Spatially Situated End-User Robot Programming in Augmented Reality

Michal Kapinus, Vítězslav Beran, Zdeněk Materna and Daniel Bambušek

***Abstract*— Nowadays, industrial robots are being programmed using proprietary tools developed by robot manufacturer. A skilled robot programmer is needed to create even as simple task as pick a well-known object and put it somewhere else. Contrary, in every-day life people are using end-user programming to make different electronic devices work in expected manner, without even noticing they are actually programming. We propose augmented reality-enabled end-user programming system allowing regular shop-floor workers to program industrial robotic tasks. The user interface prototype for this system was evaluated in the user study with 7 participants with respect to usability, mental workload and user experience.**

I. INTRODUCTION

For decades, robots have been deployed in automated manufacturing. Their use is mainly in large-scale production, because the design and construction of such automated production operation is very time-consuming and expensive. This solution pays off for a type of production that runs for at least several years. Although it is known that robotic systems are unsuitable for some operations, when it is sometimes almost impossible to replace humans, neither the method of production nor the technological advancements have made it possible to exploit the potential of humans in the production process together with robots.

Today, however, the rapid development of technology brings the ability to produce robots who are already able to work in the vicinity of humans, can to some extent perceive the world around and to some extent cooperate with human. There are new possibilities of making automated production more efficient by integrating human into the automated process. This is especially important for smaller productions. Nowadays, many manufacturing activities can be replaced by a cheaper robot or robotized production line, which is not able to handle all the tasks of the production process itself, but thanks to the possibility to cooperate with human, these tasks can be effectively solved. Smaller production in smaller manufacturing companies, however, brings a new problem: how to program these robots effectively for a new production process?

Many robot manufacturers now offer a variety of robot programming solutions, from desktop programming tools, where drag&drop and intuitive icons can be used to quickly program a manufacturing process (such as ABB RobotStudio or RoboDK), to approaches where a user directly defines the process by manipulating a robot (such as Baxter or YuMi). Today, many aspects affecting user workload or user experience when using UI devices to interact with the robots are known: the user often has to switch the context between workstation and computer, poor robot feedback worsens human-robot communication when visual or aural feedback is limited (light or sound signal, on-screen information, etc.). Although today's advanced technologies allow more natural human-machine interaction, most of the solutions are still based on a 50-year-old GUI concept. It implements a GUI using WIMP and 2D display on a desktop computer using real world metaphors.

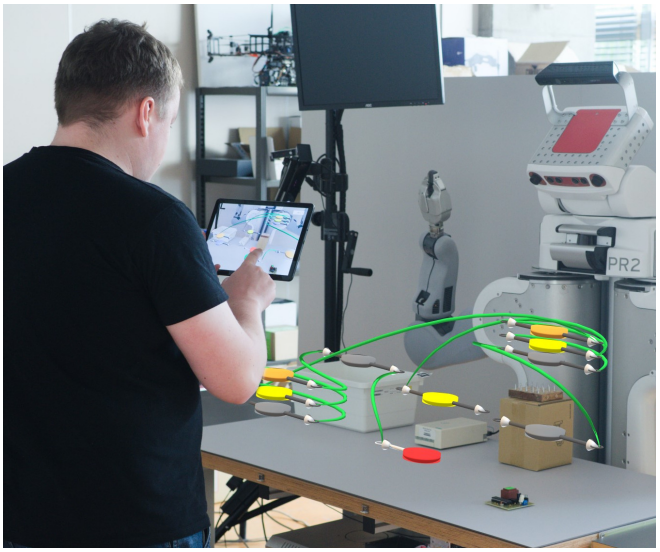

Fig. 1. ARCORO - Augment Reality for Connections and Operations of Real Objects.

The goal of our research is to return from a 2D desktop user interface back to a real 3D environment. In this work we present the application of this principle in the design of a new tool for robot programming. We define the requirements for a new type of user interface: integrate the programming tool into a real 3D environment, use scene and object knowledge to reduce user mental load, visualization of a program stage and robot's knowledge of the environment to improve user's feedback. We decided to address these challenges for the natural interaction of human with machine in real 3D environment. We design and introduce an innovative way of programming a spatial robotic task with high levels of abstraction using Augmented Reality (AR) technology (ARCORO[1]).

Based on our prior experience with simplified robot pro-

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: $ikapinus, beranv, imaterna, bambusekd@fit.vutbr.cz$

[1]Augment Reality for Connections and Operations of Real Objects

gramming in AR [1], an experimental ARCORO system, using a new concept of robot programming in 3D space using AR on a mobile device, was developed. Part of the work is also the definition of use-case, which is designed according to real demand from industry. This use-case was used to test the new ARCORO interface. The experiment was conducted with 7 participants and evaluated with respect to usability, mental workload and user experience.

II. RELATED WORK

An increasing number of collaborative robots in SMEs (small and medium enterprises) requires searching for new methods for end-user robot programming. Various techniques incorporating the AR were proposed, mostly based on visual programming [2], [3], programming by demonstration [4], [5] or combination of both [6], [1]. These methods may differ in both input and output modalities and utilizes the AR for both programming and giving visual feedback to the user.

