Interactive Spatial Augmented Reality in Collaborative Robot Programming: User Experience Evaluation

Zdeněk Materna, Michal Kapinus, Vítězslav Beran, Pavel Smrž, and Pavel Zemčík

***Abstract*— This paper presents a novel approach to interaction between human workers and industrial collaborative robots. The proposed approach addresses problems introduced by existing solutions for robot programming. It aims to reduce the mental demands and attention switches by centering all interaction in a shared workspace, combining various modalities and enabling interaction with the system without any external devices. The concept allows simple programming in the form of setting program parameters using spatial augmented reality for visualization and a touch-enabled table and robotic arms as input devices. We evaluated the concept utilizing a user experience study with six participants (shop-floor workers). All participants were able to program the robot and to collaborate with it using the program they parametrized. The final goal is to create a distraction-free, usable and low-effort interface for effective human-robot collaboration, enabling any ordinary skilled worker to customize the robot's program to changes in production or to personal (e.g. ergonomic) needs.**

I. Introduction

Contemporary collaborative robots are collaborative in the sense that for human workers, it is safe to work alongside them. However, human-robot interaction is very limited if it exists at all: The behavior of the robot is pre-programmed without cognition of an environment, a user, tools, or the parts necessary for a given task. The robots are programmed by domain experts using specialized devices and an expert is needed even for small changes in the program. It is expected that, in the near future, collaborative robots will be cheaper and thus more affordable for small and medium-sized enterprises (SMEs). In such companies, all of the aforementioned issues will be even more prominent. As robots in SMEs will have to deal with higher product variability (smaller batches, customization) it would be beneficial to allow workers with no specific skills to make changes in a robot's program. At the same time, it will be necessary to support a close human-robot collaboration, as with rising cost of human labor, it might be expected that a trend will occur to offload non-ergonomic or repetitive parts of the workflow to robots. In order to allow this, robots will have to perceive and interact.

In this work, we present a novel approach to programming collaborative robots based on cognition, spatial augmented reality (SAR) and multimodal input and output. In order to make programming as simple as possible, programming takes place on a high level of abstraction where no robot-specific knowledge is necessary. Our intention was to make interaction with robots easy, fun, safe and effective.

All authors are affiliated with the Brno University of Technology, Faculty of Information Technology, Centre of Excellence IT4Innovations, Bozetechova 1/2, Brno, 612 66, Czech Republic. Contacts: $imaterna, ikapinus, beranv, smrz, zemcik@fit.vutbr.cz$

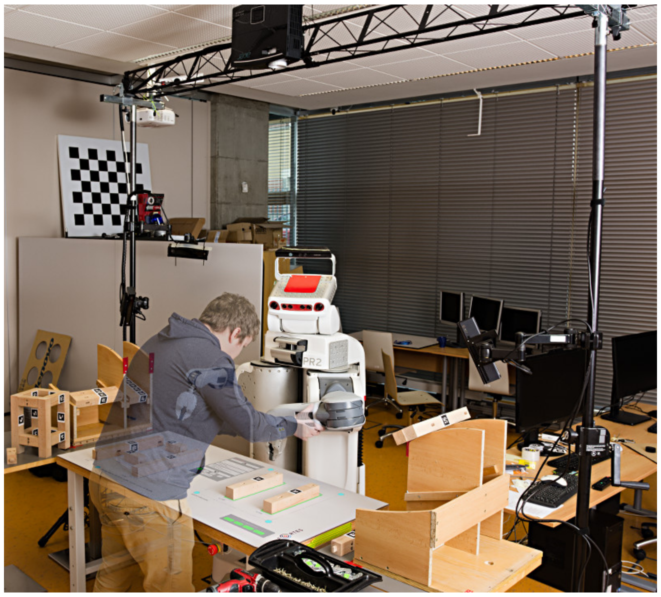


Fig. 1. Setup of the novel interactive system concept where all the interaction elements (visualization and control) are gathered in a shared workspace (example of setting program parameters using a robotic arm and gestures; image edited).

In order to evaluate the approach, we developed a proof of concept system (see Fig. 1)[1] and carried out initial user experience testing. The purpose of the testing was to discover whether there are some fundamental usability issues related to the approach as well as to find out issues related to the current implementation. In the experiment, the robot played the role of a worker's assistant, preparing parts for assembly in a fictional SME.

II. Related Work

Various approaches exist aimed at the simplification of robot programming or to support human-robot collaboration on a joint task. One of the techniques used to make programming robots more suitable for non-expert users is programming by demonstration. For instance, the approach proposed in [1] was rated by non-expert users as highly intuitive. However, the tasks are quite simple and there is

[1]The code is available at https://github.com/robofit/artable.

