the same task frame (e.g. visual servoing combined with force control), as well as operating in a dynamic task frame. These MPs have also been combined in Manipulation Primitive Nets [13], with simple sensor-level decisions based on the outcome of each single MP in the net. This makes it possible to program and execute a task such as inserting a battery in a cellphone, based on realtime feedback from force/torque sensors. However, these nets are still too complex to use for robot novices, as they require detailed parameterization in the form of e.g. force/torque limits.

A layered structure of skills and tasks similar to ours is also proposed in [21], where skills are concatenated to form tasks. However, the focus is planning on the task level, but by utilizing somewhat general skills and skill primitives. The implementation of said skills is however at its current state rather simplistic.

The notion of skills described in this paper mostly resemble the Action Recipes of RoboEarth [40, 41]. However, in RoboEarth, the skills (or Action Recipes) keep sensing and action separate. With its focus on knowledge sharing between robots, the Recipes (tasks) in RoboEarth rather require some capabilities of the robot, such as object recognition and environment models, and the Recipes are only action based.

Programming by Demonstration has been an active research topics for a number of years, and much research has been carried out in the field on recognizing human action [23, 25, 26, 30, 33, 39, 42]. This work focuses instead on what we believe is an equally valid approach to recognize human action, i.e. implementing similar actions, or skills, on the robot.

In this work we show a skill-equipped robot, that along with gesture recognition capabilities can be used for intuitive Human-Robot Interaction (HRI). The skills only require simple parameters as input, so both the outcome of the skill and the required parameters are intuitive, even for robotics novices.

3 Robot Skills

The core idea of robot skills is that they are fundamental software building blocks, that can be used to form more complex tasks. The skills should be applicable in all relevant scenarios the robot encounters. This implies that the skills need to be very general, and the motion and sensing functionalities contained within them need to be equally general. This makes the skill implementation more complex, but only internally, inside the skill – the end-users of the skills, i.e. the shop floor workers, need not worry about this.

A key issue is to find the right set of skills, which can solve most tasks in the transformable production. We have previously analyzed more than 500 human operator instructions at Grundfos, a Danish manufacturer of domestic water pumps [5]. The instructions were written in the form of Standard Operating Procedures (SOPs). The SOPs specify in writing and images how the human worker should perform various tasks around the factory, such as operating assembly equipment. The analysis revealed no more than 13 reoccurring general actions, such as "pick up this object" or "insert part P into machine M." Thus, a robot equipped with only these 13 *general* skills would be able to solve all the analyzed tasks, that are currently performed by human workers.

By implementing skills that resemble the steps in the SOPs, the outcome of each skill is immediately intuitive for the factory workers. Skills such as "pick up" or "drive to" require no further explanation for a robotics novice, as it is exactly how actions are instructed to another human. Since the skills are easy to understand, it is also intuitive for the shop floor workers to use the skills for task programming. By manually specifying the action, or skill, the operator only needs to specify where to apply the skill, e.g. objects to pick up or locations to drive to.

In the next section we will introduce our model of a robot skill, that is sufficiently general and robust for industrial scenarios.

3.1 Skill Model

Our model of a complete robot skill is shown in Fig. 1. The full model includes several aspects, but here we will only focus on the core aspects of execution, parameterization, and pre- and postconditions. The prediction of the outcome of a robot skill is mainly useful for task planning, which is not relevant for this work. For a more detailed description of the skill model, we refer to our previous work [5].

The *execution* block of the skill is not unlike a traditional robot program. However, the execution needs to be sufficiently general to be applicable for all use cases. For instance, the execution for the "pick up" skill needs to be able to handle every variety of objects in the factory, which the robot should manipulate. The execution is composed of *skill primitives*, that serve a single purpose related to the robot system. At its core, the purpose of this is hardware abstraction and ease of skill implementation. Consider two different robots which each have skill primitives that e.g., actuates a gripper, detects objects, or performs robot

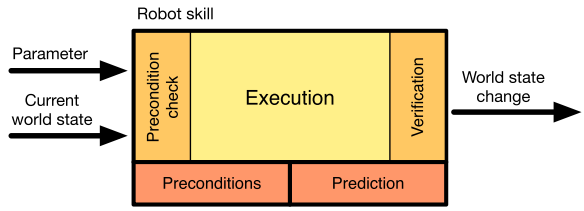


Fig. 1 Model of a robot skill

arm motions. Then, skills that rely on these primitives can be executed on both robots, which we have previously shown in [38].

The *parameters* for the skill are two-fold. As skills are object-centered, only simple and intuitive parameters are needed for input, e.g. a *location* to place an object or an *object* to pick up. These are the parameters that are supplied at task programming time. Everything else is handled within the skill. Upon skill execution, the necessary calculations are made in order to successfully execute the skill, based on the supplied parameters. These calculations are made based on lower-level intrinsic parameters, that the end user does not have to worry about, but is instead up to the skill programmer. There are two types of intrinsic parameters, where one is related to skill variation, and one is related to the hardware, accessed through the skill primitives. For a "pick up" skill, the first type could include the grasp type and orientation, and the second robot speeds, contact forces, etc. Should it be necessary for the end user to explicitly define the intrinsic parameters, such as contact forces, this should also be possible, but will not be included in this work.

In this work, a single parameter for the skills is sufficient, as we will describe later. However, the skill model is easily scalable, given a set of adequate primitives. For instance, more advanced skills such as "insert object A in machine B", still only require simple and intuitive parameters – if and only if the required primitives are available, that can align the object correctly with the machine, e.g. given prior knowledge, and insert the object correctly.