Gadre et al. [3] proposed Mixed Reality (MR) system for robot programming using Head-Mounted Display (HMD). They compared this system against a 2D keyboard and mouse system for programming pick & place task. Gadre et al. [3] found that users were significantly faster and better able to successfully program the robot using the MR interface than the 2D interface.

Quintero et al. [7] designed AR system using Microsoft HoloLens HMD capable of 3D robot trajectory specification, virtual previews of robot motion and visualization of robot parameters. Blankemeyer et al. [8] presents another HMD-based system using Microsoft HoloLens for simple pick & place task programming.

Stadler et al. [9] discussed possibility of lowering mental demand of the robot programmer, by using tablet-based AR approach in simplified industrial tasks.

Magnenat et al. [10] have shown, that overall performance of the operator could be increased by incorporating AR and visual feedback into tablet-based system for robot programming system.

Recently, several solutions based on tabletop projections emerged. Materna et al. [1] have developed Spatial Augmented Reality (SAR) system using table with touch-enabled surface and projector above the table, projecting both User Interface to program collaborative robot and showing contextual information of objects on the table and the state of the system. Gao et al. [11] provided another tabletop SAR solution for industrial end-user robot programming of manipulation tasks, using common hand gestures detected by computer vision techniques.

To investigate effects of presenting robots intentions to the human, Bunz et al. [12] conducted experiment involving mobile robot with projector mounted on top of it, projecting various patterns indicating its intended movement.

Head-up displays and projected user interfaces benefit from freeing operator's hands, which enables direct manipulation with real objects. On the other hand, contemporary head-up displays such as Microsoft HoloLens and others, suffer from narrow field of view and potential user's discomfort in long-term usage. Moreover, end-user programming systems based on hand-held AR overcomes head-up based systems in terms of speed and user experience [13]. While projected interfaces does not suffer from these issues, they are not currently able to present information in free 3D space, and therefore only suitable for tabletop scenarios [1].

The AR systems often benefit from knowledge of the environment and therefore offer new possibilities in end-user programming. The spatial situated programming incorporates real objects into programming process. For instance, Ivy [14] enables user to link different smart devices, create automated behaviour based on readings from smart sensors and visualize data flows between those devices. Reality editor [15] is another example of spatial situated programming, enabling programming of behaviour and interactions of smart objects, using hand-held AR device.

In our proposed approach, the tablet-based AR is combined with semantic information of the objects on the table, to enable regular shop-floor workers to create robotic programs. Contrary to some aforementioned solutions [11], [7], [9], [1], our system aims to both defining the flow of the program and setting its parameters. By using relatively cheap mobile device, the cost of the solution can be significantly lowered comparing to approaches using high-end HMD devices [3], [8] while still remain more flexible than projection based solutions [1], [11].

III. PROPOSED APPROACH

When designing a novel user interface concept for robot programming, we first defined the following issues of current solutions:

- Mental mapping of robot instructions to the physical place in the environment
- Context switching between programming device (e.g. computer) and the workspace
- Low abstraction of the robot instructions, relations between the instructions, conditions and parallel execution.

A. Process-based vs. Object-based approach

The goal of the programmer is to prepare a list of steps that describe: in what order the robot should perform various actions, with what objects the action should be performed, how and under what conditions the action should be performed. The result is a sequence of actions - a program. In principle, this programming task can be implemented in two ways.

Process-based method of robot programming takes advantage of the so-called top-down approach. It describes the whole process with inputs and outputs and then it continues in dividing the process into several sub-processes until it gets to the low-level problems. On the other hand, object-based method describes functionality of different low-level objects and allows to use them to build a working system piece by piece. This is also known as a bottom-up approach.

We used the real world metaphor, when we describe the manufacturing process, we usually:

- first we describe the environment: components, devices, tools and objects that are in the scene and what they do or how to use them for the task,
- then we begin to describe the process step by step, including the links between the environment objects, their specific settings, and the expected outputs,
- Finally, we summarize the expected outputs and risk parts

Based on this observation, we decided to follow an object-based approach proposing concept using spatially-aware augmented reality on mobile device.

B. Program representation

Most of the basic operations of a robotic task are related to a particular place in 3D space, either by relation to a real or virtual object, or directly to an absolute position in the scene. The program representation in our concept was inspired by flowcharts in 3D. Discrete operations (e.g. pick the object, execute operation, etc.) are represented by nodes. Each node is spatially adjacent to the position, where the action takes place. For example node representing operation *Place object to the box* is located above the intended box. This adjacency helps the user with mental mapping of the instructions to the physical space. Nodes hold information needed for their execution, e.g. type of the object which should be manipulated, position on the table, where object should be placed, etc. In addition to setting individual operation parameters, linking these nodes to create a program flow is a key challenge for usable user interface.

