Online and Offline Robot Programming via Augmented Reality Workspaces

Yong Joon Thoo, Jérémy Maceiras, Philip Abbet, Mattia Racca, Hakan Girgin and Sylvain Calinon

Abstract— Robot programming methods for industrial robots are time consuming and often require operators to have knowledge in robotics and programming. To reduce costs associated with reprogramming, various interfaces using augmented reality have recently been proposed to provide users with more intuitive means of controlling robots in real-time and programming them without having to code. However, most solutions require the operator to be close to the real robot's workspace which implies either removing it from the production line or shutting down the whole production line due to safety hazards. We propose a novel augmented reality interface providing the users with the ability to model a virtual representation of a workspace which can be saved and reused to program new tasks or adapt old ones without having to be co-located with the real robot. Similar to previous interfaces, the operators then have the ability to program robot tasks or control the robot in real-time by manipulating a virtual robot. We evaluate the intuitiveness and usability of the proposed interface with a user study where 18 participants programmed a robot manipulator for a disassembly task.

I. INTRODUCTION

Industrial robots play a vital role in manufacturing tasks such as welding and assembly due to their ability to perform these tasks with a high degree of precision and reliability. Nevertheless, despite their importance in manufacturing, the programming of such robots remains a costly and time-consuming task often requiring operators to have a certain degree of knowledge of robotics and programming [1], [2], [3], [4], [5], [6].

The most popular methods for programming industrial robots are referred to as *online* and *offline*. The *online* method requires the use of the real physical robot and consists of recording the path designed by an operator by controlling the joints of the robot via a teaching pendant with a joystick and/or keypad. The *offline* method on the other hand does not require the actual robot and consists of constructing a virtual representation of the robot and its workspace to simulate tasks prior to applying it on the real robots [7], [8]. Various simulators and 3D engines have recently been extended to support robotics platforms, making the *offline* programming of robots possible.

These methods present however multiple disadvantages. For *online* programming, the control of a robot via a teaching pendant often requires the robot in question to be removed

The authors are with the Idiap Research Institute, Martigny, Switzerland (name.surname@idiap.ch).
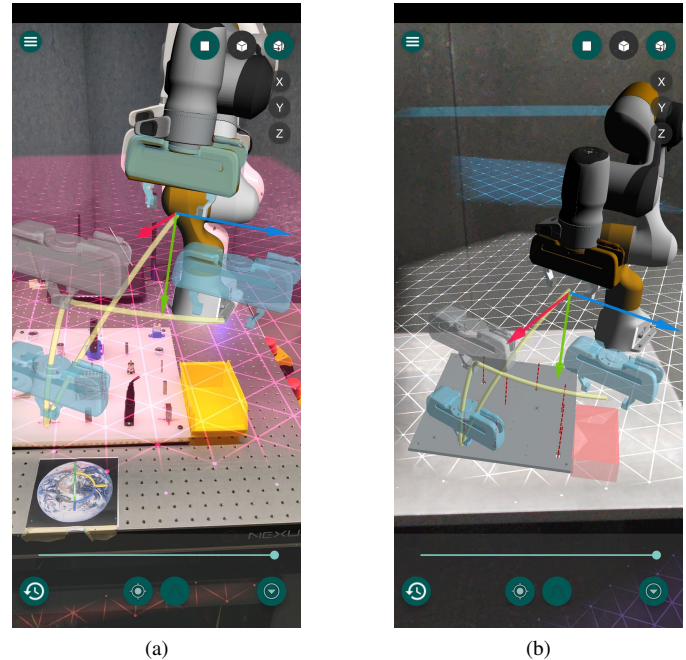
Fig. 1: Illustration of the ability to program the robot *offline* either (a) directly in the real robot's workspace or (b) in a virtual workspace by defining keypoints (displayed as transparent end-effectors) and visualising the resulting trajectory.

from production during programming or, under certain circumstances, a shutdown of the production line [2]. In the case of *offline* programming, the reconstruction of the robot's workspace in a virtual environment can be time consuming. Furthermore, the transition from the simulation to the execution on the real robot may suffer from inaccuracies, with additional costs required to overcome them [7]. Moreover, offline methods require a certain degree of familiarity with programming and simulators.

The significance of these disadvantages increases when robots need to be reprogrammed frequently. This is especially true in High-Mix Low-Volume (HMLV) manufacturing, where there is a large variety of items to be produced in small quantities [7]. As such, research has been conducted to design intuitive and efficient programming interfaces for users with little to no experience in the fields of robotics or programming [1].