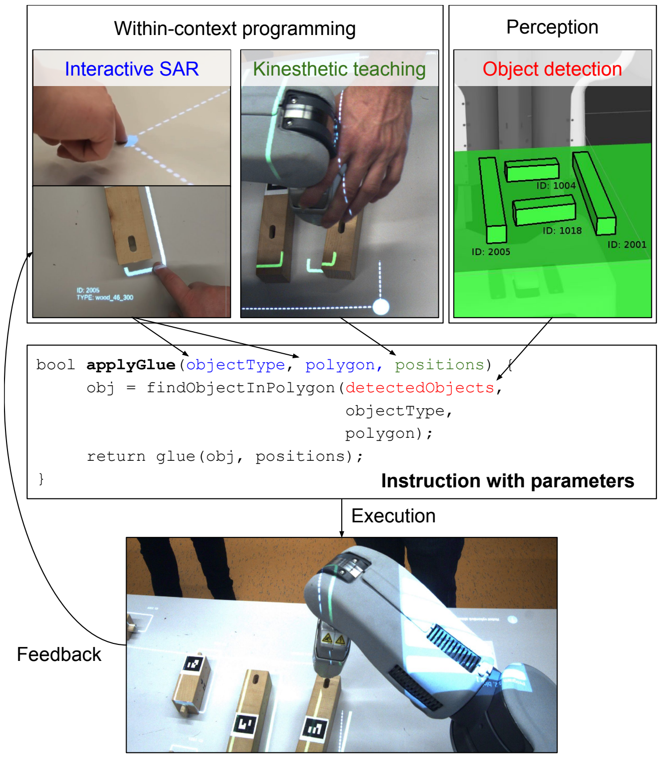
no feedback for the user. In [2], kinesthetic teaching is used in conjunction with an iconic based programming to enable users to create and edit non-trivial programs. While the usage of a graphical user interface (GUI) on a standard monitor adds more control over the program and provides feedback, it also leads to attention switches.

The system described in [3] uses behavior trees to represent the program and was successfully deployed at an SME. The program itself is created on the monitor. The parameters of the program could be set using GUI, object recognition or kinesthetic teaching. The usage of behavior trees leads to high flexibility and the creation of reusable pieces of programs; however, it also inevitably leads to a more complicated GUI. Similarly, the system described in [4] enables users to create complex programs using kinesthetic teaching and object recognition. However, three different GUIs and voice input are involved. Moreover, its target user group consists of general programmers.

The previous approaches share a common disadvantage: The inability to show information within a task context. On the other hand, [5] uses physical blocks to create a program which is highly intuitive (requires no training), although it is limited to trivial tasks. Recently, augmented reality (AR) has been used to show important information within a task context. Probably the most common approach is to use a hand-held device. In [6], the authors recruited robot programmers and evaluated a tablet-based AR interface for programming abstracted industrial tasks. From the results, it seems that the usage of an AR may lead to a decrease in the workload and higher motivation to perform accurately. However, the usage of a tablet prevents the usage of both hands. A head-mounted display frees the user's hands and according to [7] might lead to faster task completion times and higher accuracy. Unfortunately, the currently available devices have a limited field of view. Also, a head-mounted display probably would not be suitable for long-time usage. On the other hand, SAR is able to show information in context, does not require any hand-held devices, is suitable for long-term usage, and is visible to anyone. It was recently used to implement an interactive work desk [8], show instructions to workers [9], or to show robotic data and learn trajectories [10].

To the best of our knowledge, there is currently no existing interactive system targeting all of the following important issues:

- problems with attention switching when a monitor or a hand-held device is used to visualize the programming interface and system status during operation,
- too much information is presented to the user, leading to a higher mental workload,
- external devices are needed to fully interact with the robotic system (during both the programming and processing phases),
- low level of abstraction allowing only medium-expert users to program the robot.



```
bool applyGlue(objectType, polygon, positions) {
    obj = findObjectInPolygon(detectedObjects,
                              objectType,
                              polygon);
    return glue(obj, positions);
}
```

Fig. 2. Illustration of program parameters' definition (combination of manually set parameters by the user with perceived information by the system) and its execution with visual feedback.

III. Proposed Approach

We propose and initially evaluate a novel approach to collaborative robot programming with the following attributes (see also Fig. 2):

- avoiding switching of the user's attention during programming and cooperation by placing all the interaction elements in a shared workspace,
- decreasing the mental demands on the users by presenting the relevant information according to the current context,
- avoiding the usage of further external devices to interact with the system by making the shared workspace itself interactive,
- allowing non-expert users to work with the system by utilizing a high level of abstraction to program a robot.

Based on literature review and the current state of the technology, we see SAR as the most suitable instrument to visualize a user interface within a task context. While previous research has shown that gesture control is the preferred input modality for setting the parameters of common industrial tasks, we decided to use a touch-enabled table, which was also rated highly [11], and which is much more reliable. Moreover, together with SAR, it creates a similar user experience to tablets and smart phones, the usage of which is well-known to the general public. For tasks requiring 3D data input, the robot's arms could be used.

The user interface should be minimalistic, as the interface elements have to share space with real-world objects in the workspace: Tools, parts, etc. However, the design of the elements should allow convenient touch control. Depending on the state of the task, only the relevant information should be shown to lower the cognitive load [12]. The interface should clearly indicate the current state of the system, including an explicit representation of the robot's program and the context of the current program instruction (what happened before it and what is going to happen after it). Additional modalities, such as sound or light, could be used to for instance attract attention in special cases.