Each node has inputs and outputs. By connecting the inputs and outputs of various nodes, user can define the flow of the program. By connecting one output to multiple inputs, the user can specify conditional transition or parallel execution. Based on the parameters of connected nodes, the actual executed path is derived. On the Fig. 2, parallel execution of the program is defined. Workpieces of all the *nodes* are the same, i.e. once the *Execute testing* operations is done, both *Execute printing* and *Pick from tester* operations will be executed in parallel. This approach is valid only when these parallel operations don't physically manipulate the workpiece. In this example, the workpiece will be picked by robot using the *Pick from tester* operation and corresponding label will be printed at the same time. This label will be stuck to the workpiece later in the program.

Conditional execution can be seen on the Fig. 3. The *Pick from table* node has set two different workpieces, e.g. red ball and green cube, meaning one of them will be picked. From this step, two different *Place* operation can occur, each with one of the aforementioned object set as a workpiece. Based on the picked object from step *Pick from table*, corresponding *Place* operation is selected.

C. Spatially situated programming in AR

Spatially situated programming is useful in scenarios, where spatial context is important, like: robot manipulating workpieces, picking them from conveyor belt, putting them inside the pressing machine, etc. We propose the system, which is aware of semantic properties of the objects in the environment: knowledge that some object can be picked up, that a box offers inserting of some object, etc. The user can benefit from that shared knowledge of the environment and by using these information, the user can define desired actions more effectively.

The visual elements of the system are presented to the operator using the augmented reality, either in head-up display or using hand-held mobile device.

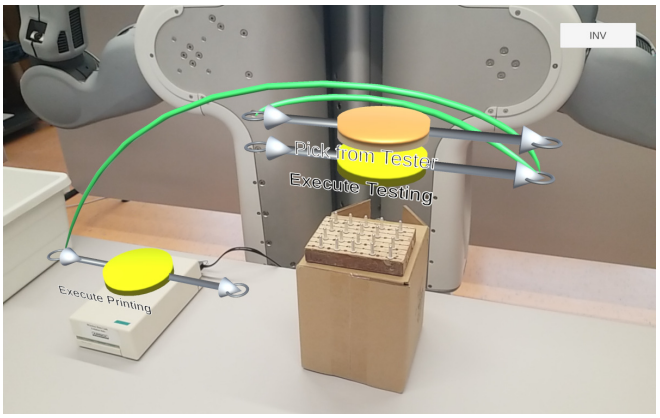

Fig. 2. Parallel execution of the program. The *Execute testing* node is connected with *Pick from tester* node and *Execute printing* node. All of the *nodes* have set the same workpiece, so during the runtime, both paths will be executed at once.

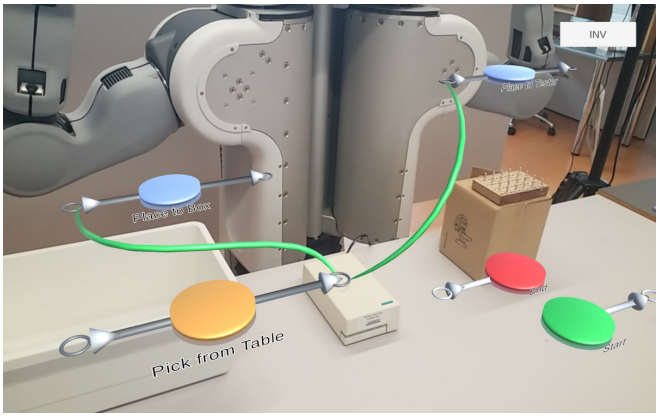

Fig. 3. Conditional execution of the program. There can be seen *Pick from Table* node, connected with two *Place* nodes. The left and right *nodes* have set different workpiece. The actual flow of the program is decided during the runtime, based on type of the workpiece picked in the *Pick from Table* node.

IV. Prototype of the User Interface

To evaluate the proposed approach, we developed the prototype of the user interface, using hand-held mobile device. In cooperation with our industrial partner, we have selected a specific industrial use-case.

A. Use-case

The selected use-case represents the process of testing the printed circuit board (PCB). The PCB has to be inserted into

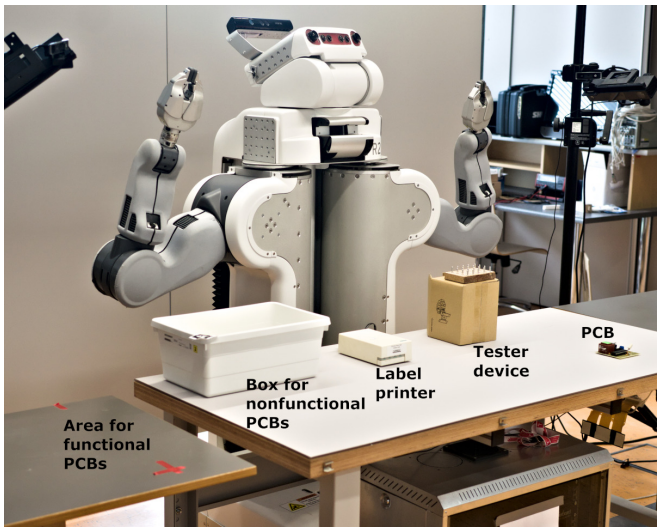

Fig. 4. Testbed used during the experiment. On the table, you can see, from right to left, example PCB, a mockup of the tester device, a printer of the labels, a box for disposing nonfunctional PCBs and another table for functional PCBs.

testing device (a.k.a. tester) and based on the test result either disposed or forward to the next stage of processing. Besides, the corresponding label should be printed and stuck to both functional and nonfunctional PCB. A mockup of the testing facility was prepared, as can be seen on Fig. 4.