To suit these needs, Augmented Reality (AR) technology has been proposed as a solution due to its ability to su-

perimpose information in various forms onto the real world. Indeed, AR has been growing increasingly popular in recent years and has been used in various fields such as architecture, construction, education, manufacturing and engineering [9]. More specifically, in the field of robotics, AR interfaces provide a new medium for interaction with the robot and enable the exchange of information during tasks [10] in addition to providing users with the ability to preview a robot's intended actions.

As such, several augmented or mixed reality interfaces have been proposed to enable users not only to control a robot [6] but also to program it by defining trajectories and tasks via the manipulation of a virtual robot in the interface [5], [8], [11], [12], [13]. These interfaces provide the operators with an intuitive way of programming the robot without an extensive knowledge about programming and combines some of the advantages of current *online* and *offline* methods by enabling users to simulate a task directly in the robot's workspace, making the transition from simulation to real execution easier. However, the proposed solutions require the users to share the same workspace as the robot such that the robot in question cannot be used for production during the programming process.

To further reduce the time during which the robots are unavailable for production, we propose an AR interface providing users with the ability to program the robot from a virtual representation of the workspace which can be built and saved by the user such that tasks can be programmed from different locations (Fig. 1b). Adjustments to the trajectory, if necessary, could then be made by displaying the designed task in the real robot's workspace resulting in less time during which the robot in question would be unavailable for production.

In Section II, we first present a state of the art of robot control and programming via augmented reality. This is followed by a description of our proposed AR interface and its components in Section III. In Section IV, we describe the experiments conducted to evaluate our interface. In Section V, we present and give a brief discussion on the results of our experiments. Finally, we conclude our paper in Section VII with some potential future work descriptions.

II. RELATED WORK

Augmented reality has been used in combination with robotics for various use cases allowing users to not only interact with or control the robot and visualise their intentions but also program the robot by defining target points in the actual workspace that the robot should reach. This enables inexperienced users with little to no programming skills to design complex robot operations and ensure that the trajectory to be taken by the real robot is collision free [2]. Various AR interfaces displayed either via a Head Mounted Device (HMD), smartphone, tablet or screen have been proposed in recent years for industrial settings as well as for collaborative tasks where the user can perform the task alongside the operator.
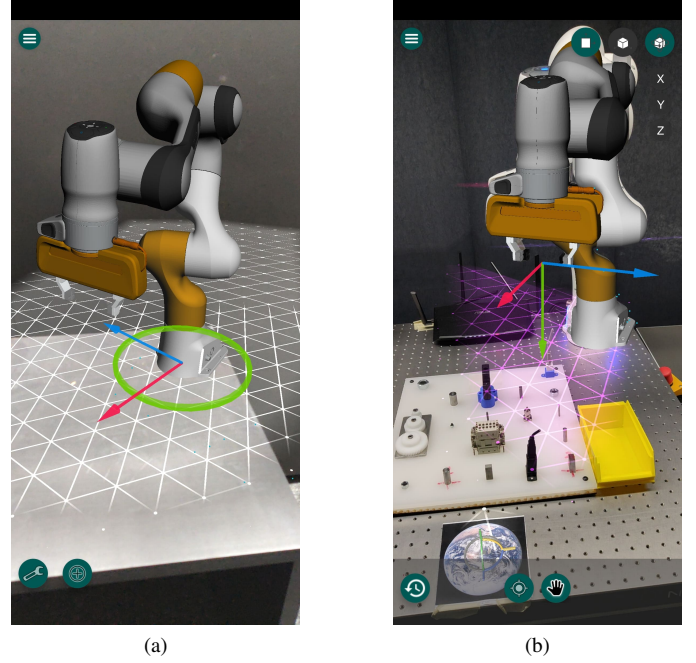

(a)   (b)

Fig. 2: Illustration of the methods to place a robot in the scene: (a) *manual* and (b) *marker calibration* (here, with a picture of Earth).

For instance, in [8], [11], [12], [13], the authors present mixed reality interfaces with a Microsoft HoloLens to program tasks by defining waypoints directly in the robot's workspace and previewing the trajectory via a virtual robot before sending the commands to a real robot. To place the waypoints and interact with the other features each respective interface offers, the interfaces employ features available on the HoloLens, namely gesture recognition as well as speech recognition and head pose in some cases [11]. The main drawbacks of such interfaces are that the selection and dragging features in the HoloLens may not always be reliable due to imperfect hand tracking [13] such that the trajectory may not be optimal and it may be difficult to perform tasks that demand a high degree of precision. To overcome this issue, in [8], the authors present an interface providing users with the ability to scale the path for a more accurate planning during such tasks, as well as the selection of the method by which the trajectory between waypoints should be interpolated (line, arc, etc.). However, despite allowing users to be hands-free, the HoloLens may be less intuitive than a 2D interface for novice users and HMDs are not necessarily available to most users. Additionally, it has been reported that such devices may cause discomfort and sickness which may be affecting their industrial acceptance [14].

