# Intuitive Robot Programming




**Charly Blanc, Julius Jankowski, Andreas Sonderegger, Sylvain Calinon, and Sarah Dégallier Rochat**

**Abstract** The introduction of robots in SMEs requires a drastic reconsideration of existing programming techniques and interfaces. The growing versatility of the market essentially demands the robots to be re-programmed frequently, calling for interfaces that are intuitive enough to require no robotic expertise, yet flexible enough to meet industrial needs. Rather than limiting the intervention of the workers, the technology should empower them to take control over the machine. Only then, the human-machine complementarity can be leveraged, combining the efficiency of automation with the flexibility of human work. Ergonomics and human factors are critical to the development of these new programming interfaces, which should typically be developed and tested in a cyclic fashion with the end-users. In this chapter, we highlight the differences between easy-to-learn and easy-to-use interfaces, as well as the differences between interfaces targeting naive/casual users and interfaces targeting expert/professional workers. We discuss the design of robot programming interfaces based on a user-centric perspective. The research developments in this project target different methods to collect demonstrations (including tracked devices and kinesthetic teaching), different graphical user interfaces (including block-based and timeline representations), and different approaches to augmented reality (including tablets/smartphones, headsets or actuated projection devices). Through this multi-modular approach, we aim at providing workers with the most natural way possible to interact with the robot depending on the task or their preferences.

**Keywords** Human-machine interfaces · Robotic programming · Learning from demonstration · Anthropocentric production systems

This work was supported by the Swiss National Science Foundation (SNSF) through the CODIMAN project.

C. Blanc · A. Sonderegger · S. D. Rochat
Bern University of Applied Sciences, Bern, Switzerland
e-mail: charly.blanc@bfh.ch

J. Jankowski · S. Calinon (✉)
Idiap Research Institute, Martigny, Switzerland
e-mail: sylvain.calinon@idiap.ch

Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland

© The Author(s) 2027

G. Barresi and S. Wischniewski (eds.), *Ergonomics in Robotics: Advances and Innovations*, Springer Tracts in Advanced Robotics 162,
https://doi.org/10.1007/978-3-032-19374-2_13

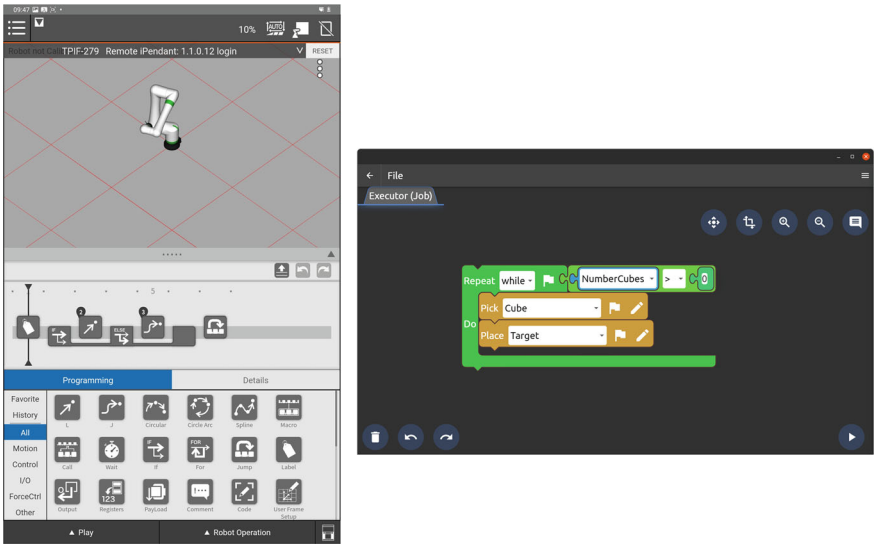
## 1 Introduction

New technologies such as artificial intelligence, augmented reality and robotics—and the associated 4th industrial revolution—are expected to drastically change the nature of work in the manufacturing industry [12, 26, 50]. Industrial production systems shall become leaner, but also more integrated, traceable, and adaptable. The past visions of fully computer-controlled factories where no worker is needed are now challenged by visions of agile companies where flexible production systems can be leveraged by human creativity to foster continuous improvement and innovation [25].

To cope with the versatility of the market and the rapid evolution of technology, agile automation solutions that can be rapidly adapted to new demands are needed. New technologies are seen as enablers for "anthropocentric production systems", that are systems that allow the close interaction of humans and machines on the shop floor through the enhancement of the worker's capabilities [48]. For this transformation to happen, a shift in paradigm in industrial interface design is required, like the democratization of computers 40 years ago: rather than expert systems, intuitable solutions that can leverage the worker's task expertise are needed [59]. Collaborative robots have opened the path to the democratization of automation: since they allow for physical interactions with the device, non-experts should be able to develop ad hoc solutions by interacting directly with the robot, without having to code. Two main modalities have been used to simplify the programming of these robots: visual programming interfaces and learning by demonstration.

Visual programming interfaces have been used since decades in the industry to simplify programming, as for instance the icon-based flow chart offered by systems such as MORPHA or the Engel Programming Language (EPL) [38]. Such visualization are also used by robot manufacturers such as FANUC (see Fig. 1a) and Franka Emika. Recently, a new approach to visual programming based on text blocks has been proposed with systems such as Blockly [27] or Scratch [47]. Each block represents a statement with open parameters and the blocks can be combined to create more complex programs. Several studies have shown the relevance of this approach in terms of efficiency and usability in pure programming tasks [62], but also when applied to robotic systems [61]. Companies such as ABB and Omron are now proposing robotic systems with block-based programming interfaces.

An important research effort was deployed recently to simplify the programming of robots by exploiting imitation mechanisms [8]. The goal is to transfer skills to robots by letting them learn new tasks from human demonstrations (learning from demonstration, LfD). Research in this field encompasses three aspects: (1) the development of models and algorithms that can extract relevant patterns from multiple demonstrations, in order to compress the data and keep only the essential components of the demonstrated skills; (2) the development of sensory integration capability to record the demonstrated skills; and (3) the development of intuitive interfaces to simplify the re-programming of robots and to collect multimodal sensory data characterizing the task to be transferred [19]. These interfaces can take various forms, including smartphones with augmented reality (AR) capability, virtual reality (VR)




(a) Fanuc CRX interface (b) Prograblock interface

**Fig. 1** Icon-based and block-based programming interfaces

headsets, teleoperation devices with force feedback, and robots endowed with torque sensing capability, allowing the user to guide the robot through the different steps of a task while the robot records movement information by proprioception (kinesthetic teaching).

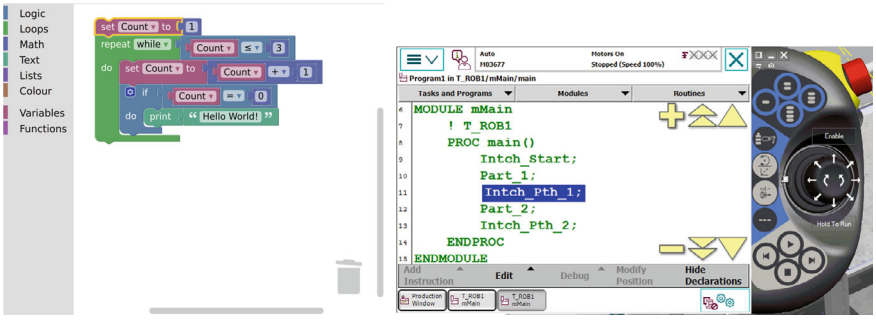
From the user point of view, these new approaches to robot programming aim to enable interaction modalities with the machine that relies much more on the understanding of the task to be performed than of the machine itself. Rather than requiring robotics experts, the system shall need task experts, that is, workers, to be efficiently put to use. In this chapter, we present implementations of a block-based programming (BBP) interface and learning from demonstration (LfD) for industrial applications.