The mockup environment consists of the table with the PCB, the testing device, the printer and the box for nonfunctional PCBs. Next to the main table, the other table intended for functional PCBs is placed. To improve the feeling of near future robotic facility, the PR2 robot was placed behind the table. The whole procedure of the use case looks like this:

1) Pick the PCB from the table
2) Place the PCB inside the tester device
3) Execute testing
4) Do in parallel ...
   a) Pick the PCB from the tester device
   b) Print corresponding label
5) Place the PCB on the table
6) Stick the label to the PCB
7) Pick the PCB
8) Place the PCB to ...
   a) the box OR
   b) the other table

Steps 4 represents parallel execution of two operations at the same time, as the robot is picking the PCB from the tester and simultaneously the printer is printing the label. The step 8 represents conditional transition, as the PCB is placed either to the box or to the other table based on the result of the testing process.

B. System

The prototype of the user interface was created using Unity3D game engine. To register motion of the mobile device and track its position in the real world, the ARCore framework was utilized. The prototype was developed for an Android-driven handheld mobile device. Display of the device shows the video stream from the back-facing camera with superimposed user interface.

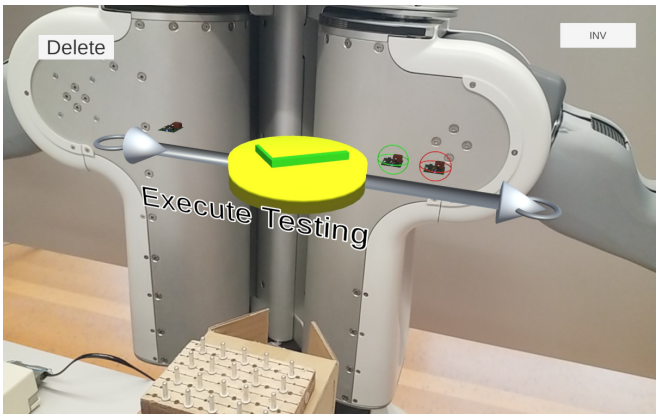

Fig. 5. The *puck* consists of central disc, two circles representing the input and the output and two pipes connecting input/output with the disc. In the first prototype, the type of the *puck* is represented by its color and the text placed in front of the disk. Above the pipes, small 3D models of input and output workpieces are placed.

Using the Unity3D, the virtual scene was created (see Fig. 7), spatially identical to the real scene described above. The virtual and real scenes are mutually calibrated using the AR marker placed in the lower left corner of the table. This calibration needs to be done once during the application startup.

The system simulates knowledge of the environment and context of all objects and devices on the table. We placed invisible virtual bounding box around each physical object on the table, so user can interact with them by touching them on the screen.

C. User interface elements

Several UI elements were designed for the prototype to allow user to interact with the system. These elements are either 2D or 3D. In this prototype, all elements representing different operations, their connections etc. are static and prepared for selected use-case, as can be seen on Fig. 7.

1) Operations: In our prototype, each operation is represented by so-called *puck* (see Fig. 5). The *puck* consists of central disc, two circles representing the input and the output and two pipes connecting input/output with the disc. The input is placed on the left of the *puck* (with inside the puck aiming arrowhead), output is placed on the right of the puck (with arrowhead aiming outside of the *puck*).

The *puck* serves as a visualization of operation and its parameters, and at the same time, as a main input point for the operator. To change any operation's parameter, the user has to select desired operation first. To enable this, the so-called edit mode was designed. User can switch between the normal and edit mode by clicking on the *puck*. While in edit mode, only the edited and directly connected *pucks* are visible to the user and the others are hidden to lower

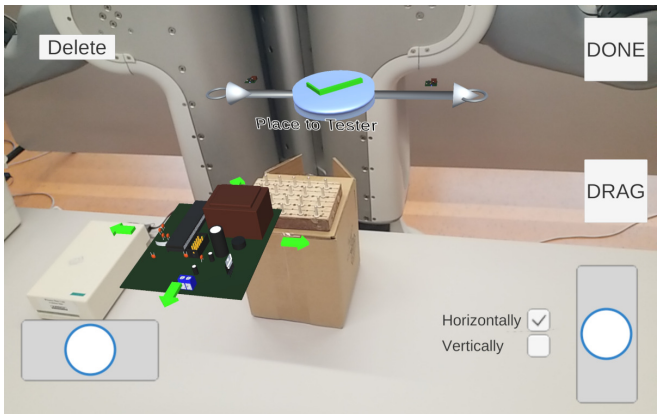

Fig. 6. Teleoperating user interface for navigating the 3D model of the PCB to the tester device. There are two joysticks on the bottom left and bottom right side of the tablet. Next to right joystick, there are two buttons for controlling whether the PCB should move in horizontal or vertical plane. Above the right joystick, there is a *DRAG* button. When it is held, the 3D model moves in the same direction and speed as the tablet.

visual clutter. In the edit mode, parameters of the operation are visible.