In recent years, various platforms and toolkits such as ARCore, ARToolKit and Mixed Reality Toolkit (MRTK) have been introduced allowing users to develop augmented or mixed reality applications for certain smartphones and tablets making the technology more accessible. Such toolkits have been used
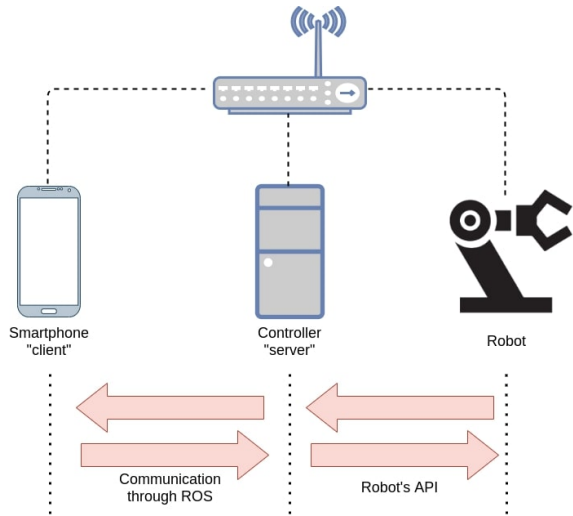
Fig. 3: Communication diagram of the application.

to develop interfaces enabling users to perform pick-and-place tasks in collaboration with a robot [14], define waypoints on a 2D surface for a tool to pass through [5] as well as control the joints of the robot via the interface and visualise a trajectory demonstrated to the robot via kinesthetic teaching [4]. However, in the case of [14], [5], the interfaces do not provide a visualisation of the whole robot which may be important in some environments such that the operators can check for eventual collisions. Whilst this is not the case with the interface presented in [4], the teaching mode presented in the interface involves demonstrating a task to the robot such that the robot is backdrivable through a gravity compensation controller, which is not available to most industrial robots.

In addition to some of the drawbacks mentioned above, most of the interfaces presented in this section require the operators to employ the interface whilst being next to the robot's workspace where the operators can interact directly with the physical objects present in the environment. Due to safety issues, similar to teaching a robot a task via a pendant, this would require shutting down the robots in the production line.

In this paper, we propose an AR interface available on smartphone and tablet providing users with the ability to not only control and program a robot in its workspace but also model the workspace via virtual objects. This provides the advantage of being able to use the interface to program robot tasks from different locations and adapt previously saved tasks to new situations, environments, tools, or robots.

III. METHODOLOGY

The interface proposed in this work was developed with Google's ARCore software library on Android Studio such that it can currently be run on most AR compatible Android devices[1]. The main functionality of the interface is that it provides non-expert users with an *offline* programming solution

by means of a virtual robot in a simulation environment, as well as an *online* programming solution by controlling the real robot with the provided augmented reality tools.

The virtual robot can be placed at a desired location within the environment and can then be connected to the real one, if required, to visualise what has been programmed offline or to directly control the robot online. We implemented several mechanisms to control the robot to achieve tasks in end-effector space. Additionally, the interface provides various options for the user to add/modify objects and obstacles, which can be moved, rotated and scaled in the workspace of the robot. The workspace created virtually can also be saved for later programming without requiring the real robot to be operated at the same time.

Videos of the AR interface and the experiments described in Section IV are available at: https://sites.google.com/view/idiap-ar-robot-interface/

A. Simulation of the robot via ARCore

We first describe the procedure to create a virtual robot in the desired workspace by explaining the connection to the real robot along with their calibration and communication. Upon launch, ARCore detects surfaces and planes which allow the users to interact with the virtual objects placed in the environment. The virtual robot is created by following the same approach as KDL[2]. It uses the description of a robot in the form of Unified Robot Description Format (URDF) to transform it into visual kinematic chains representing the real robot.