Since the skills need to be robust, they include *pre-* and *postconditions*. By implementing a checking procedure for these conditions, the skills themselves verify their applicability and outcome. This enables the skill-equipped robot to alert an operator if a skill cannot be executed (for precondition failures) or did not execute correctly (for postcondition failures). For instance, the "place object" skill would first have to verify, among other things, if the location to place the object is reachable. After presumably correct execution, it is verified that the object is in fact placed at the desired location. A formal definition of pre- and postconditions is not only useful for robustness, but also task planning. It is easy to imagine a robot that would re-plan a task, if a precondition was not satisfied. This is not the scope of this particular work, but of great interest in our future work.

As the skills need to access prior knowledge of objects and locations in the factory, the robot needs to have a notion of the current state of the world. This can be implemented in multiple ways, mostly depending on the types of tasks the robot must perform, and is still an open question, and not the focus of this paper. However, our previous experiments show that for the skill approach, the world model need only contain information about the valid parameters for the skills running on the robot system, e.g. objects or locations [37, 38].

4 Implementation

In this section we will describe how the approach is implemented on our experimental platform, the mobile manipulator "Little Helper," which is presented first. The approach relies on the combination of skills, gesture recognition for parameter specification, and a user interface for skill sequencing. We will describe these components individually, and go on to describe how these are combined for intuitive programming in the central task programming controller.

4.1 Little Helper

Unlike service robots for domestic applications, which are developed at many universities, the focus of the Little Helper is industrial applications, and for addressing the needs for transformable production. The Little Helper is built from standard industrial components, and is shown in Fig. 2. The robot system consists mainly of three hardware components.

For manipulation we use a 7 DOF KUKA Light-Weight Robot (LWR) arm, for these experiments equipped with a passive gripper for grasping the boxes, which improves the robustness of the skills, since we can abstract away grasp detection. The robot arm is mounted on a chassis which also contains the robot controller and a computer controlling the entire system. The computer is a dual Xeon processor setup with an Nvidia GTX 580 CUDA-capable graphics card. On the chassis is also mounted two Primesense RGB-D sensors, specifically a Microsoft Kinect and an Asus Xtion Pro LIVE. The Kinect is used for object detection, and is mounted close to the base of the

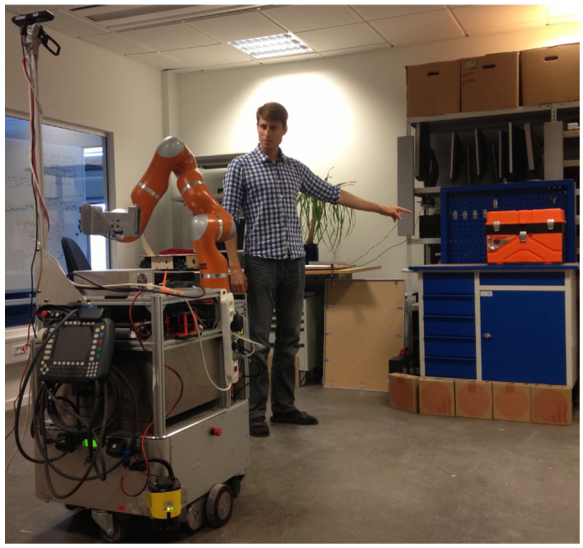


Fig. 2 A user performing a pointing gesture in front of the Little Helper robot

LWR, facing forward on the robot. The Xtion camera is used for human tracking, and is mounted on a vertical pole, to provide a full view of the instructor. The chassis is mounted on top of a Neobotix MP-655 differential drive mobile platform.

4.2 Skill Implementation

In order for skills to be successfully deployed in an industrial setting, they need to be sufficiently general. This means that the manipulation skills, such as "pick up" or "place on" needs to be able to deal with all relevant objects and locations the robot should manipulate. This mainly comes down to how general the skill primitives are, which the skill is composed of.

We have already implemented a number of skills [38], that allows us to test the proposed HRI method for the logistic use case. Since we only focus on these use cases in this paper, we only use the logistic skills. Therefore, we only need skills that are able to detect and manipulate boxes. This allows us to use a subset of simpler, more stable skills for this work on task programming.

For the use case of transporting boxes, the skills only need basic methods for grasping, object recognition and world knowledge. Since we are only manipulating one kind of object, we have equipped the robot with a passive gripper. The passive gripper enables the robot to carry boxes with heavier loads, as it does not rely on keeping a gripper closed. This also means that we only have one valid grasp point for the boxes, abstracting away advanced primitives such as grasp detection, which is not the focus of this paper. For testing this approach, boxes and locations are marked by QR codes, where the data contained in the code specifies the unique object or location. The 3D poses of the QR codes are detected using the Kinect camera on the robot – for details we refer to [37].

For these experiments we have implemented the world model as a database-like program, containing locations of previously detected QR codes and their labels. This enables us to uniquely identify a specific box or location, as well as an easy method for adding new detections, removing old entries or updating information of a specific entry, e.g. when moving a box. Since the width is also included for the locations, the robot can easily query the world model for free space at a certain location. This world model follows the same structure as in our previous work [38], and can thus be used for more advanced object representations, as long as they are valid parameters for the skills.

The skills themselves are implemented as classes, and each skill inherits from a parent skill class, in order to conform to the same class layout. The parent class contains placeholder methods for parameter-based execution, as well as evaluating all pre and postconditions in the skill. When implementing a skill, the execution method is overridden with a method that executes the exact skill, based on the parameters. The execution must, as previously stated, include both sensing and action. Methods for checking pre- or post-conditions are also added to the specific skill. All skill primitives are implemented in a single robot-specific class, since we are conducting these experiments on a single robot. This class is instantiated in the skill when robot motion or sensing is needed. Besides functions for e.g. moving the robot arm and detecting objects it also includes methods for navigating to a certain location that is previously saved in the world state.

4.3 Human Tracking and Gesture Recognition

As the fundamental engine for 3D body tracking we have used NiTE 2.2. It makes use of the OpenNI 2.2 SDK for accessing 3D sensor hardware, in our experiments the Asus Xtion sensor mounted on a pole on the robot. We choose this combination, since it has

previously proven useful for prototyping robotic applications concerning person tracking, and is directly available without the need for the end user to implement sophisticated tracking algorithms. We have previously evaluated the performance of the tracking, and found it sufficiently accurate for these purposes [20].