Most of the operations only manipulates the workpiece without changing it, i.e. the workpiece on the input is the same as the workpiece on the output of the *puck*. There are two exceptions in our use-case: *Pick from table* and *Execute testing*. The former has no workpiece on the input, because it is the first operation in our program. The picked object is automatically set as a workpiece for the output and it is added to the inventory (will be discussed later). The *Execute testing* operation works as follows. When the PCB is set as a input workpiece, it automatically creates two new workpiece types: *PCB_OK* and *PCB_NOK*. The former means *tested and OK* (functional) and the latter means *tested and not OK* (nonfunctional). These two workpiece types are also added to the virtual inventory.

2) Connections: The *pucks* themselves are not sufficient to define flow of the program, as they only define operations, but not the order in which they shall be executed. To define the flow, the operator can create connections between the *pucks*, by connecting the output of one *puck* and input of other *puck*. This connection is represented by a green spline between these two *pucks*. To make it easier for the user, once he clicks on the output of one *puck*, a big blue plus appears on the input of all other *pucks* and vice versa. By clicking on this plus, the connection is created.

In case of incorrectly created connection, the operator can use a big red cross to remove said connection. This cross is visible only for connections adjacent to currently edited *puck*. There can be several connection attached to one output or input allowing user to define conditions and parallel execution.

3) Interactive objects and context menus: To define an operation for any physical object on the table (e.g. printer, tester, etc.), the operator has to create appropriate *puck* by the object. As the system benefits from the semantic information about objects in the scene, context menu with each possible operations for the objects could be generated. By clicking on any object, this menu emerges, allowing user to define desired operation. This was enabled by creating a clickable invisible bounding box around each object in the virtual scene (semi-transparent boxes on the Fig. 7).

4) Inventory and teleoperating UI: While user composes the program, each workpiece he use in the program (e.g. PCB which shall be picked) appears in the inventory list. By clicking on the workpiece image in the inventory while in edit mode of some *puck*, user can set this object as a workpiece for this *puck*.

The operation "Place to tester" needs to specify 3D position of the workpiece while placing inside the testing device. To do so, a teleoperating user interface is prepared, allowing user to move with 3D model of the workpiece. There are two different approaches to control the position of the desk. The user can adjust the position in vertical or horizontal plane using two joysticks, placed on both side of the screen (see Fig. 6). The other way to set the position is by using so called *DRAG* button. When pressed, the desk moves in the same direction and speed as the tablet, so the operator can literally drag the desk by moving with the tablet (see Fig. 6).

D. User interaction

The screen of the mobile device is used for both visualization of the process and as a main input for the operator. The application on the device knows position and semantic information of all objects in scene (hard-coded for the prototype). Using the ARCore framework, the mobile device knows its position and orientation in the space, which enables the operator to interact with real objects by clicking on their 2D image on the screen.

V. EVALUATION

We provide qualitative results obtained from the user study with 7 participants. To evaluate our proposed approach, we created the prototype of the user interface using ARCore-enabled mobile device.

A. Experimental Procedure

Experimental protocol consisted of 4 phases: orientation, training, programming, discussion.

1) Orientation: During the orientation, the moderator introduced the evaluated system to the participant. He or she then signed an informed consent form.

2) Training: In the second phase, the participant learned how to use the mobile device to create robot instructions (a.k.a. *pucks*), how to set parameters of the instructions and how to connect them to create intended program. During the second phase, the moderator proactively helped the participant to complete the tasks and answered all questions.

3) Main task: The main task was presented to the participant. He or she was asked to program the robot to pick the PCB from the table, place it to the testing device, execute the testing process, print and stick correct label based on the result of the testing process and then place the PCB either