1) Positioning the virtual robot in the workspace: The interface provides two methods with which the virtual robot can be placed in the workspace: the *manual* method and the *marker calibration* method. The *manual* method lets the user select a location on a plane detected by ARCore upon which to place the virtual robot (Fig. 2a). The robot's orientation and position can then be adjusted with the constraint of moving on the plane upon which the robot is placed with the help of the blue and red translation axes and the green rotation ring as seen in Fig. 2a. The *marker calibration* method exploits ARCore's Augmented Images APIs[3] to superimpose the virtual robot onto the real robot as illustrated in Fig. 2b. It first detects the pose of a previously placed 2D image in the workspace with respect to the ARCore coordinate system. Using the homogeneous transformation between the image and the base of robot coordinate systems, the pose of the robot base with respect to ARCore is determined and is used to place the virtual robot.

2) Communication with the real robot: The virtual robot is connected to the real robot via Robot Operating System[4] (ROS) which allows interactions between the real and virtual robots. As ROS is already available for most of the available industrial and collaborative robots, it provides a generic interface that can work with any robot and that enables the

---

[1]https://developers.google.com/ar/devices

[2]https://www.orocos.org/kdl.html
[3]https://developers.google.com/ar/develop/java/augmented-images
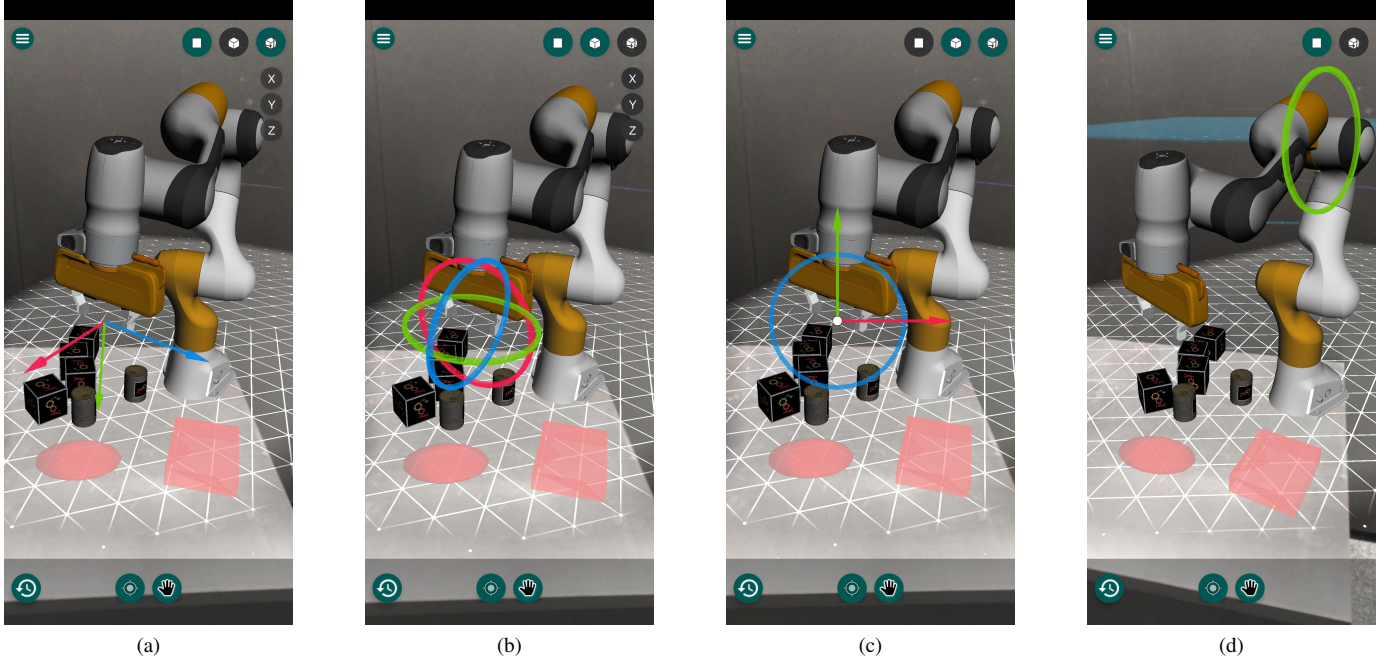[4]https://www.ros.org/

Fig. 4: Fully virtual workspace with objects and obstacles (in red) and the various ways of controlling a robot: (a) control of the end-effector position, (b) control of the end-effector orientation, (c) control of the end-effector along a plane and (d) control of the elbow.

use of multiple sources of sensory information together for the robot to work with. Our interface thus allows the ARCore device to communicate with the server via ROS, as illustrated in Fig. 3-*left*. The robot is controlled via ROS through a server computer enabling its real-time communication with the robot motors using the robot's API provided by the manufacturer of the robot, as illustrated in Fig. 3-*right*. As the implementation of our interface is built on top of ROS and not on a specific robot's API, our proposed work here is robot agnostic.