## 2 Block-Based Programming Languages

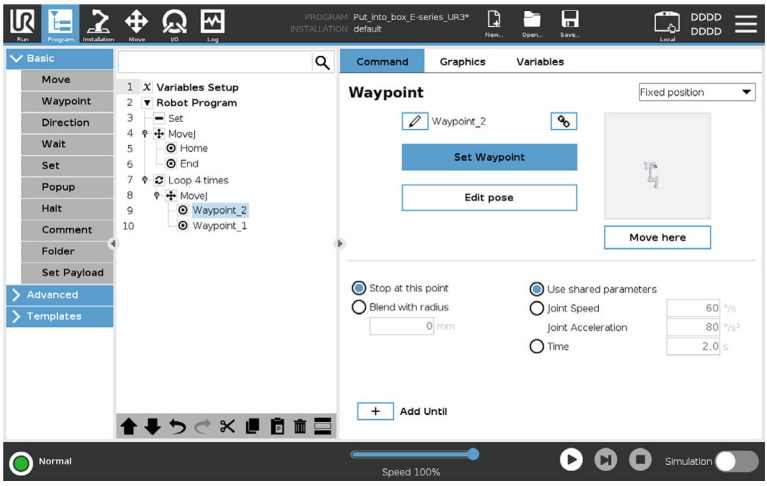
Block-based interfaces are visual programming tools that are based on blocks representing statements with open parameters. The blocks can be combined to create complex programs, as illustrated in Fig. 1b. This approach has been proven to efficiently ease programming by removing syntax-related obstacles [63] and by reducing cognitive load, but also because the user does not need to remember the language specific vocabulary since it is sufficient to recognize the correct block [6]. Previous work successfully combined Blockly with industrial robots [61] for a task where

adult novices had to program a pick-and-place task. Other studies investigated robot programming teaching for adult novices [60] using Blockly (see Fig. 2a) with a framework called CoBlox. They compared the block-based approach with two widely used industrial programming approaches, namely a traditional, text-based teach pendant (see Fig. 2b) and the tree-based, dialog-driven interface providing by the company Universal Robot (see Fig. 2c).

The block-based system obtained better scores in terms of usability, learnability, and overall satisfaction. The participants were also able to program the required tasks faster. In a recent study [38], no differences were observed in efficiency between the traditional icon-based system and block-based systems. However, some of the obstacles that were noticed with the block-based system used for the study, such as the quantity of blocks and the lack of higher level blocks, can be easily overcome. In particular, different levels of abstractions for the blocks can be defined, for instance,




(a) Blockly interface (b) ABB FlexPendant interface

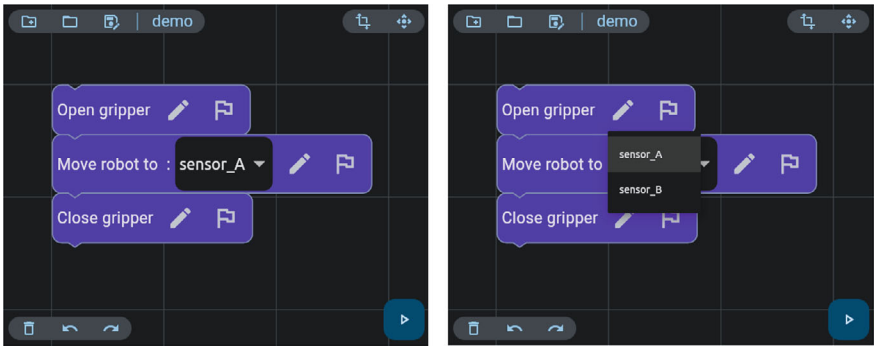



(c) Universal Robots interface

**Fig. 2** Different types of robot programming interfaces

through a three level architecture based on the concept of *skill* [49]. In this model, the higher level (*tasks*) corresponds to the abstract description of the task (such as "build motor"), while the lower level (*device primitives*) reflects the machine API (such as "move in Cartesian space to this position"). The intermediate level is based on the concept of *skills* and corresponds to the steps needed to achieve the task (such as "take rotor"). The higher level is extremely intuitive for the workers, with a limited choice of actions, similarly to the typical operator interfaces, while the lower level represents the complete flexibility offer by the interfaced device. The intermediate level is object-oriented and still meaningful for the workers, although some robotic and programming concepts (such as *frames* or *loops*) are needed for the worker to be able to use the system to its full potential. Since *tasks* are a combination of *skills* and *skills* a combination of *devices primitives*, the user can learn these concepts by imitation by simply adapting existing programs to their production needs, for instance if the working environment changes or a new type of pieces needs to be produced (see Fig. 3). In this illustration, the user has the capability to choose an object from the dropdown menu (see Fig. 3b) for the robot to grasp. An object is defined by a unique identifier enabling references in the program and a frame.

However, for the workers to develop a feeling of ownership over the machine, they need to be able to predict, to a certain extent, the behavior of the robot. For this, a minimal understanding of the system is required [30]. It is therefore important to teach the worker the needed expertise in programming and robotics to allow them to develop a sense of autonomy and control over the system. In the context of robotic automation, contrarily to most leisure applications, the users are typically willing to spend time and energy to learn about the system to be able to leverage its full potential. They acknowledge that a deeper understanding of the technology enables them to make the most of its capabilities, improving productivity and efficiency in various tasks through the utilization of these robotic systems. Tutorials were therefore developed for the users to learn the required competencies to achieve more complex tasks. They are based on typical industrial use cases and are meant to be done by the




(a) Example of the task *Pick* (b) Parameterization of the task *Pick*

**Fig. 3** Example of task reprogrammed for a new type of components

users when needed. Since interacting directly with the robot can be a source of stress if the user fears to get hurt by it or to break it [36], a simulation of the system was developed for the tutorials that can be used on a computer or tablet or with augmented reality (AR) goggles. In the following paragraphs, we describe more in details the different aspect of the interface.

## *2.1 A Flexible yet Intuitive Block-Based Programming Language: Prograblock*

*No-code programming*. A common obstacle with programming is the learning of the syntax [24]. We therefore designed a no-code, block-based interface, *Prograblock*, illustrated in Fig. 1b. The blocks are parameterizable and can be edited. New blocks can be created by combining existing blocks, allowing for a greater flexibility. A similar system can be used for controlling the robot behavior and the vision algorithms, but also to control other ancillary devices, such as a vibrating bowl or the machine that need to be tended. The different blocks are classified in three different categories corresponding to different levels of abstraction as explained in the next paragraph.

*Skill-based three-layer architecture*. The programming interface utilizes three different levels of abstraction, going from common and intuitive human language (tasks) to the flexible but complex machine API (device primitives), with an intermediate language based on skills. Skills are object-oriented and serve as a communication bridge: they offer an intermediate language that can be used to describe the process steps in relation to the object being handled [49]. Skills are sensorimotor behaviors (such as pick and place) and require the coupling of a sensory information (for instance, the detection of the position of a component) and an action (for instance, moving to the location to be reached). New tasks can be created by simply combining skills, and new skills by combining device primitives. The user can create a new task (or skill) by simply editing an existing one. The idea behind the three-level architecture is to encourage the user to gain control over the system by progressively learning the lower-level languages, while maintaining a feeling of safety and control.

*Tutorials*. Tutorials were also developed for the users to learn the needed competencies to switch between the levels of abstraction and complete more complex tasks. In the case of robot programming, different concepts need to be understood, such as, for instance, loops or conditions in programming, as well as robot kinematic singularities. The tutorials are thus based on typical industrial use cases, with a small description of the concepts that will be learned. In the Experiments section below, we present the six first tutorials that have been developed and tested with students. The tutorials can be done directly with the robots since the elements can be 3D-printed or with a simulator (see Fig. 4).

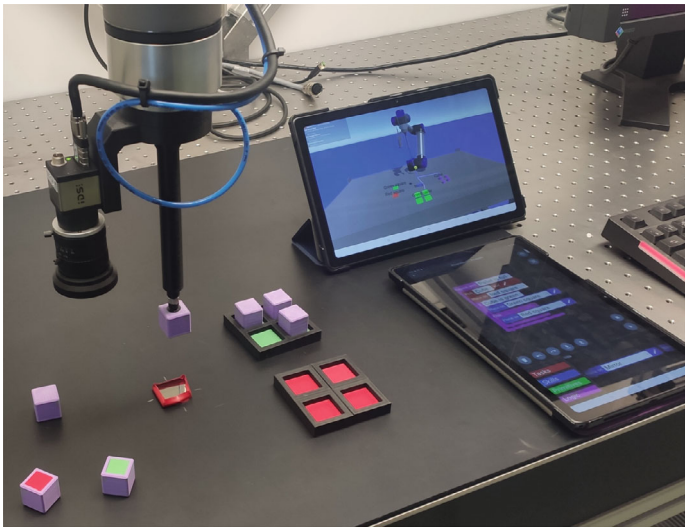


**Fig. 4** Experimental setup

*Simulator*. A simulation of the system was developed for the user to be able to check the programmed behavior before executing it on the robot, following the concept of digital twin (see Fig. 5). We use it for the tutorials since the environment is known and can be modelled in advance. In the cases where the physical set-up is used, the simulation is automatically adapted to match the real set-up. It means that, since the environment is flexible (the positions of the elements are not predefined), the position of the objects in the simulation will be adapted to match the real set-up based on the camera feedback. The system can be used on a tablet, a computer or with AR goggles.