In order to make programming as well as the user interface as simple as possible, we decided to use complex instructions with a high amount of underlaying autonomy, at the price of lowering expressivity (see Fig. 2). While theoretically, with the system from [3] one can create complex instructions from basic ones, it also makes the user interface complex and the program representation complicated. For instance, one has to set several poses, specify open and close gripper commands, etc. We believe that, for the sake of simplicity, the user should be abstracted from such low-level commands and the robot should perform them automatically.

To achieve a high level of abstraction and effective collaboration, the robot needs to perceive its surroundings as well as track its human coworker(s) and plan motions according to the current situation.

IV. Proof of concept system

To evaluate the proposed approach, a proof of concept system has been developed. The system allows end-user programming of selected industrial tasks.

A. Setup

The experimental setup (see Fig. 1) was designed to be easy to deploy and modular. It is centered around a standard workshop table equipped with a capacitive touch foil. On the sides, two speaker stands are placed, connected by a truss. The truss is equipped with an Acer P6600 projector. There is a Microsoft Kinect V2 camera on each stand for object detection and calibration of the system. On one stand, there is an additional Kinect for user tracking. Each stand has its own processing unit (Intel NUC) where the projector and sensors are connected (in the study, only one projector was utilized). The unit is connected to the central computer using a wired network. The system is designed to be modular in a way so that it supports $1..n$ stands.

As a demonstrator of a near-future collaborative robot, we use the intrinsically safe PR2. The robot provides an additional set of sensors (Kinect and cameras on the head, cameras in the forearms). There is also a physical stop button under the table which shuts down the robot's motors.

B. System design

The system's state and behavior are defined and controlled by the central node and it can be manipulated by an arbitrary number of user interfaces. For instance, we currently use two interfaces: GUI projected on the table and a sound interface, providing audio feedback (e.g. confirmation of action, errors, etc.).

All parts of the system must be mutually calibrated first. Calibration of the Kinects utilizes an AR tracking library[2] to detect three markers placed on the table. One marker serves as an origin of the coordination system; the two others determine the X and Y axes. The PR2 robot is calibrated in the same way, using a head-mounted Kinect. To calibrate the projectors, a checkerboard pattern is displayed by each projector, and its corners are detected using already calibrated Kinects. In order to calibrate the touch-enabled surface, the points are projected on the table and the user has to click them. Then, homography is computed and used to convert the internal coordinates of the touch device into the common coordinate system.

Each of the objects used in our study has a set of two AR tags printed on the body, and multimarker detection is used to gain a unique ID of the object and its pose. Each object has an object type and a bounding box defined.

The manipulation pipeline is based on MoveIt! [13] and a library for grasp planning[3].

C. Program representation

The program in our system is a set of instructions, collected into blocks. Each program contains $1..n$ blocks; each block contains $1..n$ instructions. Every instruction execution can result in success (e.g. a successfully picked up object) or failure (e.g. failed to apply glue). Based on this result, the next instruction is determined. With this approach, simple branching and cycling of the program are possible (e.g. picking up objects from a feeder until the picking up failed, i.e. until there are no objects left). For an example of a program structure in the form of a graph, see Fig. 5.

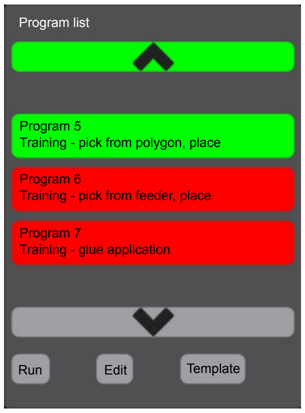
Contrary to the conventional methods of programming robots, no precomputed joint configurations or arm paths are stored. By combining the perception capabilities of the system and on-the-fly motion planning, we do not rely on e.g. storing exact object positions.

It can be expected that the parameters of the program will be changed more often than the structure of the program. For this reason, we have divided the programming process into two parts. First, an empty template is created offline. This template can be seen as a description of an industrial technological process. It contains a set of instructions with defined transitions; however, without parameters. Thus, the template can be created once and later be adapted to conform to different products by setting instruction parameters.

D. Supported instructions

The system currently supports the following parametric instructions: *pick from polygon* (to pick up an object from a table), *pick from feeder* (to pick up parts from a gravity feeder), *place to pose* (to place a previously picked-up object on a selected place on the table) and *apply glue* (simulated

[2]http://wiki.ros.org/ar_track_alvar
[3]https://github.com/davetcoleman/moveit_simple_grasps

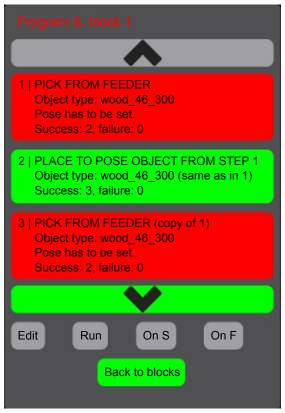

Program list

Program 5
Training - pick from polygon, place

Program 6
Training - pick from feeder, place

Program 7
Training - glue application

(a) List of programs. Green ones are ready to run, red ones need to set parameters.