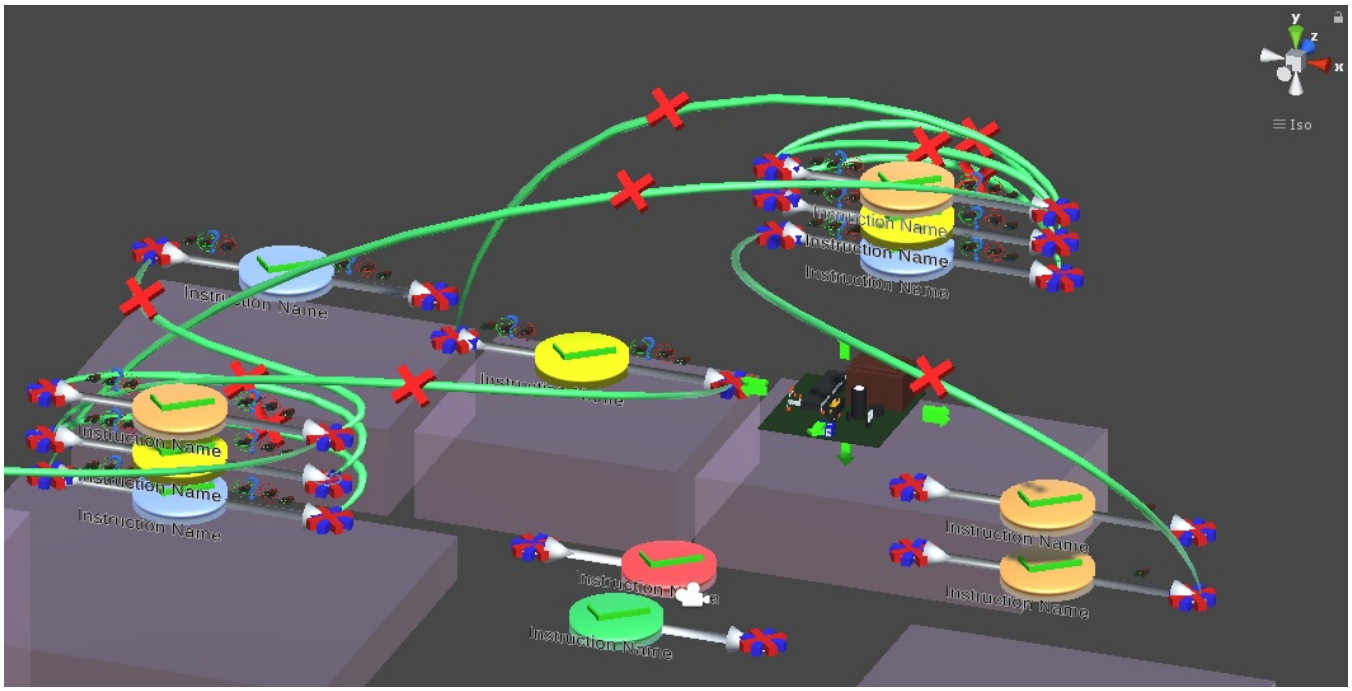

Fig. 7. Unity scene of the prototype UI. Semitransparent boxes define interactive places, which user can use to define intended operation. Above each of these boxes, there are *pucks* of various colors, representing different operations. They are connected with green splines, representing flow of the program (for the sake of clarity, only subset of possible connections are displayed in this figure).

to the box or on the other table (again, based on the result of the test).

After the task was presented to the participant, he or she began to work on the task by him or her self. The moderator was available to answer additional question or to help in case of problems with the prototype, but did not actively step into the programming process. Each participant worked until he or she claimed that the task is done. Moderator than reviewed the created program and either confirmed the correctness or suggested to the participant what should be altered.

4) Discussion: After completing the main task, the participant filled out the questionnaire. Besides, participants were asked for their thoughts of the system, additional questions, etc.

B. Sensors and collected data

The whole process of the experiment was recorded on several cameras. One of them was placed on participant's forehead, aiming to mobile device in participant's hands, other one was aiming towards the participant and two more cameras were recording the workspace. The screen of the mobile device was also recorded, together with indication of participant's input. To record voice of both moderator and participant, lavalier microphones were used.

C. Participants

There were 7 participants of various ages and genders, all of them with none or very limited knowledge of programming and augmented reality. These participants will be labeled as Participant A, B, C, D, E, F and G. Table I shows the demographic data of the participants.

VI. RESULTS AND FINDINGS

The section provides measured results and observed findings of the experiment. The main goal of the presented experiment was to prove, that non-expert users are able to program the selected use-case, using the ARCORO system. We focused mainly on usability issues, mental workload of the participants and the user experience.

A. Qualitative and quantitative data

As a metric for the system usability, the SUS[2] [17] method was chosen. To evaluate SUS score for our system, each participant had to score 10 items with one of five responses that range from Strongly Agree to Strongly disagree. Table II shows the SUS score for each participant individually, the mean SUS score from all participants was 82.86 (SD=9.29). According to Sauro-Lewis curved grading scale [17], SUS score in range of 80.8–84.0 is rated by grade **A**, and is at the 90–95th percentile. This shows promising potential for future research in this field, and shows, that the created prototype user interface is highly usable.

To measure the mental workload of the participants, simplified NASA-TLX[3] method was utilized. The mental workload can negatively affect the performance of the operator, therefore is important to measure this attribute from the earliest phases of prototyping. Although the mental workload in laboratory scenarios cannot be generalized directly to the workload in real environment, it still can be useful to reveal potential issues. The mean TLX in our experiment was 27.38

[2]System Usability Scale
[3]NASA Task Load Index

Participant	Age	Gender	Education	Experience with augmented reality	Experience with programming	Attitude towards new technology
A	24	F	bachelor degree	little	little	late majority
B	24	F	bachelor degree	some	little	early majority
C	34	M	master degree	none	none	early adopter
D	22	M	secondary	little	little	late majority
E	21	M	secondary	little	little	early adopter
F	24	F	bachelor degree	little	none	early adopter
G	33	M	secondary	little	little	early majority

TABLE I

DEMOGRAPHIC DATA OF THE PARTICIPANTS. THE SCALE FOR BOTH EXPERIENCE-RELATED QUESTIONS WERE NONE, LITTLE, SOME, QUITE A LOT, MANY. THE ATTITUDE TOWARDS NEW TECHNOLOGY SCALE IS BASED ON ROGERS [16] DIFFUSION OF INNOVATIONS.