Since the NiTE tracker by default is tracking all detected persons in the camera view, we need to disable the tracking of persons that are not currently instructing the robot to avoid false positives in the gesture recognition process. Therefore, we first recognize a specific *attention* gesture (Fig. 3a) that specifies which person is currently the instructor. Only the gestures performed by the instructor are published to the teaching controller, and other people are ignored until they perform the *attention* gesture.

In total, we consider the following 5 gestures for programming, shown in Fig. 3: *attention*, *pointing*, *follow*, *stop*, and *abort*. The exact meaning of each gesture depends on the current state in the programming process, and will be elaborated in Section 4.5.

The gestures are recognized based on thresholding features of the instructors body pose. The instructor needs to maintain the pose for a certain period of time, in order to avoid false positives. The thresholding is performed for all poses at the same time. However, in the case of two pose thresholds being satisfied at the same time, only one of the poses are registered, as explained below. All gestures are simple events, apart from the pointing gesture, where a pointing direction should also be recognized. For these experiments we use the direction of the forearm as the pointing direction. For additional details of the gesture recognition we refer to our previous work [37].

4.4 Visual Feedback and User Interfaces

For both visual feedback and teaching through pointing, we have implemented two graphical user interfaces. One is related to the feedback from the tracker and gesture recognition, and is visible on a monitor attached to the robot chassis. The other is the main GUI, which is used for specifying and teaching tasks, and is shown on the touchpad.

The purpose of the tracker GUI is to show which person currently has the focus, and to show the progress of gesture recognition. This interface is particularly useful during programming, as well as in learning how to perform the 5 different gestures, as the user can see what is currently being recognized.

The main interaction with the robot during programming is done in the GUI on the touchpad. The GUI features a tab-based layout, where the two prominent tabs are for task *sequencing* and *teaching*, and is shown in Fig. 4. The third tab is for direct robot control, and is usually not meant to be available to the end user.

In the sequencing tab, the instructor can construct a task by concatenating the, at this point unparameterized, skills from the available skills on the robot. The user can drag and drop the skills from the skill library to the task specification, and reorder them if necessary. Both the skill library and the current task specification is visualized as lists, where the task specification is ordered and the skill library unordered. Using this GUI, it is therefore only possible to construct tasks that are sequences of skills, i.e. sequential robot programs. Therefore, it is presently not possible for the end user to specify conditional or loop constructs. However, in many cases the sequential program is sufficient, and we will later present a method for including skill combinations, that loop through certain parameters.

The user can proceed to the teaching phase in two ways; either the parameter for a single skill can be specified immediately when adding the skill, or the parameters for the entire sequence can be specified after the full sequence has been specified. In both cases, the parameter specification takes place in the teaching tab.

The teaching tab (see Fig. 4) both serves to initialize the teaching operation, but also to provide feedback on the current status of the teaching. On the left hand side, the skill sequence is listed, with

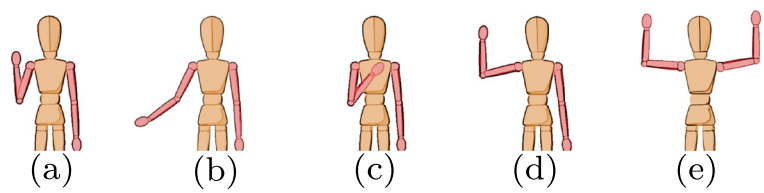


Fig. 3 The gestures available for programming: **a** *attention*, **b** *pointing*, **c** *follow*, **d** *stop* and **e** *abort*

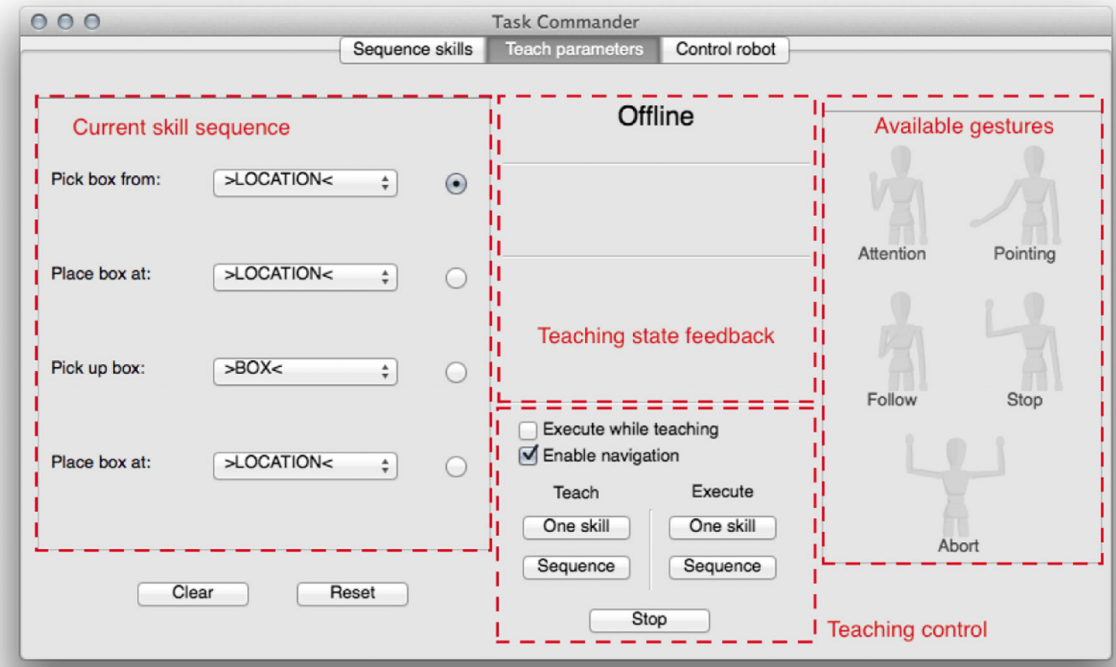


Fig. 4 The user interface for task teaching. In the image the teaching feedback tab is shown, and functional areas in the GUI are highlighted in red

drop-down boxes for the parameters. It is possible to specify the parameters directly using these boxes. However, in a real industrial scenario the objects and locations would most likely have unique names or IDs that are not meaningful to an operator, which is why these boxes are not used for our experiments. In the bottom middle part of the tab, the user can initialize the teaching of a single skill, or the whole sequence, as well as executing a previously parameterized skill or sequence. It is also possible to instruct the robot to execute each skill immediately after teaching, in order to verify correct execution immediately.