## *2.2 Experiments*

Changes brought by the introduction of technology are usually accompanied by resistance and mistrust from the existing workforce [11]. This is especially true in the case of change linked with the implementation of technology that carries out one's own tasks and thus could pose a potential threat to one's own job [39]. Resistance in the workforce to the introduction of new technologies can have a negative impact on satisfaction, productivity and can jeopardize the success of the technology introduction [9, 53]. Therefore, it is important to make sure that the introduction of new technology will have a positive impact on the workers. To achieve this, we suggest adopting a user-centered approach, in which the development and introduction of the new technology is done in close collaboration with the people

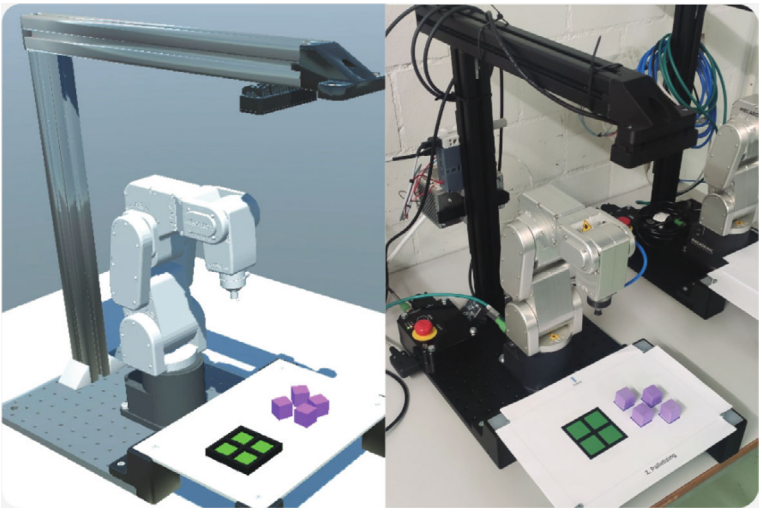


**Fig. 5** Digital twin (simulation on the left, real system on the right)

concerned. Observations and interviews with the target operators have therefore been performed with the workers and a series of workshops is planned in the future to test the interface with them and adapt it to their needs once the system is robust enough. A preliminary evaluation of the whole system was performed with students through 2 different studies, as described below.

**Description of the tutorials**. In a first attempt to test this block-based, skill-oriented interface, two different experiments were performed with students. In all the experiments, the students had to perform six different exercises under different conditions. The four first ones were guided tutorials, while the two last ones were for testing the acquired competencies. The scenarios for the exercises were based on industrial applications such as pick and place or palletizing. The exercises were done with a simulation of the robot. The goal of the exercises were to help the user better understand the different functionalities of the interface, as well as some programming concepts, such as loops and conditions. More details about the design of the tutorials can be found in our previously published study [2].

**Study 1: Effect of the Interface on Self-efficacy and Programming Anxiety**. We aim at helping lay people develop a feeling of control over the robot through human-centered design approaches as well as the development of tailor-made tutorials. In order to test whether this was achieved with the proposed solution, a first study involving 58 student participants aged 16 to 39 years, with varying programming experience was conducted. The measures included programming anxiety [22], programming self-efficacy [22], attitudes towards robots [42], robot anxiety [5], using

established questionnaires such as the Computer Programming Anxiety Questionnaire [22] and the Robot Attitude Questionnaire [42]. Assessments were performed before and after participants interacted with a robotic interface and also examined user experience [37] and task performance within a tutorial and test framework. We hypothesized a more positive attitude towards robots and a lower anxiety after interacting with the interface compared to the baseline measurement.

**Results**: Analysis of data revealed a significant decrease of **programming anxiety** after using the robot programming interface. Furthermore, participants reported significantly more positive **attitudes towards robots** after interacting with the programming tool compared to the baseline measurements. Participants' evaluation of their **robot anxiety** decreased also after using the robot programming tool, the effect did however not reach significance level. **Programming self-efficacy** did not change significantly after using the robot programming tool. In terms of user experience evaluation, most participants (79%) were able to complete the tasks during the 30 min given while only a minority of the participants (29%) completed the tasks without help. After completing the tasks, participants rated their user experience positively in terms of attractiveness, perspicuity, dependability, and stimulation. However, they gave lower ratings for efficiency and novelty.

**Discussion**: The purpose of our intuitive robotic interface is to make the programming of robots as easy as possible for laypeople, in such a way that their fears and reservations about the technology are reduced, but also increasing their positive attitudes toward robots and their deployment.

Considering the results of study 1, our objectives have been partially achieved: the initially rather high ratings of anxiety towards programming have been considerably reduced after interacting with the interface. Also, participants' ratings of their attitudes towards robots significantly increased after using the interface. In a similar direction, the evaluation of robot anxiety also decreased after using the interface. This indicates (small sample) that the use of intuitive and easy-to-use programming interfaces can positively influence the initially existing fears regarding programming and the attitude toward the use of robots. User experience evaluation unveiled that not all participants were capable to complete all the tasks, requesting to some extent help from the test facilitators. This indicates that, despite our efforts, the interface might not be intuitive enough yet and needs to be further improved so that everyone can build up self-efficacy and the necessary knowledge to program robots in the future.

Interestingly, however, the interaction with the interface did not positively influence participants' evaluation of their self-efficacy regarding their programming abilities. This could be related to the fact that the interaction with the tool lasted only a relatively short time (about 60 min), a duration which may not be sufficient to influence this important component of self-assessment of personal competences. This could be expected according to Yildiz et al. [65], who tested the effect of robot programming on computational thinking skills toward self-efficacy perception during seven weeks, two lessons per week. Their results showed a significant increase of self-efficacy.

Furthermore, a study by Hsu and Hwang [28] on a similar block-based interface found reduced learning anxiety among young students, aligning with our findings on programming and robot anxiety. Their study also indicates that block-based programming enhances playfulness and enjoyment in learning computational thinking.

Reich-Stiebert and Eyssel's research indicates positive attitudes toward robots in STEM (science, technology, engineering and mathematics) education contexts [46], consistent with our findings. Nomura et al. also found that personal robot experiences significantly influence these attitudes [43], suggesting that positive experiences with a block-based interface may enhance users' attitudes towards robots.

The implementation of an intuitive robotic interface in industrial settings offers several significant benefits. It can enhance workforce training by enabling production workers without prior programming knowledge to quickly learn and reprogram robots, reducing downtime and increasing production efficiency. Additionally, the interface can lower technological anxiety and improve attitudes toward robots, fostering greater acceptance and utilization of robotic technologies. This user-friendly approach also supports continuous learning and skill development, promoting innovation as workers conceive new ways to utilize robots. Moreover, by improving safety and ergonomics and enabling rapid adaptation to market changes, companies can gain competitively and better respond to evolving customer demands.

Despite the demonstrably positive effects of using the proposed interface, the user experience evaluation indicated that not all test participants managed to solve the test tasks independently after completing the tutorial. The analysis of their interactions with the interface as well as their feedback suggest that this might be related to reduced situational understanding linked with the use of the 3D simulation. A possible remediation would be to present a robot to the participants. Whether this actually influences the results was the subject of the second study conducted.

**Study 2: Influence of Experiencing a Real Robot vs. Mere Interaction with a 3D-Simulation** . In study 1, participants learned and tested their programming skills on a 3D-simulation of a robotic arm. The results of this study showed that not all participants were able to independently solve the tasks set after completing the tutorial. A possible reason may be due to the lack of situational understanding imposed by the mere use of a virtual environment. Seeing ones own programming efforts performed by a real robot might instead provide such situational understanding and hence improve programming performance. In order to address this research question, we conducted a second study to investigate the consequences of being introduced to a real robot on programming performances. We expected that participants attending such an introductory session would perform better in the programming tasks using the 3D-simulation compared to participants not attending the session.