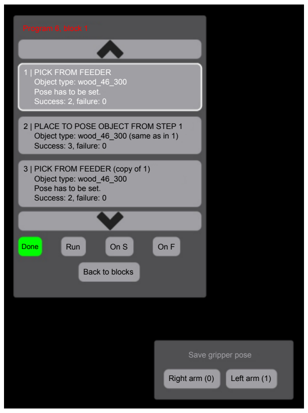

Program 6, block 1

1 | PICK FROM FEEDER
Object type: wood_46_300
Pose has to be set.
Success: 2, failure: 0

2 | PLACE TO POSE OBJECT FROM STEP 1
Object type: wood_46_300 (same as in 1)
Success: 3, failure: 0

3 | PICK FROM FEEDER (copy of 1)
Object type: wood_46_300
Pose has to be set.
Success: 2, failure: 0

(b) List of instructions. Green ones are ready to run, red ones need to set parameters.

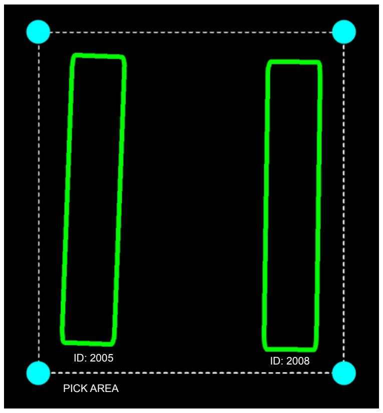

Program 6, block 1

1 | PICK FROM FEEDER
Object type: wood_46_300
Pose has to be set.
Success: 2, failure: 0

2 | PLACE TO POSE OBJECT FROM STEP 1
Object type: wood_46_300 (same as in 1)
Success: 3, failure: 0

3 | PICK FROM FEEDER (copy of 1)
Object type: wood_46_300
Pose has to be set.
Success: 2, failure: 0

(c) A small dialog shows if the robot is able to detect an object in the feeder and allows the user to save the arm pose.



(d) Polygon defining the area on the table from which the objects will be picked up. The green outlines correspond to detected objects.

Fig. 3. Examples of different widgets from a proof of concept system.

gluing). Each of these instructions has certain parameters to be set by the user.

The object type must be set for all of these instructions. For the *pick from polygon* and *apply glue*, a polygon defining the area of interest on the table has to be set, so that the user can limit objects of the given type affected by the instructions.

For the *pick from feeder*, a pre-picking pose (see Fig. 4(c)), used for object detection, has to be set using the robot's arm. While executing this instruction, the robot moves to the stored pose, observes the objects with its forearm camera and picks up the closest object in the direction of the gripper. For *apply glue*, the poses where the glue is supposed to be applied have to be set using an arbitrary arm of the robot.

There are also a couple of non-parametric instructions: *get ready*, *wait for user*, and *wait until user finishes*. The first one moves the robot's arms to their default position. The other instructions allow the synchronization of the system and the user. The *wait for user* instruction will pause the program execution until the user is in front of the table, while *wait until user finishes* will pause the program until the user finishes current interaction with the objects on the table. In our experiments, the behavior of these two instructions was simulated and controlled by the Wizard of Oz approach.

E. User Interaction

The interaction between the user and the system is currently achieved using three modalities: GUI projected on the touch-enabled surface (which serves as an input for the system and feedback for the user), kinesthetic teaching (input to the system only), and sound (feedback for the user only).

The GUI is composed of various widgets. The list of programs (see Fig. 3(a)) shows all the programs stored in the system. The color of each entry suggests whether the program has set all the parameters (green; only these can be started) or some of them are not set (red). Any program can be templated (it is duplicated as a new program, with no parameters set) or edited (the user may set or adjust its parameters). During the program editation, the user can see a list of blocks of the selected program and can edit a selected block or get back to the list of programs.

When editing a block of a program, the list of instructions is shown (see Fig. 3(b)). The selected instruction is always in the middle (with exception for the first and the last one) so the user can see its context. Similarly to the program list, each instruction has either a red or a green background, indicating whether it has all the parameters set. When all the parameters have been set, the selected instruction can be executed. Moreover, a gray instruction background suggests a non-parametric instruction. There are also buttons to navigate through the program, to select an instruction following either the successful or failed execution of the current instruction.

When a program has been executed, the list of instructions differs slightly. All the instructions are grayed out and are not interactive, and the buttons for pausing and stopping the program are displayed. The instruction detail shows: The type of the instruction (e.g. *pick from feeder*), the parameters (e.g. object type) and transitions for success and failure.

The user is notified about the state of the system and the errors, as well as the currently available actions, using a notification bar shown next to the front edge of the table.

It is important for the user to know the state of the system, so for every detected object an outline and ID are displayed (see Fig. 3(d)). The type of the object is displayed upon clicking on the outline. For the purpose of setting the parameters, more information is shown, such as a polygon defining the area on the table, the outline of the object showing the position for object placement, etc. The same is also shown during the program execution, so the user knows in advance what object the robot will work with.

Various dialogs exist which allows the user to specify additional information. For instance, while programming an *pick from feeder* instruction, the user has to specify a pre-pose for object detection by manipulating the robot's arm and then confirming the position using a dialog. The pose is saved after pressing a button corresponding to the arm used (see Fig. 3(c)). The whole procedure is shown in Fig. 4 (a-e).