In the upper middle part of the teaching tab, the user is given written feedback on the teaching operation, supplied by the teaching controller described in Section 4.5. This could be e.g. that a pointing gesture is expected, or that the robot is currently following the operator. The teaching controller also specifies which gestures are currently valid, which is shown on the right hand side of the tab, where the valid gestures are highlighted.

4.5 Teaching Controller

The central teaching controller handles execution and teaching of tasks. In teaching, the program is supplied the sequence of skill types with missing parameters from the touchpad GUI. The program then establishes connection to the tracker, and enters a finite state machine, where the recognized gesture and current state determines how the system should react, and when to save the user input. The feedback in the touchpad GUI is supplied based on the current state. Upon task execution, the complete sequence of skills, including parameters, is supplied to the controller, which then executes the skills accordingly, one by one. A State Transition Table of the teaching phase is shown in Table 1.

If the instructor performs the *follow* gesture, the robot will follow him until told to stop, i.e. while in the Following state. This functionality has been implemented as a PD controller, which outputs translational

Table 1 State transition table for the teaching controller

	Waiting	Watching	Following
Attention	Watching	–	–
Pointing	–	Waiting[1]	–
Follow	Following	–	–
Stop	Watching[2]	Waiting	Waiting
Abort	**end**	–	–

The system starts in the Waiting state. The entries in the table specify the next state, given the current state (top row) and gesture input (left hand column). Notes [1] and [2] are explained in the text

and rotational velocity commands to the mobile platform, based on the location of the instructor in the frame of the tracking camera.

The parameter for a skill is computed as soon as the robot recognizes a *pointing* gesture, while in the Watching state (note [1] in Table 1). The transition to the Waiting state only occurs after the parameter is computed. When a *pointing* gesture is recognized, the current scene is scanned for objects or locations (marked by QR codes), and the distance to the pointing vector is calculated for all detections. The distance D between a QR code's 3D position q and the normalized pointing vector of the forearm d is given by

$$D = |(p_h^r - q) - ((p_h^r - q) \cdot \mathbf{d}) \cdot \mathbf{d}| \quad (1)$$

where p_h^r is the position of the hand. The exact chosen parameter is given by the QR code closest to the pointing vector. Should the user wish to cancel a chosen parameter, and redo the pointing for one skill, he/she can use the *stop* gesture to cancel the input and redo the pointing (note [2] in Table 1). The *stop* gesture is only available for canceling the input in the Waiting state immediately after a parameter has been detected.

Upon teaching the last skill in the sequence, the teaching phase is terminated by the *abort* gesture, and the interface returns to the teaching tab, where the programmed parameters for each skill are shown in the task specification. The user can now perform a test run of the skill, either by executing the whole task or one skill at a time.

5 Experiments

Preliminary experiments in the production facility at Grundfos proved that the approach had merit, i.e. that it was easy to use for factory workers and only a brief introduction was necessary [34]. For a more systematic evaluation we have verified our approach through two sets of experiments, with a total of 24 people. In both cases, the participants were asked to program a specific task, i.e. the use case of replacing boxes at an assembly station. This is a sample use case, chosen since it represents the kind of tasks that are difficult and not cost efficient to automate in the factories today, and thus serves to verify the approach. Other, unrelated task could be programmed in the same manner, given a set of skills that are appropriate for the task at hand.

All participants went through the same steps of

1. receiving a 10 min introduction to the system,
2. programming the sequence of skills once,
3. using the gestures to parameterize the skills 3 times, and
4. answered a survey on the intuitiveness and functionality of the approach.

The first experiment has been conducted in two different lab environments, and the second in the industrial environment at KUKA Roboter in Augsburg, Germany. The exact scenario and set of available skills differ slightly between the two experiments. In this paper we present additional results compared to our previous work [37], as well as qualitative feedback from all participants.

5.1 Experiment scenarios

In the laboratory experiment, the users were asked to replace a single, full box at the assembly station with an empty one. Thus, the robot should be instructed to pick up a box from the station, place it in a warehouse, pick up an empty box from the warehouse and place it back at the assembly station. The teaching thus includes not only pointing at objects or locations, but also maneuvering the robot to the two distinct locations using the follow-functionality. For this experiment, the robot was equipped with the skills `Pick up <object>`, `Place at <location>`, and `Pick`

`from <location>`. The first two skills relate to picking up a specific object or placing a currently grasped object at a specific location. The third skill is an abstraction of the `Pick up <object>` skill, as it instead relates to picking up an object at a specific location, regardless of which object it is. All skills include navigating the mobile platform near the goal object or location (as acquired from the world model), since this is relevant in nearly all logistic tasks, where transportation is usually required. Furthermore, the skills include detecting objects and locations in the scene, and manipulating the environment based on the current location of the desired object or location. Assuming the empty box is known to be empty, and it is this exact box that should be placed back at the assembly station, the correct task description for this experiment is:

1. `Pick from <AssemblyStation>`
2. `Place on <Shelf>`
3. `Pick up <EmptyBox>`
4. `Place on <AssemblyStation>`