This study involved 58 master students from a business school, aged between 23 to 46 years with no prior programming experience. Participants were divided into two groups to observe the effect of different introductory sessions on learning robotic programming. One group witnessed a live demonstration of a robotic cell, which included a collaborative robot with a gripper and a camera performing a sorting task,

emphasizing that the robot operates solely on pre-programmed instructions. The other group proceeded directly to the programming tutorial without this demonstration. Participants worked in pairs throughout the tutorial and subsequent test session, fostering knowledge sharing as they interacted with the application used in the study 1.

Performance indicators were assessed using task completion time, the number of trials, and the number of program modifications. Participant experience was evaluated with the System Usability Scale (SUS [10]) for interface usability and the Animated Visualization of Affective States (AniSAM [51]) for emotional states.

**Results**: No differences were observed between the two groups in the tutorial phase regarding time spent on the tutorial, total number of trials and total number of modifications. For the simple task in the subsequent test session, no differences were observed between the two experimental groups neither. However, for the complex task in the test session, significant differences were observed, with participants who received an introductory session completing the task faster and making a lower number of modifications compared to the group not receiving the introductory session.

Analysis of participants' affective state revealed higher levels of **arousal** for participants in the introductory session group compared to participants not receiving an introduction. Effect of time and the interaction (introductory session x time) did not reach significance level. With regard to **valence**, data analysis revealed that the effect of the experimental manipulation was not significant while the effect of time was significant. Pairwise comparisons however indicated no significant differences between the three measurement points. Furthermore, the interaction (introductory session x time) did not reach significance level. The usability assessment of the system resulted in a SUS-score [10] of 71.7 which can be considered to be between good and excellent in terms of adjective ratings [4].

**Discussion**: The key findings of study 2 relate to the impact of the experimental manipulation on performance data in the test tasks. Although there was no discernible effect of the introductory session on the initial, less challenging task, participants demonstrated improved performance in the more difficult second task when they engaged in the introductory session.

In the first test task, the objective was to stack various cubes on different pallets. In the second test task, the robot arm had to navigate around an obstacle while stacking cubes. Most participants successfully programmed the robot to avoid the obstacle during the pickup and placement phases using a waypoint. However, a noteworthy proportion encountered difficulties in realizing that this avoidance procedure was also necessary when the robot arm returned to the pickup area due to the repetitive nature of the pickup and placement actions inherent to the stacking task. The observed improvement in performance among those who participated in the introductory session might be attributed to a better comprehension of how robots operate. Robots, being machines, strictly follow a predefined list of actions programmed by humans and lack the ability to interpret contextual information. Viewing the robot as adhering to a set of instructions without contextual understanding led to the correct

resolution of the task. This suggests that attending the introductory session shapes participants' mental models (as described by Kieras [33]), enabling them to predict and explain the system's behavior.

In this second study, participants who attended the introductory session exhibited a higher level of arousal compared to the control group. While the exact cause of this effect remains speculative, it is conceivable that the heightened arousal is linked to witnessing a real robot arm in operation, potentially associated with increased stress. It is plausible that any negative consequences of this stressful experience were offset by the positive effects of gaining a deeper understanding of how robots function. In fact, Tisza et al. have found that arousal positively correlates with students' learning gains [57]. Further research is necessary to replicate this finding and ascertain its underlying cause.

Interpreting valence changes is difficult without significant post-hoc comparisons, suggesting further studies with increased statistical power are needed for definitive conclusions. The limitations of this study include its small sample size and paired programming, which might have distorted results due to group dynamics. Plans for a replication study with greater statistical power are underway to confirm these findings.

The results from study 2 indicate that an introductory session can significantly improve performance on complex tasks by helping participants understand how robots operate. This approach could be highly beneficial in industrial settings, allowing production workers with no prior programming experience to reprogram industrial tasks independently. This capability would enable quick adaptation to market changes. The introductory sessions help shape workers' mental models, making them better equipped to predict and explain robot behavior.

This method could be applied in industrial settings, empowering production workers with no prior programming experience to independently reprogram industrial tasks, thus facilitating quick adaptation to market changes. However, due to the unique characteristics of this target group, additional studies are required to assess feasibility. We are currently collaborating with a company to implement our system on their shop floor and plan to initiate trials with employees soon. The project aims to gradually improve the interface based on their feedback.

## 3 Learning from Sparse Human Demonstrations

Robot learning from demonstration (LfD) refers to the process used to transfer new skills to a robot by relying on demonstrations from a user [3, 8]. It is inspired by the imitation capability developed by humans and animals to acquire new skills. LfD aims at making programming accessible to novice users by providing them with an intuitive interface they are familiar with, as humans already exchange knowledge in this way. The goal is to provide user-friendly interfaces that do not require knowledge in computer programming or robotics. LfD can be considered at various levels, from

the transfer of low-level motor control to the transfer of high-level symbolic reasoning capabilities. For a given skill to be acquired, several learning strategies can be considered, from the copying of the demonstrated actions to more elaborated abstractions, such as the extraction of the underlying objectives of the actions [21, 45]. The terms behavioral cloning and inverse optimal control can be used to refer to these two broad learning strategies, respectively. They have connections with imitation mechanisms studied in ethology and developmental psychology, where one can distinguish action-level mimicry to goal-level emulation (see e.g. [64]).

LfD is motivated by many applications and can be applied to various robots. In an industrial context, it is motivated by the lowered barrier to entry the field and a democratized access to robotics. These changes require quick and cost-effective adaptations of existing assembly lines, as well as the handling of small volumes such as personalized products. It typically requires the sharing of a common workspace between the workers and the robots. In practice, LfD enables robots to be reprogrammed by the persons who know the tasks to achieve, but who do not necessary have expertise in robot programming. In this context, LfD removes the costly and timely step of soliciting external expertise each time the robots need to be reprogrammed.

In service robotics, LfD aims at providing personalized assistance and services that could not be preprogrammed in advance due to the broad variety of tasks, persons, or environments that a robot can encounter. In some of these applications, LfD can rely on interactive social cues and on the natural human propensity to teach news skills to others—a communication behavior we already use to transfer knowledge.

**Learning by Bilateral Interaction**

In addition to the skill transfer perspective, a key aspect of LfD is the exploitation of the social learning mechanisms behind LfD (see [41] for an overview). In a classical machine learning problem it is often assumed that expert datasets are available (typically, assuming a preliminary access to the samples that are relevant solutions to the problem). Similarly, the learning strategy to be used is also often explicitly specified. These developments do not account for the way in which data are collected. In contrast to many machine learning applications in which the learning systems are independent of the acquired data, a remarkable characteristic of LfD is that the iterative interaction with the user and the robot can be exploited to influence the quality and nature of the collected data. It was observed in robotics that several learning strategies need to be combined to acquire skills efficiently (see, e.g., [15]).

**Demonstration Modalities**

In LfD, the demonstrations can take various forms, requiring the design of interfaces, devices, and control techniques that can be used for efficient collection of data. Some of these can be linked to social learning processes such as *observational learning* and *kinesthetic teaching*. The first refers to the visual observation of a demonstrator achieving a task with the goal of reproducing it. In robotics, observational learning will typically exploit vision systems, but it is often conceded that the perception system can extend to marked-based systems (mocap) or other forms of motion recording devices (e.g., gyroscopes and accelerometers).

Kinesthetic teaching, also called direct teaching, refers to the process of moving the robot manually, by haptic interaction, while the robot records the demonstration through its sensors (proprioception). In the case of articulated robots, the user can demonstrate the task in the workspace of the robot by constraining the movement to the robot capability. Compared to observational learning, kinesthetic teaching simplifies the correspondence problem [41] (also called motion retargeting or body mapping problem). A potential downside is that the user does not execute the task on his/her own and can control only a limited number of articulations simultaneously. In practice, such limitation often restricts kinesthetic teaching to tasks that do not involve highly dynamic movements. At a technical level, kinesthetic teaching is often implemented by having the robot actively compensate for the effect of gravity (typically implemented with torque-controlled robots), sometimes also complemented by compensation of inertia, friction and Coriolis effects.

LfD also considers demonstration modalities in the form of specialized devices used to teleoperate the robot. This includes commands from a graphical user interface as discussed in the previous sections, but also various forms of joysticks and exoskeletons. These interfaces either act in a passive way, or they can be provided with feedback mechanisms. The most sophisticated interfaces exploit recent developments in telepresence to allow the user to demonstrate tasks by feeling as if she/he was executing it on her/his own.