B. Motion control

We present here the methods available to control the robot's joints and end-effector as well as to plan trajectories. As opposed to joystick or teaching pendant based methodologies of programming robots, we provide more diverse options to move the robot.

Users are able to reposition the robot's end-effector using the translation axes shown in Fig. 4a and orient it using the rotational rings, each describing the rotation around the translation axes, as shown in Fig. 4b. These changes in the task space are applied to the joint space through a weighted inverse kinematics algorithm. Internally, a joint impedance controller was used to reach new joint space targets.

The third option enables the user to move the end-effector constrained on a 2D plane defined by the device's orientation, as seen in Fig. 4c. This option provides a more intuitive way to interact with the robot from the perspective of the user who may also be moving. The final option is the direct control of the joints that are highlighted in the interface such as the elbow

joint, as illustrated in Fig. 4d. The application also provides the user with the tools to create, save, load and replay end-effector trajectories on the device. To define a trajectory, the user can place keypoints in the workspace by controlling the robot via the techniques described above. Fig. 5a shows four of these workspace keypoints, depicted by partially transparent robot grippers.

These locations are then sequentially connected by an iterative Linear Quadratic Regulator (iLQR) [15] used as a trajectory planner, which determines the corresponding joint trajectory from the task space locations. From this trajectory, an equivalent task space trajectory is computed and depicted with a yellow curve as in Fig. 5a. Additionally, the interface provides the ability to replace the keypoints with multivariate Gaussian distributions which are represented by ellipsoids in the scene (Fig. 5b). This enables the user to specify the precision and coordination required at each point by rotating the ellipsoid and modifying the scale (described in the following section) along each axis resulting in a smoother trajectory.

As all of these techniques can be performed solely on the virtual robot or by directly controlling the real robot, the proposed interface is compatible for *online* and *offline* programming.

C. Simulation of virtual objects and workspaces

In *offline* programming, the user employs a simulator to visualise a virtual robot and define its motions to achieve a task. Industrial robotics tasks are often defined by the workspace where the robot is located and by the objects
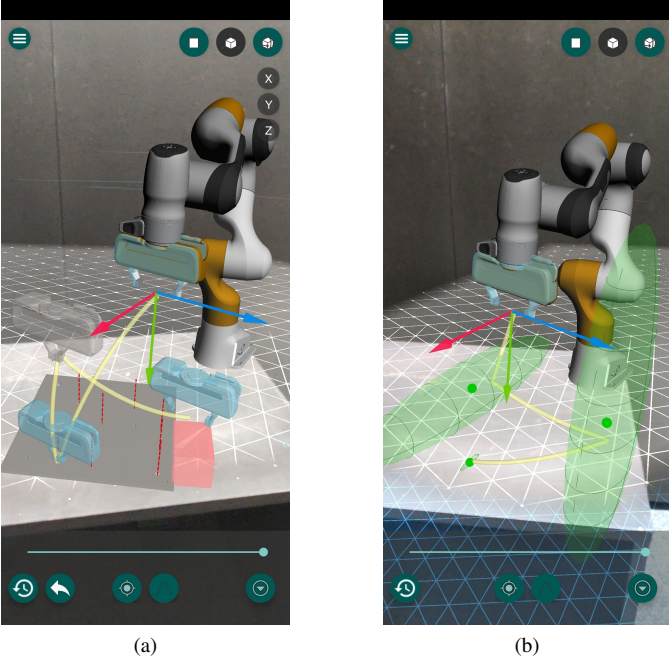
(a)　　　　　　　　　　　　(b)