Participant	SUS	NASA TLX	UEQ ATT	UEQ PRA	UEQ HED	time to set (s)
A	95.00	25.00	2.67	2.50	2.12	535
B	80.00	25.00	2.00	2.42	0.75	427
C	67.50	47.22	1.17	2.00	1.75	460
D	85.00	27.78	1.67	2.25	2.25	507
E	92.50	27.78	2.67	2.75	2.88	431
F	82.50	19.44	2.00	2.08	2.38	521
G	77.50	19.44	1.33	1.83	0.88	806

TABLE II

DETAILED RESULTS OF ALL MEASURED RESULTS FOR EACH PARTICIPANT.

(SD=9.41), which means that the workload was lower then in at least 80% of studies analyzed by Grier [18].

For any interactive system to be successful, a high-quality user experience is the key. Among several methods to measure the user experience, we selected the UEQ[4], because of its simplicity for both participant and evaluator and reliable results. The system was overall rated as *Excellent* in all UEQ categories, i.e. *Attractiveness* (mean score 1.93, SD=0.58), *Pragmatic* attributes (mean score 2.26, SD=0.28) and *Hedonic* attributes (mean score 1.86, SD=0.72). All categories were evaluated using the standard UEQ benchmark [19].

The mean time for the main task completion was 527 seconds (SD=130s). The main task consisted of settings following operations and their parameters and of creating connections between them: 3x *pick object*, 4x *place object*, 3x *execute (testing, printing and sticking)*. For each operation, workpiece had to be set. Moreover, for one of the *place object* operations, an exact position of the PCB inserted to the tester had to be set. The completion time excludes delays caused by prototype errors.

B. General findings

During the experiment, we found no fundamental problem forcing us to reconsider the proposed approach. Although minor issues were observed or self-reported by participants, all participants were able to complete the task.

[4]User Experience Questionnaire

All participants reported, that the *pucks* (representing operations) were unnecessary large. In cases when there were more *pucks* above the same object, for instance *place object to tester, execute testing and pick object from tester*, the state and parameters of those *pucks* were unclear and it was hard to recognize mutual connections. To avoid this, design of *pucks* needs to be refined and better strategy of *pucks* placement should be adopted in further versions.

The participants were instructed to inform the moderator once they though they have successfully finished the programming. Most of the created programs contained one or more errors, which would lead to failure during execution. The participant C explicitly reported, that he is unable to check if the program is correct. The participant A in the end went through all created *pucks* to check whether all parameters are correctly set and connections between *pucks* are as intended.

After the errors were pointed out by the moderator, each participant was able to correct the error and to successfully finish the task. This has shown, that debugging system has to be improved and better system state indicators should be involved. To support users awareness of the program correctness, the program flow visualization needs to be improved.

Only two of the participants found out, that they can benefit from active movement of the mobile device inside the scene, to achieve higher accuracy when clicking on interface components. Most of them were just standing in certain distance from the table and using only vertical rotation in cases when FOV of the tablet was too narrow. The participant B stated, that it was more comfortable for her to just stand at one place to observe the whole situation and that she would appreciate the possibility of zooming the scene on the screen to avoid miss-clicks.

The usual procedure for most of the participants consisted of creating the *puck*, followed by creating the connection between said *puck* and previously created *puck*, repeated until the whole program was created. The participant A followed a different approach. At first she created most of the *pucks* to label all desired operations and once she was satisfied with *pucks*, she started to create connections between them.

Participants A, B, C and E were using only one hand to control both joysticks (placed on different side of the

screen) while the rest of the participants were using both hands, as was intended when designing the user interface. The participant A was the only one to use a *DRAG* button, to set the initial position of the desk, followed by refining the final position using the joysticks.

Although minor issues were observed during the experiment, all of the participants rated the system positively. The participants agreed that the system is easy to use and requires no special knowledge from the operator.

VII. CONCLUSIONS

The aim of this work is to reflect current needs in the area of programming robots in low and medium complex tasks in a shared collaborative environment. We have designed a new concept of robot programming using augmented reality on a mobile device. The main goals pursued in the design of the new concept were: eliminating the need to switch user context between desktop and work environment by mapping instructions directly into a real 3D environment, reducing user mental stress by using semantic information about real objects and increasing the abstraction of instructions and their relations.

We have defined a simple use-case that is inspired by the real demands from the industry. In the experiment, we observed mainly usability of designed UI, workload of user and user experience with designed spatial programming concept. We have evaluated with 7 users which has shown that, despite some shortcomings discussed, this is the direction that can be taken. All participants were able to perform all the tasks independently after a short training. All participants evaluated the usability of the interface mostly positively.