**Movement Primitives**

A key research aspect underlying LfD is the design of compact and adaptive movement representations that can be used for both analysis and synthesis. The term *movement primitives* is often employed in this context. The goal of such encoding strategy is to represent movements as a set of adaptive building blocks that can be (re)organized in parallel and in series to create more complex behaviors, see [17] for a review.

**Exploitation of (Co)variations**

LfD algorithms typically need to cope with several forms of variations, which make the problem harder than a simple *record and play* process. First, variations can arise from the constraints of the task to be transferred (e.g., dropping a sugar cube in a small cup requires more precision than dropping a bouillon cube in a big pot). Then, variations can arise from the kinematic structure of the robot (a redundant arm can achieve the same task in different manners). Most often, these variations are better described by taking into account the covariations (instead of each dimension separately). From a statistical perspective, this corresponds to the use of full covariance matrices instead of diagonal covariance matrices. In high-dimensional problems, the full covariance matrices will often be considered with a low rank structure (e.g., PCA decomposition in joint angle space). From a motor control perspective, such approach allows the encoding of coordination patterns and synergies that are of primary importance in many different skills, and at diverse levels [32].

One of the challenges of LfD is to exploit the detected covariations in the demonstrations to retrieve an adaptive and robust controller for the reproduction of the task. This can, for example, take the form of competing constraints through the weighting

of movement primitives activated in parallel, or through a hierarchical organization of the task [35]. This can also be incorporated more directly within a control strategy by retrieving a controller following a minimal intervention control strategy [18, 58].

**Task-Adaptive Movements**
An important aspect of LfD is to enable robots to acquire skills that can be adapted to new situations. A common approach to achieve such generalization capability is to associate *task variables* (what describes the situation) with *movement variables* (what describes the skill) and then use a regression technique to retrieve new *movement variables* from new *task variables* (see, e.g., [44]). Such a regression approach is generic since the task variables can represent a wide range of context features organized in a vector form.

An alternative approach is to encode demonstrations from the perspective of multiple coordinate systems [16, 56]. This is achieved by providing a list of "observers" or frames of reference that can potentially be relevant for the movement or the task to transfer. Such an approach is motivated by the observation that skillful movement planning and control often require the orchestration of multiple coordinate systems that can have varying levels of importance along the task (see, e.g., [7]). Typical examples are movements in object-, body-, head-, or gaze-centered frames of reference that can collaborate in various manners for the different phases of a task. Invariance and coordination extraction in movements are also closely related to the coordinate systems in which the analysis takes place [52]. Task-parameterized movement primitives [16, 56] take inspiration from these lines of research to encode movements from the perspective of multiple coordinate systems, where a statistical analysis is simultaneously conducted in each coordinate system. Such a model encapsulates the variations and coordinations in each frame, enabling the robot to learn from demonstration how to handle perturbations and how to switch between coordinate systems, resulting in improved extrapolation capability (e.g., for the adaptation of movements to new positions of objects involved in a manipulation task).

## *3.1 Learning from Demonstrations of Key Positions*

Inspired by a teacher showing a student the correct tennis swing by moving the student's body into static key poses (e.g. *take back*, *impact*, *follow through*), advanced kinesthetic teaching techniques require the user to demonstrate only the skeleton of the task in the form of key positions of the robot as snapshots of desired behavior [29], which can also be called viapoints or waypoints. A learning and control algorithm can then be employed to optimize the low-level motor control to replicate the demonstrated skeleton while filling the information gaps by dynamic and efficient control commands. These techniques allow the user to accurately control all articulations of the robot, enabling an LfD framework for both accurate and highly dynamic movements.

During the learning phase, the user input is processed into a distribution of optimal trajectories. The user provides a set of $K$ demonstrations $\mathcal{D} = \{\mathcal{D}_k\}_{k=1}^{K}$, with $N$ key positions per demonstration, i.e., $\mathcal{D}_k = \{\{^k\boldsymbol{y}_n\}_{n=1}^{N}, \boldsymbol{s}_k\}$. The approach can be applied in the robot joint space as well as in the robot's end-effector space. $\boldsymbol{s}_k$ reflects the state of the environment during the $k$th demonstration. While the $K$ demonstrations show different solutions of one task, we assume that the number of key positions is constant among all demonstrations and that the order of the key positions is given by the user. In contrast to learning from full trajectories, there is no information about the timing of the demonstrated task (except for the order in which key positions are given). We recover this information by means of optimal control by exploiting the velocity and acceleration limits that we want to impose. These limits are given (e.g., by the user) in the form of element-wise intervals, e.g., $\dot{y}^i \in [-\dot{y}^i_{\text{lim}}, \dot{y}^i_{\text{lim}}]$ and $\ddot{y}^i \in [-\ddot{y}^i_{\text{lim}}, \ddot{y}^i_{\text{lim}}]$ for the $i$th DoF.

The learning phase consists of: (1) constructing optimal basis functions; and (2) learning a distribution of optimal trajectories by computing a probability density function of the basis function weights and the correlations with the task context.

**Optimal Basis Functions**

In order to construct optimal basis functions, we formulate a cascaded time-optimal control problem. The block diagram in Fig. 6 (*right*) gives an overview of the cascaded optimization scheme.

The inner optimal control problem considers a linear second-order system that is constrained to pass through key positions $\boldsymbol{y}_n$ at a given time $t_n$ with minimal effort, i.e.,

$$\begin{aligned} \boldsymbol{y}^*(t) = \arg\min_{\boldsymbol{y}(t)} \quad & \int_0^T \dddot{\boldsymbol{y}}(t)^\top \dddot{\boldsymbol{y}}(t) dt, \\ \text{s.t.} \quad & \boldsymbol{y}(t_n) = \boldsymbol{y}_n, \quad n = 1, \ldots, N, \\ & \ddot{\boldsymbol{y}}(0) = \ddot{\boldsymbol{y}}_1, \quad \ddot{\boldsymbol{y}}(T) = \ddot{\boldsymbol{y}}_N. \end{aligned} \tag{1}$$

A corresponding timing parameter $\boldsymbol{h} \in \mathbb{R}^{N-1}$ that encodes the time passed between two consecutive key positions, i.e., $h_n = t_{n+1} - t_n$. Consequently, the total duration

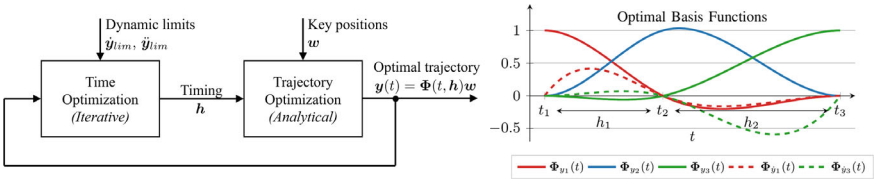



**Fig. 6** *Left*: Within the learning phase, we solve a cascaded optimization problem for each key position demonstration. We exploit linear relations in the resulting optimal trajectory by extracting basis functions $\boldsymbol{\Phi}(t, \boldsymbol{h})$ for a demonstrated skill. *Right*: Basis functions constructed from the optimal control problem in (1) for a single DoF and $N = 3$ key positions

of the motion is given by $T = \sum_{n=1}^{N-1} h_n$. In [66], it has been shown that the solution of the linear optimal control problem in (1) is a polynomial spline in time. We find that the optimal trajectory can be expressed as a linear function of the key points that may come from a human demonstration. The analytical solution to the optimization problem in (1) allows us to derive *optimal basis functions* that map the key positions to an optimal trajectory. Figure 6 (*right*) shows a set of basis functions for a task instance with three key positions with a general timing parameter $\boldsymbol{h}$. Each colored signal represents a basis function that corresponds either to one of the key positions or to one of the boundary velocities.

**Time Optimization**
Within the outer optimization problem, the timing parameter $\boldsymbol{h}$ can be optimized. This is required to achieve efficient and smooth robot motions despite the fact that the human demonstrations do not provide information about timing and velocities of the robot. We find a timing parameter $\boldsymbol{h}^*$ that locally minimizes the movement duration $T$ while satisfying the constraints through the sequential least squares quadratic programming algorithm [34]. For each demonstration of key positions $\boldsymbol{\mathcal{D}}_k$ (setting $\dot{\boldsymbol{y}}_1 = \dot{\boldsymbol{y}}_N = \boldsymbol{0}$), we compute the optimal timing $\boldsymbol{h}_k^*$ and select a conservative timing parameter $\bar{\boldsymbol{h}}$ by selecting each element (i.e., the individual duration between consecutive key positions) as the maximum that appeared among all demonstrations. This conservative timing parameter ensures that the robot does not violate velocity and acceleration constraints when reproducing motions demonstrated by a human through key positions.