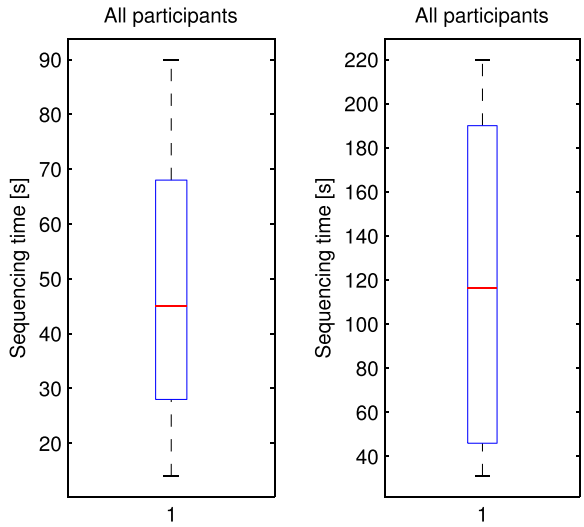
In the industrial scenario, we assume that several full boxes can be located at the assembly station, to investigate how users perceive more advanced skills that include several parameters. The goal is to replace all full boxes with empty ones. The full and empty boxes are stored at two different locations. For this experiment the robot is equipped with one additional skill compared to the lab experiment: `Pick all from <location> and place at <location>`. This skill is a *composite skill*, combining the `Pick from <location>` and `Place at <location>` skills, and thus requires two parameters, which are the locations to pick from and place on. The composite skill act as a loop, and thus performs these two skills as long as there are objects on the picking location and free space for an object at the placing location. Since the scenario requires all the full boxes to be replaced, regardless of the number of boxes, the composite skill is the right choice for both moving full boxes to one warehouse and moving empty ones to the assembly station. Thus, the correct task description becomes:

1. `Pick all from <AssemblyStation> and place at <FullBoxWarehouse>`
2. `Pick all from <EmptyBoxWarehouse> and place at <AssemblyStation>`

The lab experiment was conducted with 17 people, in two different environments. The participants were dispersed with respect to age, gender, height and previous experience with robot systems. Two environments were used to determine the effect of external disturbances on the tracker. The industrial experiment was carried out with 7 people, in a single environment. These participants were mainly male and had some experience with robot systems. The purpose of the lab experiment was to systematically assess the effects of age, gender, height and robot experience, while excluding potential influences from a factory environment. The second experiment was carried out at KUKA, in an industrial environment resembling the final use case, and additionally examined the use of more advanced skills.

5.2 Results

Participants generally found the *sequencing* step intuitive, and programmed the correct sequence fast. The times used for sequencing in both experiments are shown in Fig. 5, and also reveal that sequencing the skills was quite intuitive. However, the exact choice of skills to use was quite different across the participants. Even though the outcome of the two distinct skills that pick up an object is quite different (`Pick up <object>` vs `Pick from <Location>`), there



(a) Lab scenario (b) Industrial scenario

Fig. 5 Box charts showing sequencing time for the two scenarios

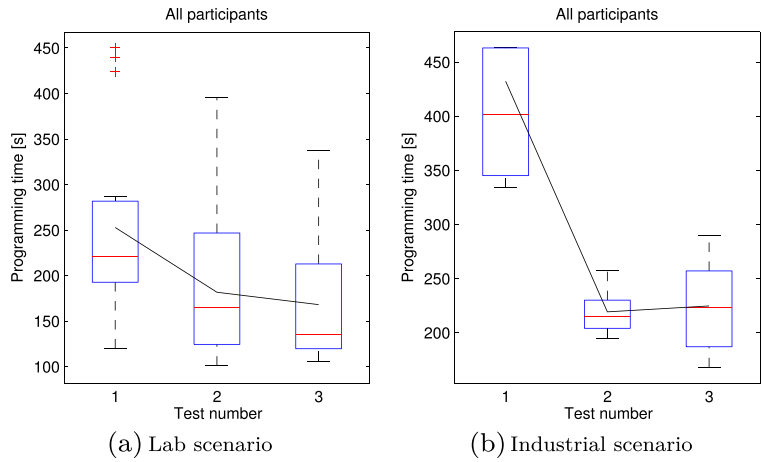
was no general tendency to use one skill instead of the other. When asked directly to program a different scenario of picking specific boxes from a shelf and placing them at various locations, there was still an even distribution in the choice of skills. From this it is clear that there is some confusion regarding the choice of exactly which skill to use in a certain situation.

For the industrial scenario, 5 participants used the correct skills for the sequence (the composite skill), and the remaining 2 participants used single skills for picking and placing. As this would result in unwanted behavior of the robot, and in order to compare results for teaching, the correct sequence was used for the teaching experiment. Even though only two skills had to be specified for the industrial scenario, it generally took much longer time than for the lab experiment (see Fig. 5b). For both experiments the times the participants needed for sequencing are somewhat divided into two blocks, and are thus not evenly distributed. This is mostly true for the industrial experiment, where 4 participant specified the sequence within $1min$, while none of the remaining 3 participants did it in less than $3min$. This is most likely due to the fact that some participants asked for minor assistance during the sequencing step, which suggests this step is not perfectly clear to everyone.

For the *teaching* part of the experiment, we have previously investigated the impact of age, environment, gender, height and previous experience with robot systems in the lab [37], suggesting no major difference in teaching times concerning these factors. The teaching times for all participants in the lab experiment are shown in Fig. 6a. We observe a general improvement in teaching time, both with respect to the mean time and the variance. This suggests that the approach is in fact quite intuitive, as participants generally learned how to use the gestures effectively within the 3 attempts, and even on the first attempt specified the task parameters within minutes.