Positive adoption of the new concept can also be attributed to the use of equipment that most users are used to working with. In the future, we want to verify this unambiguity and compare the usability of the concept with other, yet less common devices, such as HoloLens glasses. In the next research, we will also focus on improving the orientation in the programmed task, solving the UX deficiencies found in this study, and integrating the UI into a real robotic system.

ACKNOWLEDGMENT

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] Z. Materna, M. Kapinus, V. Beran, P. Smrž, and P. Zemčík, "Interactive spatial augmented reality in collaborative robot programming: User experience evaluation," in *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, Aug 2018, pp. 80–87.

[2] C. Mateo, A. Brunete, E. Gambao, and M. Hernando, "Hammer: An android based application for end-user industrial robot programming," in *2014 IEEE/ASME 10th International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, Sep. 2014, pp. 1–6.

[3] S. Yitzhak Gadre, E. Rosen, G. Chien, E. Phillips, S. Tellex, and G. Konidaris, "End-user robot programming using mixed reality," 10 2018.

[4] J. Aleotti, G. Micconi, and S. Caselli, "Object interaction and task programming by demonstration in visuo-haptic augmented reality," *Multimedia Systems*, vol. 22, no. 6, pp. 675–691, Nov 2016. [Online]. Available: https://doi.org/10.1007/s00530-015-0488-z

[5] P.-C. Li and C.-H. Chu, "Augmented reality based robot path planning for programming by demonstration," 12 2016.

[6] 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.

[7] C. P. Quintero, S. Li, M. K. Pan, W. P. Chan, H. F. Machiel Van der Loos, and E. Croft, "Robot programming through augmented trajectories in augmented reality," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018, pp. 1838–1844.

[8] S. Blankemeyer, R. Wiemann, L. Posniak, C. Pregizer, and A. Raatz, "Intuitive robot programming using augmented reality," *Procedia CIRP*, vol. 76, pp. 155–160, 01 2018.

[9] S. Stadler, K. Kain, M. Giuliani, N. Mirnig, G. Stollnberger, and M. Tscheligi, "Augmented reality for industrial robot programmers: Workload analysis for task-based, augmented reality-supported robot control," in *Robot and Human Interactive Communication (RO-MAN), 2016 25th IEEE International Symposium on*. IEEE, 2016, pp. 179–184.

[10] S. Magnenat, M. Ben-Ari, S. Klinger, and R. W. Sumner, "Enhancing robot programming with visual feedback and augmented reality," in *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 2015, pp. 153–158.

[11] Y. Gao and C.-M. Huang, "Pati: A projection-based augmented table-top interface for robot programming," in *Proceedings of the 24th International Conference on Intelligent User Interfaces*, ser. IUI '19. New York, NY, USA: ACM, 2019, pp. 345–355. [Online]. Available: http://doi.acm.org/10.1145/3301275.3302326

[12] E. Bunz, R. T. Chadalavada, H. Andreasson, R. Krug, M. Schindler, and A. Lilienthal, "Spatial augmented reality and eye tracking for evaluating human robot interaction," in *RO-MAN 2016 Workshop: Workshop on Communicating Intentions in Human-Robot Interaction, New York, USA, Aug 31, 2016*, 2016.

[13] N. Dass, J. Kim, S. Ford, S. Agarwal, and D. H. P. Chau, "Augmenting coding: Augmented reality for learning programming," in *Proceedings of the Sixth International Symposium of Chinese CHI*, ser. ChineseCHI '18. New York, NY, USA: ACM, 2018, pp. 156–159. [Online]. Available: http://doi.acm.org/10.1145/3202667.3202695

[14] B. Ens, F. Anderson, T. Grossman, M. Annett, P. Irani, and G. Fitzmaurice, "Ivy: Exploring spatially situated visual programming for authoring and understanding intelligent environments," in *Proceedings of the 43rd Graphics Interface Conference*, ser. GI '17. School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada: Canadian Human-Computer Communications Society, 2017, pp. 156–162. [Online]. Available: https://doi.org/10.20380/GI2017.20

[15] V. Heun, J. Hobin, and P. Maes, "Reality editor: Programming smarter objects," in *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*, ser. UbiComp '13 Adjunct. New York, NY, USA: ACM, 2013, pp. 307–310. [Online]. Available: http://doi.acm.org/10.1145/2494091.2494185

[16] E. Rogers, *Diffusion of innovations*. Free Press of Glencoe, 1962. [Online]. Available: https://books.google.cz/books?id=zw0-AAAAIAAJ

[17] J. Sauro and J. R. Lewis, *Quantifying the User Experience: Practical Statistics for User Research*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2012.

[18] R. A. Grier, "How high is high? a meta-analysis of nasa-tlx global workload scores," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 59, no. 1, pp. 1727–1731, 2015. [Online]. Available: https://doi.org/10.1177/1541931215591373

[19] M. Schrepp, A. Hinderks, and J. Thomaschewski, "Construction of a benchmark for the user experience questionnaire (ueq)," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 4, pp. 40–44, 06 2017.