**Probability Distribution of Optimal Trajectories**
The linear mapping from human-demonstrated key positions and optimal trajectories can be exploited when multiple demonstrations of the same task are available. In this case, the set of demonstrated key positions can be interpreted as samples from a probability distribution over the space of key positions. The optimal basis functions then allow us to map not only single demonstrations to optimal robot trajectories but also Gaussian distributions of key positions to Gaussian distributions of robot trajectories. By also providing a context variable $\boldsymbol{s}$ that describes the state of the workspace for a given demonstration, correlations between the workspace state and the robot's motion can be learned.

**Learning in Multiple Frames**
Such correlations can be learned in a variety of coordinate systems. If the robot's motion is coupled with the location of object, for example if the robot is supposed to pick up this object, correlations are easier to detect and to learn in an object-centric coordinate system. In [18], poses of objects in the environment are specifically handled by learning the movement statistics in a coordinate system that is attached to the corresponding object. The same can be applied to the statistics of key positions in various frames. The robot's motion can then be generated by mapping the statistics in all coordinate systems into a world-fixed frame and subsequently fused by computing the product of Gaussians.

### 3.2 User Study

We here present the results of a user study comparing the effectiveness of teaching through key positions (Method **KP**) against full trajectories (Method **FT**), alongside other experiments showcasing the effectiveness of the proposed approach.

We conducted a study where 16 novice robot users (mean age 28.8, SD 3.2) provided demonstrations to solve a pick-and-place task with a 7-axis Franka Emika manipulator.

*Experimental Setup*: The task consisted of picking a toy drill (from the YCB dataset [20]) and placing it inside one of two boxes, as shown in Fig. 7. The participants provided 8 demonstrations by displacing the gravity-compensated robot arm. For each demonstration, a drill pose was randomly selected from a pool of predefined demonstration scenarios, along with a corresponding robot's starting configuration.

*Conditions and Protocol*: Participants first filled a brief questionnaire consisting of three 7-point Likert scale statements, aimed at assessing their robotics expertise. An average score of 1.95 was observed (Cronbach's $\alpha = 0.9$), indicating novice users. Each method was used by 8 participants.

*Results*: We compared the participants' effectiveness of teaching with Method **KP** and Method **FT** and with different numbers of demonstrations, in terms of success of the task reproductions and of quality of the reproduced trajectory.

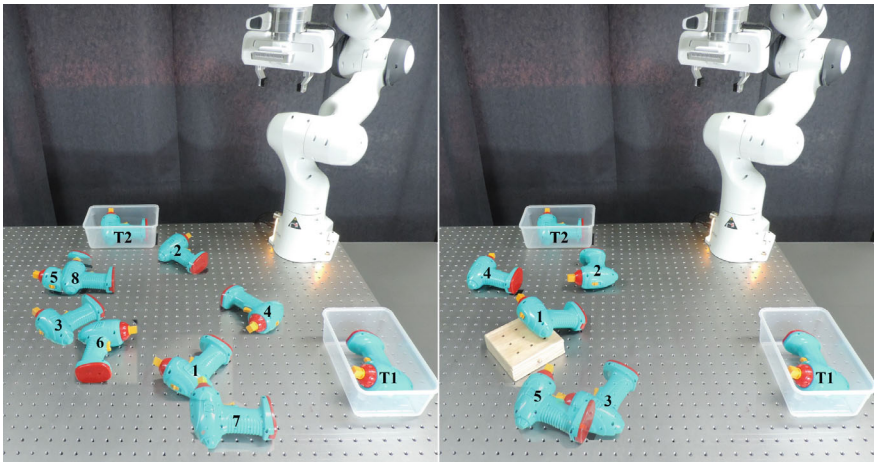



**Fig. 7** Experimental setup of the user study: drill pick-up (1–8, left figure) and drop locations (T1 and T2) used for the demonstration collection, and test locations used in the evaluation phase (1–5, right figure)

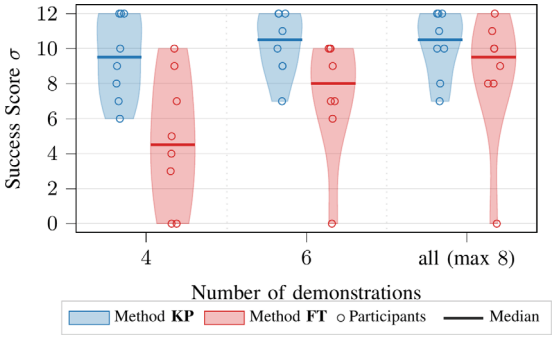



**Fig. 8** Distribution of cumulative success score $\sigma$, separated by method and by number of demonstrations utilized for training

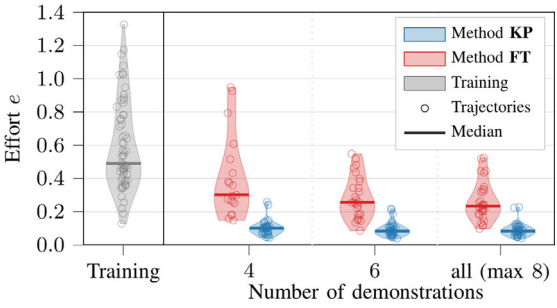



**Fig. 9** Distribution of acceleration effort $e$ of trajectory reproductions, separated by method and by number of demonstrations utilized for training. The leftmost violin instead shows $e$ of the collected demonstrations

Figure 8 compares the distribution of cumulative score $\sigma$ for each teaching method. We observe a statistically significant score difference between methods when trained with 4 and 6 demonstrations (Mann–Whitney U test,[1] $p < .01^{**}$), with a median score of 9.5 for Method **KP** vs 4.5 for Method **FT** when trained with 4 demonstrations. This difference can be explained by the fact that Method **KP** needs to learn in a lower-dimensional space than Method **FT**; a relevant difference when learning from a few demonstrations. As expected, $\sigma$ improves for both methods with more than 6 demonstrations, with negligible differences between methods.

Figure 9 compares the distribution of acceleration effort $e$ for each teaching method and for the participants' demonstrations. The acceleration effort of the robot is a metric for measuring the efficiency of the generated robot motion. A low-acceleration robot motion may require less energy to execute and in higher predictability for the human co-worker. Method **KP** produces trajectories with significantly lower $e$ compared to Method **FT** (Mann–Whitney U test, $p < .01^{**}$ for any number of demonstrations used in the training). This is a direct consequence of the underlying optimization process.

As also observed in previous work [1], participants are almost twice as fast when providing full trajectory demonstrations (median 18 s) compared to key position demonstrations (median 28 s). However, the success score achieved by Method

[1] Data was tested for normality, always rejecting the null hypothesis (Shapiro-Wilk test, $p < .01^{**}$). We, therefore, used non-parametric tests.

**KP** trained with 4 demonstrations ($\sigma^4_{KP,\text{median}} = 9.5$) is comparable with the one of Method **FT** with twice the demonstrations ($\sigma^8_{\text{FT, median}} = 9.5$). It can be argued that, in a real scenario, operations other than the demonstration collection, such as setting the environment or the robot arm's initial configuration, would dominate the process, making the aforementioned time difference negligible.

## 4 Conclusion

In this chapter, we have introduced two approaches to simplifying robot programming: block-based programming (BBP) and learning from demonstration (LfD). These methods offer complementary strengths for intuitive robotic control. BBP, with its visual drag-and-drop interface, is well-suited for the programming of trajectories, providing precise control over task logic and structure. However, it can become cumbersome when handling complex behaviors, particularly for tasks that cannot be easily defined by a trajectory, such as polishing or other nuanced processes. LfD, on the other hand, allows robots to learn tasks through human demonstration, making it more intuitive for continuous or motor-driven behaviors. However, LfD lacks the precision and granular control inherent in BBP. Combining these approaches creates a powerful hybrid system: BBP can define high-level task structures, such as conditions and sequences, while LfD can handle complex, dynamic actions like object manipulation. This hybrid system allows users to refine LfD-learned behaviors within BBP or convert demonstrated tasks into reusable blocks for more sophisticated workflows. The result is an iterative improvement process in which users begin with basic programming in BBP and enhance it with LfD, creating a flexible and adaptable system that balances precise control with intuitive learning.