Although the lab and industrial experiment differ, the participants still had to teach 4 parameters using gestures. Furthermore, the environment mostly resembles the open environment in the lab experiment, and none of the participants were taller than $1.8m$. This suggests the results from the industrial experiment generally should be similar, or even slightly faster, than for the lab experiment. However, for the industrial scenario, the participants had to maneuver the robot to 3 different locations, one more than for the lab experiment. As it was apparent during the experiment, maneuvering the robot is the operation that takes the longest during teaching, and this seems a contributing factor when comparing the teaching times across experiments in Fig. 6. Even though the variance is low for all attempts in the industrial environment, the mean is higher than for the lab experiment. For the 2^{nd} and 3^{rd} attempt, the teaching is at least $1min$ slower than the lab experiment. Interestingly, the participants performed better and more consistent on the 2^{nd} attempt. However, given the limited number of participants in this particular experiment, it is hard to determine the cause of this.

Across all participants, there was one participant whom the system could not track. Even when trying to restart the gesture recognition software and tracker, the tracker could not get an accurate tracking of the participant, as it continuously detected that the participant was turning around, even though she



(a) Lab scenario (b) Industrial scenario

Fig. 6 Box charts showing teaching times for all participants in the two scenarios, for 3 attempts. The black curve shows the mean teaching time

was standing still. The cause of this error is still unknown, and has not been observed before or since the experiment.

For a final qualitative analysis of our approach, we asked all participant to assess the intuitiveness and functionality, i.e. how well the system performed. Each participant rated the intuitiveness of the sequencing and teaching steps, and the overall approach, as well as the functionality of the touchpad GUI, the tracking and gesture recognition and the overall system. Each factor was rated on a scale from 1 to 5, where 1 was the lowest score, i.e. not intuitive or non-functioning, and 5 the highest, i.e. completely intuitive or functioning flawless. The scores are shown in Table 2.

Generally, participants found the sequencing very intuitive, and to some extent also the unified approach of skills and gesture recognition. However, the teaching step was not perceived as just as intuitive as the sequencing step. Comments from the participants generally concerned the clearness of the visual feedback and the choice of gestures. Some participants noted that it was hard to keep an eye on the feedback on both the monitor on the robot and on the touchpad, and the feedback from the gesture recognition could be more clear. It was also difficult for some participants to distinguish between the *stop* and *attention gesture*, as they are somewhat similar.

The participants had no comments on the functionality of the touchpad interface, and generally gave it good scores. However, the tracking and gesture recognition received mediocre scores, with the tracker receiving the lowest scores. In fact, we observed that most problems during teaching can be related to poor tracking of the instructor, resulting in poor gesture recognition, which suggest that this should be the main focus point for deploying a system like this. All comments regarding the gesture recognition were related to the *follow* gesture, as this was very hard to recognize for some participants. This is also related to the tracker, since the tracking of the arm would in some cases be lost, when the arm was moved in front of the torso, as required by the *follow* gesture. However, the participants gave good scores to the functionality of the combined system.

6 Discussion

The experiments show that the approach described in this paper is in fact intuitive, as all participants programmed the tasks within minutes, even on the first attempt, and even without any previous experience with robots. Furthermore, both the variance and mean of the teaching time is significantly reduced after 3 attempts of programming the robot. The fastest teaching time was 1 min 41 s for the lab experiment, and 2 min 48 s for the industrial experiment. There were some issues related to the tracking of the instructor, due to the orientation of the camera and noisy tracking in small environments.

Overall, the approach of skills was intuitive for all users, as the outcome of the skill is easily understandable. General feedback from the participants was also positive, especially with regard to the approach. However, feedback on the performance of the tracker, and when to use certain gestures was not so positive. The gestures were generally easy to use, but some attention will have to be paid to making the use of gestures more intuitive, and especially the tracking more robust.

Table 2 User feedback on the presented approach

	Intuitiveness of			Functionality of			
	Sequencing	Teaching	Approach	GUI	Tracker	Gesture rec.	System
1	–	–	–	–	–	–	–
2	–	–	–	–	17.4 %	8.7 %	–
3	4.2 %	33.3 %	12.5 %	17.4 %	*47.8 %*	*43.5 %*	13.0 %
4	*66.7 %*	*54.2 %*	*66.7 %*	34.8 %	26.1 %	39.1 %	*78.3 %*
5	29.2 %	12.5 %	20.8 %	*47.8 %*	8.7 %	8.7 %	8.7 %

Each parameter was rated on a scale from 1 to 5, where 1 is the lowest score, and 5 the highest. Entries indicate the percentage of participants who rated each parameter with a given score, and the highest percentages are highlighted in italics

Although these experiments have shown a simple skill implementation including world knowledge, the approach is easily scalable to new scenarios. The scalability is dependent on the types of objects the skills and robot system can handle. If truly general skills, that can manipulate all objects in the factory, are implemented, the same approach can be used to program logistic tasks involving these objects. And as long as the system can recognize the parameters for the skills (which is a requirement for the skills to function in the first place), the pointing gestures are still valid for parameter specification, even when several parameters are needed for each skill.

Compared to traditional industrial robot programming, this approach seems both faster and more general. Even if all objects and locations were known with absolute certainty, programming the task using traditional programming methods would still be significantly slower, and require expert knowledge. Initially, it was our goal to compare the programming time using the method outlined above to traditional robot programming. However, given *a)* the fast programming of the task by the participants using our approach, *b)* the complexity of the robot system, and *c)* the expert knowledge required to program tasks like this using e.g. proprietary teach pendants, this comparison would have little merit.

Using visual tracking of the instructor during teaching is not the only method for supplying parameters to the skill sequence. If the objects to manipulate are previously detected, the parameters could simply be specified directly in the GUI, as explained in Section 4.4. However, due to the size and dynamic nature of factory layouts, it would not be feasible to have the robot update this information autonomously on a regular basis. In our method, only the objects relevant for the particular task being programmed are saved during programming time, and updated during execution time. In this way, the robot can be programmed to e.g. place objects at locations that are not known a priori, where the world model is populated during teaching.