F. Known Limitations

The main input modality – touch foil – is prone to false readings when metal objects are placed on it, which makes it unsuitable for certain industrial settings. In the future, it might be replaced with or complemented by a vision-based approach (e.g. one from [8]). 3D interaction is currently limited to the kinesthetic teaching of positions, with no means for their later visualization.

V. Evaluation

In order to evaluate the proposed approach and to discover the main usability issues of the early prototype, a user experience testing was carried out[4]. Prior to the experiment itself, a pilot experiment with three subjects (faculty staff) took place, which helped us to verify the functionality of the prototype and to create the final experiment design.

As measures, we choose a combination of qualitative and quantitative data. Self-reported data were obtained using a questionnaire consisting of the System Usability Scale (SUS) [14], NASA Task Load Index (TLX) [15] in its raw form (simplified, with a scale in the range [1..7]) and a custom questionnaire focusing on the specifics of the system. We recorded the task completion times and the corresponding number of moderator interventions as quantitative data.

A. Experiment protocol

The experiment protocol consisted of four phases. None of the phases of the experiment was time-limited. There were one moderator and one operator in a separate room in charge of system monitoring, data recording, and WoZ (used solely to simulate user activity recognition).

1) Introduction: At the beginning of the experiment, the participants signed an informed consent form. They were told a story about a fictional SME producing wooden furniture: *"The company cannot afford a dedicated robot programmer, so it bought a collaborative robot programmable by any ordinary skilled worker. The robot will serve as an assistant preparing the parts for the workers who will do the assembly."* They were given information about safety, the parts of the workspace (interactive table, robot, feeders with furniture parts), and basic usage of the interface.

2) Training: The training phase consisted of three simple programs demonstrating the supported instructions. No specific product was assembled in this phase. The parameters of each program were first set by the participant and then the program was executed. During the execution, errors (e.g. a missing object) were intentionally invoked in order to gain familiarity with the error resolution dialog. In this phase, the moderator proactively helped the participants to complete the tasks and answered all the questions. A short practice of the think-aloud protocol followed. After that, the participants were told to set the parameters of those three programs independently while thinking aloud.

3) Main task: The assembly process of a target product (a small stool) was explained and the participants assembled it manually. Next, the structure of the corresponding program and the expected workflow were explained.

After the questions were answered, the participants started working. When finished, they started the program and collaborated with the robot on the task of producing a stool. Two stools were produced and the participants were told that there was a demand to adapt a product - to produce a higher stool. After the parameters of the program had been adapted, they produced one more.

4) Feedback: After finishing the tasks, an open discussion took place. The participants were asked for their impressions, additional questions, etc. Then, they were asked to fill in the questionnaire.

B. Stool assembly

The intended workflow of the main task is that the user does the assembly while the robot prepares the parts needed in the next steps "on background". The program is divided into three blocks (see Fig. 5). Blocks 1 and 2 have the same structure and serve to prepare the parts for the sides of the stool (two legs, two connecting parts, application of glue). The purpose of two blocks is that the user might set parts within one block to be supplied from e.g. the left feeder and in the other block from the right feeder. Block 3 serves to prepare the connecting parts for the final assembly of the sides of the stool.

C. Participants

In cooperation with an industrial partner (ABB Brno), six regular shop-floor workers of various ages, genders and technical backgrounds were selected (out of 27 volunteers) to take part in our study. These participants will be labeled as Participants A, B, C, D, E and F. Five of them work in quality control; one (E) works as a mechanic. The demographic data of the participants can be seen in Table I.

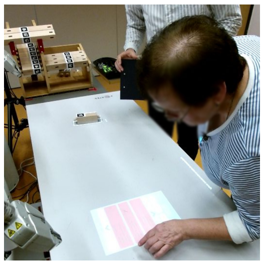
VI. Results

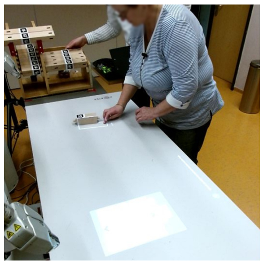
The section provides results of the experiment.

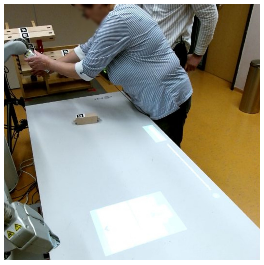
A. Qualitative and quantitative data

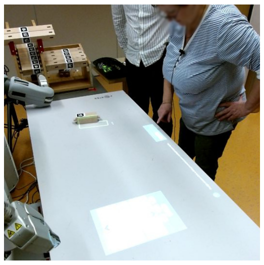
Table II shows the results per participant. The mean time to complete the main task was 2711 s (SD 620 s) with 11.7 (SD 6.7) moderator interventions. The main task consisted of setting the following instructions: 5x *pick from feeder* (2 parameters), 12x *place to pose* (1 parameter), 2x *apply glue* (4 parameters), resulting in settings of 30 parameters in total. The mean time for program adaptation task was 1053 s (SD 215 s). It consisted of setting: 2x *pick from feeder*, 2x *apply glue*, and optionally, adjustment of place poses (based on previously set poses), resulting in at least 12 parameters in total. These times include the delays caused by system errors (unreliable object detection, unstable manipulation pipeline, etc.). The mean SUS rating was 75.8 (SD 8.9), while for

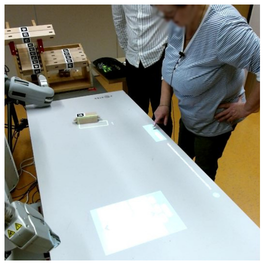
[4]Overview of the experiment: https://youtu.be/cQqNLy6mE8w.