The two implementation of BBP and LfD that we have presented allow non-expert users to learn to autonomously program robots and develop a feeling of control over the machines. However, tools must be designed specifically for the target users, taking into account their unique backgrounds, motivations, and interests, to ensure software dependability [14]. If further testings with target users is still needed to fully guarantee this dependability, our results corroborate the relevance of the proposed approaches. On the long term, the integration of empowering tools for production work might enhance work conditions and, hopefully, give back the control over their work to the workers, turning them into makers rather than passive users [54]. Indeed, two visions regarding the future of partial automation can be identified: Human Emulation and Human Complementary [55]. On the one hand, in the emulation scenario, the machine will become more and more intelligent and autonomous to finally be able to replace humans. The human intervention is a temporary solution to help the robot perform the work until it can do it autonomously. The tasks left to the worker are typically extremely monotonous, possible leading to deskilling [23] and a feeling of mechanization [13]. On the other hand, in the complementary scenario, it is argued that humans and machines have complementary capabilities that can be optimally

combined through collaboration. Indeed, Human-Robot Collaboration allows for automation productivity with human flexibility. Technology is a tool for humans to render work more efficient. If robots as human emulators represent a danger for workers, robots as human complements (or tools) can on the contrary help improve working conditions and even re-humanize work [40]. Indeed, procedural, repetitive tasks that requires the human to act like a machine can be automated, while tasks requiring common sense, creativity and dexterity can be left to the human. However, in order to achieve this empowerment through technology, programming solutions that foster worker reskilling are needed [31]. In this chapter, we have presented two possible approaches that allow to cover a wide range of tasks. Our approach is particularly interesting for small to medium enterprises (SMEs) that typical produce smaller volumes. Flexibility is key since the systems need to be usable for different tasks to reach the volume needed to make the system profitable, but, even more importantly, for the company to be able to re-task the robot rapidly. Indeed, if cycle time is crucial for mass production and requires expert, specialized solutions that are optimized for speed, the critical factor for smaller production volumes is the time needed to switch from one application to another, the cycle time being less important. In addition, because of the market versatility and the associated rapid changes in the market demand, reactivity is a key issue: since most SMEs cannot afford to hire an expert for their production, having to outsource the reprogramming of the system for new tasks also implies a delay in production launch. Solutions that allow non-robotic experts to re-task production systems are thus needed not only to reduce costs, but also to save time. In this chapter, we have presented two promising approaches to empower production workers and give them the needed autonomy to easily program robots on their own, contributing to the development of a Human Complementarity approach to automation.

## References

1. Akgun, B., Cakmak, M., Yoo, J.W., Thomaz, A.L.: Trajectories and keyframes for kinesthetic teaching: a human-robot interaction perspective. In: Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction, pp. 391–398. (2012)
2. Blanc, C., Boudry, L., Sonderegger, A., Nembrini, J., Dégallier-Rochat, S.: Empowering production workers to program robots: a no-code, skill-based approach. In: International Conference on Computer-Human Interaction Research and Applications, pp. 21–39. Springer (2023)
3. Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. Robot. Auton. Syst. **57**(5), 469–483 (2009)
4. Bangor, A., Kortum, P., Miller, J.: Determining what individual SUS scores mean: adding an adjective rating scale, (2009)
5. Barbeite, F.G., Weiss, E.M.: Computer self-efficacy and anxiety scales for an Internet sample: testing measurement equivalence of existing measures and development of new scales. Comput. Hum. Behav. **20**(1), 1–15 (2004)
6. Bau, D., Gray, J., Kelleher, C., Sheldon, J., Turbak, F.: Learnable programming. Commun. ACM **60**(6), 72–80 (2017)

7. Bennequin, D., Fuchs, R., Berthoz, A., Flash, T.: Movement timing and invariance arise from several geometries. PLoS Comput. Biol. **5**(7), 1–27 (2009)
8. Billard, A.G., Calinon, S., Dillmann, R.: Learning from humans. In: Siciliano, B., Khatib, O. (eds.) Handbook of Robotics, pp. 1995–2014. Springer, Secaucus, NJ, USA (2016) 2nd edn., chap. 74
9. Bitonneau, D.: Conception de systèmes cobotiques industriels: approche robotique avec prise en compte des facteurs humains: application à l'industrie manufacturière au sein de Safran et ArianeGroup, Université de Bordeaux (2018). Ph.D. thesis
10. Brooke, J.: SUS: A "Quick and DirtyâŁž Usability Scale, pp. 207–212. CRC Press (1996)
11. Bruckman, J.C.: Overcoming resistance to change: causal factors, interventions, and critical values. Psychol.-Manag. J. **11**(2), 211–219 (2008). Routledge. Eprint: https://www.tandfonline.com/doi/pdf/10.1080/10887150802371708
12. Brynjolfsson, E., McAfee, A.: The Second Machine Age: Work, Progress, and Prosperity in a Time of Brilliant Technologies. W. W. Norton & Company (2014). Google-Books-ID: WiKwAgAAQBAJ
13. Bui, L.: Asian roboticism: connecting mechanized labor to the automation of work. Perspect. Glob. Dev. Technol. **19**(1–2), 110–126 (2020)
14. Burnett, M., Cook, C., Rothermel, G.: End-user software engineering. Commun. ACM **47**(9), 53–58 (2004)
15. Cakmak, M., DePalma, N., Arriaga, R.I., Thomaz, A.L.: Exploiting social partners in robot learning. Auton. Robot. **29**(3–4), 309–329 (2010)
16. Calinon, S.: A tutorial on task-parameterized movement learning and retrieval. Intell. Serv. Robot. **9**(1), 1–29 (2016)
17. Calinon, S.: Mixture models for the analysis, edition, and synthesis of continuous time series. In: Bouguila, N., Fan, W. (eds.) Mixture Models and Applications, pp. 39–57. Springer (2019)
18. Calinon, S., Bruno, D., Caldwell, D.G.: A task-parameterized probabilistic model with minimal intervention control. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 3339–3344. , Hong Kong, China (2014)
19. Calinon, S., Lee, D.: Learning control. In: Vadakkepat, P., Goswami, A. (eds.) Humanoid Robotics: A Reference, pp. 1261–1312. Springer (2019). https://doi.org/10.1007/978-94-007-6046-2_68
20. Calli, B., Walsman, A., Singh, A., Srinivasa, S., Abbeel, P., Dollar, A.M.: Benchmarking in manipulation research: using the Yale-CMU-Berkeley object and model set. IEEE Robot. Autom. Mag. **22**(3), 36–52 (2015)
21. Coates, A., Abbeel, P., Ng, A.Y.: Apprenticeship learning for helicopter control. Commun. ACM **52**(7), 97–105 (2009)
22. Connolly, C., Murphy, E., Moore, S.: Programming anxiety amongst computing students–a key in the retention debate? IEEE Trans. Educ. **52**(1), 52–56 (2009). Conference Name: IEEE Transactions on Education
23. de Assis Dornelles, J., Ayala, N.F., Frank, A.G.: Collaborative or substitutive robots? Effects on workers' skills in manufacturing activities. Int. J. Prod. Res. 1–34 (2023)
24. Derus, S.R.M., Ali, M., A.Z.: Difficulties in learning programming: views of students. In: 1st International Conference on Current Issues in Education (ICCIE 2012), pp. 74–79. (2012)
25. Dimirovski, G.M., Talha Dinibütün, A., Kile, F., Neck, R., Stahre, J., Vlacic, L.B.: Control system approaches for sustainable development and instability management in the globalization age. Annu. Rev. Control. **30**(1), 103–115 (2006)
26. Ford, M.: Rise of the Robots: Technology and the Threat of a Jobless Future, Basic Books (2015)
27. Fraser, N.: Ten things we've learned from Blockly. In: IEEE Blocks and Beyond Workshop (Blocks and Beyond), pp. 49–50. (2015)
28. Hsu, T.-C., Hwang, G.-J.: Interaction of visual interface and academic levels with young students' anxiety, playfulness, and enjoyment in programming for robot control. Univ. Access Inf. Soc. **22**(1), 213–225 (2023)