It should be noted that if the system is deployed in a real industrial setting, the training of factory workers to use the system would be much more elaborate than just a 10 min introduction. As experiments already show significant improvement for just 3 attempts at programming the robot, given this short introduction, the workers would in a short time be very proficient at programming the robot.

7 Conclusion and Future Work

In this paper we have presented a simplified Teaching by Demonstration approach for industrial mobile manipulators. Instead of recognizing human action, we supply the robot with a set of skills a priori, which the instructor uses to manually specify a skill sequence, or task. The robot then only recognizes the parameter for each skill in the task, through gesture recognition, where the exact parameter is what the instructor points at.

The advantages of this approach is that it is highly practical, as it allows the robot to deal with objects that are not at predefined locations. The system is also intuitive for most people, regardless of previous experience with robotics. While most current research in Teaching by Demonstration still lacks proper end user integration, the proposed method is comparably straight forward to implement and use.

The approach has been extensively verified through experiments in three different environments, including an industrial environment, with a total of 24 participants. The participants were dispersed with respect to age, gender, height and previous exposure to robot systems. All participants programmed a task by specifying the sequence and parameters within minutes, even on the first attempt, and only given a 10 min introduction to the system and approach. Generally, participants found the approach intuitive, but there is still room for improvement, especially regarding the functionality of the tracking and choice of gestures.

The reader may argue that the programming is so intuitive because the tasks and skills considered here are simple. However, there are many handling tasks in the industry, similar to the tasks discussed in this paper, which are not automized. The reason is exactly that even state of the art programming techniques do not allow to program these tasks on the fly. Since the skills unify sensing and action, they can deal with objects in a dynamic environment, which human-populated factory halls inherently are, and this is where usual industrial robot programming approaches fail.

Our next steps are mainly to expand the functionality of the skills, and make them more general. If the skills can handle a higher variety of objects, the tasks can also become more advanced, and a bigger variety – and thus higher number – of tasks can be programmed using this approach. We will also continue to improve the HRI aspect of the approach, by incorporating new methods for both sequencing and parameter specification, as well as improving the ones presented in this paper.

Acknowledgments The authors would like to thank Grundfos and KUKA for the possibility of doing experiments in the factories. We would also like to thank the volunteers for experiments. This work has partly been supported by the European Commission under grant agreement number FP7-260026-TAPAS.

References

1. Factory-in-a-Day project (2014). http://www.factory-in-a-day.eu/
2. STAMINA project (2014). http://stamina-robot.eu
3. TAPAS project (2014). http://tapas-project.eu
4. Archibald, C.C.: A computational model for skills-oriented robot programming. PhD thesis, University of Ottawa, Ottawa (1995)
5. Bøgh, S., Nielsen, O.S., Pedersen, M.R., Krüger, V., Madsen, O.: Does your robot have skills? In: Proceedings of the 43rd International Symposium on Robotics (ISR), Taipei (2012)
6. Bischoff, R., Kazi, A.: Perspectives on augmented reality based human-robot interaction with industrial robots. In: Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on, vol. 4, pp. 3226–3231 (2004)
7. Björkelund, A., Edstrom, L., Haage, M., Malec, J., Nilsson, K., Nugues, P., Robertz, S., Storkle, D., Blomdell, A., Johansson, R., Linderoth, M., Nilsson, A., Robertsson, A., Stolt, A., Bruyninckx, H.: On the integration of skilled robot motions for productivity in manufacturing. In: 2011 IEEE International Symposium on Assembly and Manufacturing (ISAM). doi:10.1109/ISAM.2011.5942366 (2011)
8. Bobick, A.: On human action. In: Moeslund, T.B., Hilton, A., Krüger, V., Sigal, L. (eds.) Visual analysis of humans, pp. 279–288. Springer, London (2011)
9. Bobick, A.F.: Movement, activity and action: the role of knowledge in the perception of motion. Phil. Trans. R. Soc. B Biol. Sci. **352**(1358), 1257–1265 (1997). PMID: 9304692 PMCID: 1692010
10. Bruyninckx, H., De Schutter, J.: Specification of force-controlled actions in the "task frame formalism"-a synthesis. IEEE Trans. Robot. Autom. **12**(4), 581–589 (1996). doi:10.1109/70.508440
11. Chen, H., Sheng, W.: Transformative CAD based industrial robot program generation. Robot. Comput. Integr. Manuf. **27**(5), 942–948 (2011). doi:10.1016/j.rcim.2011.03.006
12. Ekvall, S., Kragic, D.: Robot learning from demonstration: a task-level planning approach. Int. J. Adv. Robot. Syst. **5**(3), 223–234 (2008)
13. Finkemeyer, B., Kröger, T., Wahl, F.M.: Executing assembly tasks specified by manipulation primitive nets. Adv. Robot. **19**(5), 591–611 (2005)
14. Gadensgaard, D., Bourne, D.: Human/Robot multiinitiative setups for assembly cells. In: ICAS 2011, The Seventh International Conference on Autonomic and Autonomous Systems, pp. 1–6 (2011)
15. Galindo, C., Fernández-Madrigal, J.A., Gonzílez, J., Saffiotti, A.: Robot task planning using semantic maps. Robot. Auton. Syst. **56**(11), 955–966 (2008). doi:10.1016/j.robot.2008.08.007
16. Gottschlich, S., Ramos, C., Lyons, D.: Assembly and task planning: a taxonomy. IEEE Robot. Autom. Mag. **1**(3), 4–12 (1994)
17. Hartmann, M.: DYNAPRO: Erfolgreich produzieren in turbulentenMärkten. Logis Verlag (1996)
18. Hartmann, M.: DYNAPRO II: erfolgreich produzieren in turbulenten Märkten. Logis Verlag (1997)
19. Hartmann, M.: DYNAPRO III: Erfolgreich produzieren in turbulenten Märkten. Logis Verlag (1998)
20. Høilund, C., Krüger, V., Moeslund, T.: Evaluation of human body tracking system for gesture-based programming of industrial robots. In: Proceedings of the 2012 7th IEEE Conference on Industrial Electronics and Applications (ICIEA), pp. 477–480. IEEE (2012)
21. Huckaby, J., Vassos, S., Christensen, H.I.: Planning with a task modeling framework in manufacturing robotics. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5787–5794 (2013)
22. Hvilshøj, M., Bøgh, S., Nielsen, O.S., Madsen, O.: Autonomous industrial mobile manipulation (AIMM): past, present and future. Ind. Robot. Int. J. **39**(2), 120–135 (2012). doi:10.1108/01439911211201582
23. Inamura, T., Toshima, I., Tanie, H.: Embodied symbol emergence based on mimesis theory, vol. 23, pp. 4–5 (2004). http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.100.7250
24. Jen, Y.H., Taha, Z., Vui, L.J.: VR-Based robot programming and simulation system for an industrial robot. Int. J. Ind. Eng. Theory Appl. Pract. **15**(3), 314–322 (2008)
25. Kang, S.B., Ikeuchi, K.: Toward automatic robot instruction from perception-temporal segmentation of tasks from human hand motion. IEEE Trans. Robot. Autom. **11**(5), 670–681 (1995). doi:10.1109/70.466599
26. Kang, S.B., Ikeuchi, K.: Toward automatic robot instruction from perception-mapping human grasps to manipulator grasps. IEEE Trans. Robot. Autom. **13**(1), 81–95 (1997). doi:10.1109/70.554349
27. Krüger, N., Piater, J., Wörgötter, F., Geib, C., Petrick, R., Steedman, M., Ude, A., Asfour, T., Kraft, D., Omrcen, D., et al.: A formal definition of object-action complexes and examples at different levels of the processing hierarchy (2009). Technical report. http://www.paco-plus.org
28. Kröger, T., Finkemeyer, B., Wahl, F.: Manipulation primitives — a universal interface between sensor-based motion control and robot programming. In: Schütz, D., Wahl, F. (eds.) Robotic systems for handling and assembly,