(a) User selects instruction to be set from list (pick).

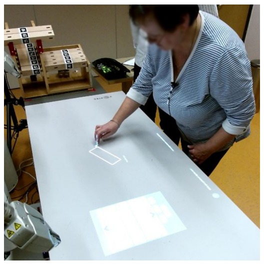

(b) Object type is set by touching its outline.

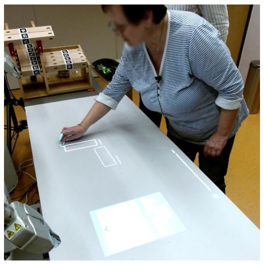

(c) Robot arm is used to teach detection position.

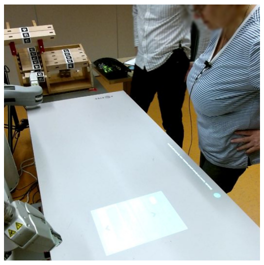

(d) Dialog shows if robot is able to detect object in feeder.


(e) User saves position (confirmation sound is played).

(f) User selects follow-up instruction (place).


(g) User adjusts place pose by dragging it on the table.


(h) Another pose, first one also shown for convenience.


(i) User tests *pick from feeder* instruction.

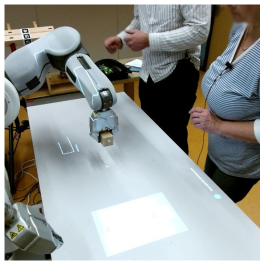

(j) Test of *place to pose* instruction.

Fig. 4. An example of human-robot interaction during the experiment. In this case, the user sets parameters for two *pick from feeder* instructions (one shown) and consequent *place to pose* instructions (both shown). Then, instructions are tested. Two input modalities are used: touch table and robot arm.

part.	**gender**	**age**	**education**	**experience with robots**	**attitude towards new technology**
A	F	57	vocational (technical)	none	skeptical
B	M	46	secondary (technical)	seen robot at least once	neutral
C	F	27	secondary (economics)	none	neutral
D	M	33	secondary (technical)	seen robot at least once	early adopter
E	M	24	secondary (technical)	works on workplace with robots but not next to them	neutral
F	M	34	undergraduate (technical)	none	skeptical

TABLE I
DEMOGRAPHIC DATA OF THE PARTICIPANTS.

comparison, the system from [4] was scored 66.75 (SD 16.95). The mean TLX was 33.3 (SD 8.8).

From the custom questions (see Table III) it seems that the participants in general liked interacting with the system and felt safe; however, they were confused from time to time. However, during the experiment, in most cases it was enough to tell them to check the notification area and they were able to continue afterwards.

B. Programming

Observation of the users has shown that the current visualization of the robot program is probably not sufficient, as it often took considerable time to realize what was currently being programmed, especially for the case of repeating sequences of program items (e.g. *pick from feeder*, *place to pose*, *pick from feeder*, *place to pose*). Not fully consistent terminology (e.g. program instruction was sometimes referred to as item and sometimes as step) may have contributed to this. Probably because of the similar appearance, for some participants it was difficult at the beginning to distinguish between a program block and a program instruction.

Probably the most common issue during programming was the participant forgetting to press the *Edit* button in order to switch from the view-only mode to the parameter settings mode for the selected instruction. The participants often tried to adjust for example place pose and were confused as to why it was impossible. Also, it was often unclear that it is only possible to execute individual instructions. Initially, two participants thought that the instructions (displayed in the program visualization) were for them, so they should perform e.g. *pick from feeder*. One participant asked if there are also assembly instructions for the workers.

There have been cases where the user accidentally changed the selected object type. Despite the fact that this was covered during training, some of the participants thought that the object type is selected when they put an object of that type on the table. It seems that although the objects of a selected type were highlighted differently (with a green outline), most of the participants only guessed what type was selected, or rather, checked it in the program visualization where the information was in textual form.

C. Individual instructions

1) Pick from feeder: Participants were often confused, as it was required to select the object type on the table and then to use a robot arm to set the pose enabling the detection of parts in the feeder. We noticed cases where the