29. Jankowski, J., Racca, M., Calinon, S.: From key positions to optimal basis functions for probabilistic adaptive control. IEEE Robot. Autom. Lett. **7**(2), 3242–3249 (2022)
30. Kaasinen, E., Schmalfuß, F., Özturk, C., Aromaa, S., Boubekeur, M., Heilala, J., Heikkilä, P., Kuula, T., Liinasuo, M., Mach, S., Mehta, R., Petäjä, E., Walter, T.: Empowering and engaging industrial workers with Operator 4.0 solutions. Comput. Ind. Eng. **139**, 105678 (2020)
31. Kadir, B.A., Broberg, O.: Human well-being and system performance in the. Int. J. Ind. Ergon. **76**, 102936 (2020). transition to industry 4.0
32. Kelso, J.A.S.: Synergies: atoms of brain and behavior. In: Sternad, D. (ed.) Progress in Motor Control. Advances in Experimental Medicine and Biology, vol. 629, pp. 83–91. Springer, US (2009)
33. Kieras, D.E., Bovair, S.: The role of a mental model in learning to operate a device. Cogn. Sci. **8**(3), 255–273 (1984)
34. Kraft, D.: A software package for sequential quadratic programming. Forschungsbericht. Wiss. Berichtswesen d. DFVLR, Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln (1988)
35. Kulic, D., Takano, W., Nakamura, Y.: Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive hidden Markov chains. Int. J. Robot. Res. **27**(7), 761–784 (2008)
36. Lasota, P.A., Fong, T., Shah, J.A.: A survey of methods for safe human-robot interaction. Found. Trends Robot. **5**(4), 261–349 (2017)
37. Laugwitz, B., Held, T., Schrepp, M.: Construction and evaluation of a user experience questionnaire. In: Holzinger, A. (ed.) HCI and Usability for Education and Work. Lecture Notes in Computer Science, pp. 63–76. Springer, Berlin, Heidelberg (2008)
38. Mayr-Dorn, C., Winterer, M., Salomon, C., Hohensinger, D., Fürschuss, H.: Assessing industrial end-user programming of robotic production cells: a controlled experiment. J. Syst. Softw. **195**, 111547 (2023)
39. McClure, P.K.: "You're fired", says the robot: the rise of automation in the workplace, technophobes, and fears of unemployment. Soc. Sci. Comput. Rev. **36**(2), 139–156 (2018)
40. Mormann, H., Endrissat, N.: Humanizing organizations: the professional role of HRM in organizational transformation. Die Unternehmung **76**(4), 418–429 (2022)
41. Nehaniv, C.L., Dautenhahn, K. (eds.): Imitation and Social Learning in Robots, Humans, and Animals: Behavioural. Social and Communicative Dimensions. Cambridge University Press, Cambridge, UK (2007)
42. Ninomiya, T., Fujita, A., Suzuki, D., Umemuro, H.: Development of the multi-dimensional robot attitude scale: constructs of people's attitudes towards domestic robots. In: Tapus, A., André, E., Martin, J.-C., Ferland, F., Ammi, M. (eds.) Social Robotics. Lecture Notes in Computer Science, pp. 482–491. Springer, Cham (2015)
43. Nomura, T., Kanda, T., Suzuki, T.: Experimental investigation into influence of negative attitudes toward robots on human-robot interaction. AI Soc. **20**, 138–150 (2006)
44. Paraschos, A., Daniel, C., Peters, J., Neumann, G., Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z.: Probabilistic movement primitives. In: Weinberger, K.Q. (ed.) Advances in Neural Information Processing Systems (NIPS), pp. 2616–2624. Curran Associates, Inc (2013)
45. Ratliff, N., Ziebart, B.D., Peterson, K., Bagnell, J.A., Hebert, M., Dey, A., Srinivasa, S.: Inverse optimal heuristic control for imitation learning. In: International Conference on Artificial Intelligence and Statistics (AIStats), pp. 424–431. (2009)
46. Reich-Stiebert, N., Eyssel, F.: Learning with educational companion robots? Toward attitudes on education robots, predictors of attitudes, and application potentials for education robots. Int. J. Soc. Robot. **7**(5), 875–888 (2015)
47. Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., Kafai, Y.: Scratch: programming for all. Commun. ACM **52**(11), 60–67 (2009)
48. Romero, D., Stahre, J., Taisch, M.: (2020). The operator 4.0: towards socially sustainable factories of the future

49. Schou, C., Andersen, R.S., Chrysostomou, D., Bøgh, S., Madsen, O.: Skill-based instruction of collaborative robots in industrial settings. Robot. Comput.-Integr. Manuf. **53**, 72–80 (2018)
50. Schwab, K.: Shaping the Future of the Fourth Industrial Revolution. Currency , (2018)
51. Sonderegger, A., Heyden, K., Chavaillaz, A., Sauer, J.: AniSAM & AniAvatar: animated visualizations of affective states. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, p. 4837 (2016)
52. Sternad, D., Park, S.-W., Mueller, H., Hogan, N.: Coordinate dependence of variability analysis. PLoS Comput. Biol. **6**(4), 1–16 (2010)
53. Sætren, G.B., Hogenboom, S., Laumann, K.: A study of a technological development process: human factors-the forgotten factors? Cognit. Technol. Work **18**(3), 595–611 (2016)
54. Taylor, M.P., Boxall, P., Chen, J.J.J., Xu, X., Liew, A., Adeniji, A.: Operator 4.0 or Maker 1.0? Exploring the implications of Industrie 4.0 for innovation, safety and quality of work in small economies and enterprises. Comput. Ind. Eng. **139**, 105486 (2020)
55. Terveen, L.G.: Overview of human-computer collaboration. Knowl.-Based Syst. **8**(2), 67–81 (1995)
56. Ti, B., Razmjoo, A., Gao, Y., Zhao, J., Calinon, S.: A geometric optimal control approach for imitation and generalization of manipulation skills. Robot. Auton. Syst. **164**, 104413 (2023)
57. Tisza, G., Sharma, K., Papavlasopoulou, S., Markopoulos, P., Giannakos, M.: Understanding fun in learning to code: a multi-modal data approach. In: Proceedings of the 21st Annual ACM Interaction Design and Children Conference, IDC '22, pp. 274–287. Association for Computing Machinery, New York, NY, USA (2022)
58. Todorov, E., Jordan, M.I.: A minimal intervention principle for coordinated movement. In: Advances in Neural Information Processing Systems (NIPS), pp. 27–34. (2002)
59. Villani, V., Pini, F., Leali, F., Secchi, C.: Survey on human-robot collaboration in industrial settings: safety, intuitive interfaces and applications. Mechatronics **55**, 248–266 (2018)
60. Weintrop, D., Afzal, A., Salac, J., Francis, P., Li, B., Shepherd, D.C., Franklin, D.: Evaluating CoBlox: a comparative study of robotics programming environments for adult novices. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, pp. 1–12. ACM, Montreal, QC, Canada (2018)
61. Weintrop, D., Shepherd, D.C., Francis, P., Franklin, D.: Blockly goes to work: block-based programming for industrial robots. In: 2017 IEEE Blocks and Beyond Workshop (B&B), pp. 29–36 (2017)
62. Weintrop, D., Wilensky, U.: To block or not to block, that is the question: students' perceptions of blocks-based programming. In: Proceedings of the 14th International Conference on Interaction Design and Children, pp. 199–208. (2015)
63. Weintrop, D., Wilensky, U.: Comparing block-based and text-based programming in high school computer science classrooms. ACM Trans. Comput. Educ. **18**(1), 3:1–3:25 (2017)
64. Whiten, A., McGuigan, N., Marshall-Pescini, S., Hopper, L.M.: Emulation, imitation, over-imitation and the scope of culture for child and chimpanzee. Philos. Trans. R. Soc. B **364**(1528), 2417–2428 (2009)
65. Talha Yildiz, T., Seferoglu, S.S.: The effect of robotic programming on coding attitude and computational thinking skills toward self-efficacy perception. J. Learn. Teach. Digit. Age **6**(2), 101–116 (2021)
66. Zhang, Z., Tomlinson, J., Martin, C.: Splines and linear control theory. Acta Math. Appl **49**, 1–34 (1997)

**Open Access** This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (http://creativecommons.org/licenses/by-nc-nd/4.0/), which permits any noncommercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if you modified the licensed material. You do not have permission under this license to share adapted material derived from this chapter or parts of it.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