Springer Tracts in Advanced Robotics, vol. 67, pp. 293–313. Springer, Berlin (2011)
29. Kröger, T., Finkemeyer, B., Winkelbach, S., Eble, L.O., Molkenstruck, S., Wahl, F.: A manipulator plays jenga. IEEE Robot. Autom. Mag. **15**(3), 79–84 (2008). doi:10.1109/MRA.2008.921547
30. Krüger, V., Herzog, D., Baby, S., Ude, A., Kragic, D.: Learning actions from observations. IEEE Robot. Autom. Mag. **17**(2), 30–43 (2010). doi:10.1109/MRA.2010.936961
31. Krüger, V., Kragic, D., Ude, A., Geib, C.: The meaning of action: a review on action recognition and mapping. Adv. Robot. **21**(13), 1473–1501 (2007)
32. Kulic, D., Kragic, D.: Learning action primitives. In: Moeslund, T.B., Hilton, A., Krüger V., Sigal, L. (eds.) Visual analysis of humans, pp. 333–354. Springer, London (2011)
33. Lopes, M., Santos-Victor, J.: A developmental roadmap for learning by imitation in robots. IEEE Trans. Syst. Man Cybern. B (Cybernetics) **37**, 308–321 (2007). doi:10.1109/TSMCB.2006.886949
34. Madsen, O., Bøgh, S., Schou, C., Andersen, R., Damgaard, J., Pedersen, M.R., Krüger, V.: Integration of two autonomous mobile manipulators in a real-world industrial setting. In: IROS Workshop on Robotic Assistance Technologies in Industrial Settings (RATIS). Tokyo (2013)
35. Mitsi, S., Bouzakis, K.D., Mansour, G., Sagris, D., Maliaris, G.: Off-line programming of an industrial robot for manufacturing. Int. J. Adv. Manuf. Technol. **26**(3), 262–267 (2005). doi:10.1007/s00170-003-1728-5
36. Neto, P., Mendes, N.: Direct off-line robot programming via a common CAD package. Robot. Auton. Syst. **61**(8), 896–910 (2013). doi:10.1016/j.robot.2013.02.005
37. Pedersen, M.R., Herzog, D., Krüger, V.: Intuitive skill-level programming of industrial handling tasks on a mobile manipulator. 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2014)
38. Pedersen, M.R., Nalpantidis, L., Bobick, A., Krüger, V.: On the integration of hardware-abstracted robot skills for use in industrial scenarios. In: 2nd International IROS workshop on cognitive robotics systems: replicating human actions and activities. Tokyo (2013). http://renaud-detry.net/events/crs2013/papers/Pedersen.pdf
39. Takashi, I.A.: Towards an assembly plan from observation: task recognition with polyhedral objects. IEEE Trans. Robot. Autom. **10**(3), 368–385 (1994). http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.12.8697
40. Tenorth, M., Perzylo, A., Lafrenz, R., Beetz, M.: Representation and exchange of knowledge about actions, objects, and environments in the RoboEarth framework. IEEE Trans. Autom. Sci. Eng. **10**(3), 643–651 (2013). doi:10.1109/TASE.2013.2244883
41. Waibel, M., Beetz, M., Civera, J., D'Andrea, R., Elfring, J., Galvez-Lopez, D., Haussermann, K., Janssen, R., Montiel, J.M.M., Perzylo, A., Schiessle, B., Tenorth, M., Zweigle, O., van de Molengraft, R.: RoboEarth. IEEE Robot. Autom. Mag. **18**(2), 69–82 (2011). doi:10.1109/MRA.2011.941632
42. Yeasin, M., Chaudhuri, S.: Toward automatic robot programming: learning human skill from visual data. IEEE Trans. Syst. Man Cybern. B Cybern. **30**(1), 180–185 (2000)