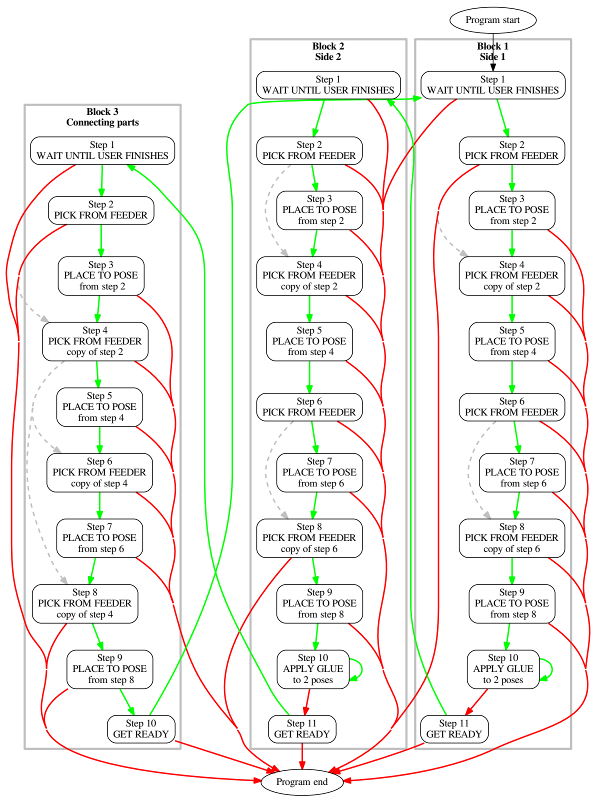


Fig. 5. Stool production program. The green edges represent *on_success* transition, while the red ones represent *on_failure*. The grey edges show dependencies. In the case of *apply glue*, there is a loop. The robot applies glue to one object in a specified area. If an object is found, the program flow continues to the *on_success* instruction - it tries to apply glue to another object. If there is no object without glue applied, the flow continues to *on_failure* (next instruction).

participant tried to select an object by knocking on it (instead of clicking on its outline), both on the object on the table and in the feeder. The participants commonly skipped the object selection, grabbed the robot arm and tried to set the pose, even above the object on the table, despite the fack that they were learning picking from feeder. After pressing *Edit*, dialog buttons for saving the arm pose (grayed-out at the time) were sometimes used “to select arm” before any other interaction. Most users took a new part from the feeder and put it on the table when they needed to select the object type even though there were already objects of that type that could have been used for this purpose. When adapting the program, it happened twice, that the participant by mistake set the position for the other feeder (e.g. the instruction originally used the left feeder, and they switched to the right one). This would mean that the robot would not be able later to place the object, as the following place pose (on the opposite side of the table) would be out of its reach.

2) Place to pose: Common sources of problems were unreachable place poses, or place poses too close to each other, which prevented the robot from placing parts successfully. The only possibility was to find out by trial and error. For all the participants, it was difficult initially to handle separated translation (by dragging) and rotation (using a pivot point). Some of them intuitively attempted to use multi-touch gestures (not supported by the interface thus far), including one participant who does not own any touch devices. Although the initial position of the place pose was in the middle of the table, some participants had trouble finding it, especially if there were many objects around. Some of them tried to drag the outline of a detected object or even placed an object into the outline of the place pose. Visualization of the place poses from other instructions (differentiated by a dotted line and a corresponding instruction number) were confused a few times with the current place pose and the users tried to move them.

For successful collaboration with the robot, it was necessary to organize the workspace so that the robot could prepare the parts for the next steps, while the user did the assembly. Only Participant B explicitly thought about organization of the workspace. The others had minor problems with it or required help. Participant C placed the parts in a very chaotic way. The participants were explicitly told during training that they may move widgets (e.g. program visualization) across the table; however, most of them did not use it and rather adjusted the place poses so that they did not collide with the widget.

3) Glue application: The most common issues were object type selection (attempts to select using the robot’s arm) and difficulties with the number of actually stored poses (shown textually). The fact that it is necessary to store required poses only with regard to the one object and the fact that the robot will do it in the same way for other objects in a given area was also generally unclear.

D. Program execution

During the program execution, errors occurred relatively often, especially when the robot tried to place an object; erroneous detection prevented it from doing so. In the event of an error, a dialog appeared and sound was played. Most issues were solved just by pressing the *Try again* button. The participants were explicitly told to pay attention to errors. Some of the participants reacted immediately, others after some time and one seemed to ignore the errors and had to be told to solve them. Once in a while it was necessary to warn a participant that he or she was blocking the robot by occupying part of the table where the robot was meant to place parts.

E. General findings

No one complained about imperfections of the projection (shadows, inaccurate registration), low readability of the text, interface response times, etc. Each participant had an issue at least once with a non-touchable margin of the interactive table, which was not indicated by the projected interface. There were also issues with pressing the buttons twice, where user tried, for example, to select an instruction which was immediately unselected. While inactive buttons were grayed

Measure	A	B	C	D	E	F
System Usability Scale	87.5	67.5	77.5	75.0	85.0	62.5
Simplified TLX	25.0	33.3	30.6	22.2	41.7	47.2
time to set program (s)	3849	3025	2618	2217	2661	1897
interventions	21	7	20	12	6	4
time to adapt program (s)	1088	1447	1118	958	738	968
interventions	11	4	12	2	2	2

TABLE II

QUALITATIVE MEASURES, TASK COMPLETION TIMES (STOOL PROGRAM) AND NUMBER OF MODERATOR INTERVENTIONS (INCLUDING ANSWERING QUESTIONS).

Statement	A	B	C	D	E	F
Collaboration was effective.	5	4	5	5	4	4
I felt safe.	4	5	5	5	5	5
Robot motions were uncomfortable.	2	1	1	1	1	1
It was easy to see what the robot was about to do.	4	5	5	4	4	2
The robot hindered me at work.	1	2	1	1	1	1
I watched every movement of the robot.	3	1	2	3	4	2
Learning the robot using its arm was intuitive.	4	4	5	5	5	4
Learning the robot using the interactive table was intuitive.	4	4	5	5	5	3
Interactive table shows all necessary information.	5	2	5	5	5	4
Sometimes I did not know what to do.	5	5	4	2	4	4

TABLE III

CUSTOM QUESTIONNAIRE, 1 - TOTALLY DISAGREE, 5 - TOTALLY AGREE

out, most users tried to press them anyway when they thought they should work.

With many objects on the table or during the stool assembly, there was considerable visual clutter. Interestingly, no one mentioned it. Difficulties with moving interface elements (e.g. place pose) across longer distances were observed, especially if there were many objects on the table. Again, no one complained or asked if there was an alternative method to dragging.

As a complementary modality, there were sounds (confirmation, warning, error). Only Participant B explicitly appreciated it.

Regarding safety, only Participant A once noted that a particular movement was probably not safe. No one used the emergency stop button.

VII. CONCLUSIONS

In this work, we targeted problems of the existing solutions in the area of interaction between the human workers and the industrial collaborative robots, particularly in the context of programming robots in SMEs. The proposed and tested interaction system is an attempt to reduce the mental demands and attention switching by centering all interaction elements in the shared workspace. This is achieved by the interactive SAR (combination of projection and a touch-enabled table) and kinesthetic teaching. Non-expert users program a robot on a high level of abstraction, and work within the task context, free of any additional external devices and with immediate visual feedback.

The conducted user experience tests proved the potential of our concept when all six regular shop-floor workers were able to program the robot to prepare parts for a stool assembly, to collaborate with the robot, and to adapt the program for an alternative product within a reasonable time.

During the experiment, no fundamental issues forcing us to reconsider the approach were found. However, the task state awareness in particular has to be improved as well as support for the workspace layout. The participants rated the system positively despite a number of minor usability issues and system errors caused by its experimental nature.

In addition to the revision of the interface to solve the usability issues, we plan to investigate multi-touch support, group operations, intelligent placement of user interface elements, and visualization of robot reachability.

ACKNOWLEDGMENTS

The work was supported by Czech Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project ”IT4Innovations excellence in science - LQ1602”.

REFERENCES

[1] E. M. Orendt, M. Fichtner, and D. Henrich, “Robot programming by non-experts: Intuitiveness and robustness of one-shot robot programming,” in *RO-MAN*. IEEE, 2016, pp. 192–199.

[2] M. Stenmark, M. Haage, and E. A. Topp, “Simplified programming of re-usable skills on a safe industrial robot: Prototype and evaluation,” in *HRI*. ACM, 2017, pp. 463–472.

[3] K. R. Guerin, C. Lea, C. Paxton, and G. D. Hager, “A framework for end-user instruction of a robot assistant for manufacturing,” in *ICRA*. IEEE, 2015, pp. 6167–6174.

[4] J. Huang and M. Cakmak, “Code3: A system for end-to-end programming of mobile manipulator robots for novices and experts,” in *HRI*. ACM, 2017, pp. 453–462.

[5] Y. S. Sefidgar, P. Agarwal, and M. Cakmak, “Situated tangible robot programming,” in *HRI*. ACM, 2017, pp. 473–482.

[6] S. Stadler, K. Kain *et al.*, “Augmented reality for industrial robot programmers: Workload analysis for task-based, augmented reality-supported robot control,” in *RO-MAN*. IEEE, 2016, pp. 179–184.

[7] E. Rosen, D. Whitney *et al.*, “Communicating robot arm motion intent through mixed reality head-mounted displays,” *arXiv preprint arXiv:1708.03655*, 2017.

[8] R. Xiao, S. Hudson, and C. Harrison, “Supporting responsive cohabitation between virtual interfaces and physical objects on everyday surfaces,” *HCI*, vol. 1, no. 1, p. 12, 2017.

[9] M. Funk, “Augmented reality at the workplace: a context-aware assistive system using in-situ projection,” 2016.

[10] F. Leutert, C. Herrmann, and K. Schilling, “A spatial augmented reality system for intuitive display of robotic data,” in *HRI*. IEEE Press, 2013, pp. 179–180.

[11] Z. Materna, M. Kapinus *et al.*, “Simplified industrial robot programming: Effects of errors on multimodal interaction in woz experiment,” in *RO-MAN*. IEEE, 2016, pp. 200–205.

[12] D. Wurhofer, V. Fuchsberger *et al.*, “Insights from user experience research in the factory: What to consider in interaction design,” in *Human Work Interaction Design. Work Analysis and Interaction Design Methods for Pervasive and Smart Workplaces*. Springer, 2015, pp. 39–56.

[13] S. Chitta, I. Sucan, and S. Cousins, “Moveit![ros topics],” *Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[14] J. Brooke *et al.*, “Sus-a quick and dirty usability scale,” *Usability evaluation in industry*, vol. 189, no. 194, pp. 4–7, 1996.

[15] S. G. Hart and L. E. Staveland, “Development of nasa-tlx (task load index): Results of empirical and theoretical research,” *Advances in psychology*, vol. 52, pp. 139–183, 1988.