Fig. 5: Trajectory planning with (a) keypoints illustrated by transparent end-effectors and (b) Gaussians (green ellipsoids) where their scale represents the (co)variations allowed along each axis, which are then converted to full precision matrices within the iLQR optimal control technique employed for planning.
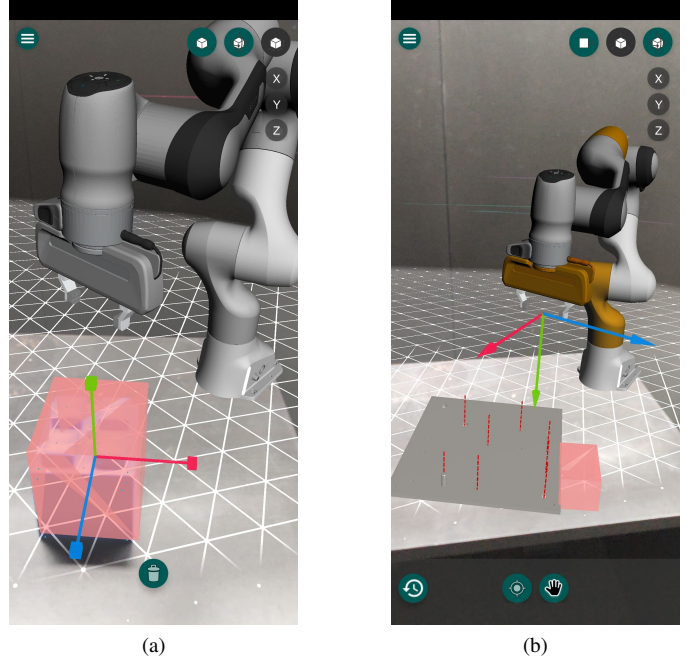

(a)　　　　　　　　　　　　(b)

Fig. 6: (a) definition of an obstacle by scaling and positioning a red virtual box to encompass an object; (b) virtual representation of the workspace used for the *offline* condition.

and tools that it can interact with. To this end, we propose to simulate, via the AR interface, workspaces with all their objects/tools available, without requiring the control and/or presence of the real robot.

This provides operators with the ability to design trajectories in a virtual workspace without the need to occupy the real workspace (see Fig. 6b). During the testing phase, adjustments, if required, could then be performed directly in the real robot's workspace by using the *marker calibration* method to superimpose the virtual robot onto the real one and loading the saved trajectories.

The interface allows users to place virtual objects and obstacles (see Fig. 4) in the shape of rectangular prisms, spheres and cylinders of modifiable sizes. The difference between the obstacles (in red in Fig. 4) and the objects (in black in Fig. 4) is that the former is mainly defined for the user to plan a collision-free trajectory of the robot. This means that whereas the robot can apply actions on the objects using the physics engine of the interface, we preferred to exclude these interactions on the obstacles to facilitate the programming. Note that here the collision-free trajectory is defined by the user interaction on the interface and the implementation of an obstacle avoidance planner is left as future work.

These objects and obstacles can be moved using the same methods, described in the previous section, to control the robot's end-effector and their scale along each axis can be modified using a similar representation to the one used to translate the end-effector. The creation of such objects and obstacles helps the user to create a virtual workspace that matches at best to the real one (see Fig. 6a) and use it as a template such as in Fig. 6b. Trajectories can then also be planned in this virtual workspace (Fig. 5a) using the functionalities described in the previous section and all the virtual components within the scene (i.e. workspace, objects, obstacles and trajectories) can be saved in a file on the device so that they can be reused/adapted for future tasks.

IV. EXPERIMENTS

To evaluate the intuitiveness of the proposed interface and the programming of a robot purely within a smartphone-based AR setup, we conducted a study where 18 participants programmed a peg disassembly task. This study was approved by Idiap Research Institute's Data and Research Ethics Committee.

1) Experimental Setup: The participants programmed a 7-axis Franka Emika Panda robot manipulator, with a peg disassembly task of the National Institute of Standards and Technology (NIST) task board 1 [16], see Fig. 7. For the AR interface, we used an Android smartphone (*6.39-inch* touchscreen display with a resolution of *1080 × 2340* pixels). The participants of our study (age range 20–35) had varying knowledge of robotics systems but little to no experience of virtual, augmented and mixed reality.

2) Conditions and Protocol: The participants were asked to perform the disassembly of a cylindrical peg from the NIST

task board and place it into a disposal box, as shown in Fig. 7. Two conditions were tested in the study: the *online* and the *offline* programming of the disassembly task.

In the *online* condition, the participants directly controlled the real robot's motions from the AR interface and their inputs were immediately translated into robot motions. Prior to the task, the virtual robot was aligned with the real one based on the marker calibration method described in Section III and the interface was connected to the robot as illustrated in Fig. 2b. Furthermore, the real and virtual robots were first set to a default joint configuration after which participants could then start the experiment by displacing the virtual end-effector via the options described in Section III-B. The task was considered successfully completed when the peg was correctly disposed inside of the box. A failure was recorded when the peg was either incorrectly disposed, dropped or an action performed by the user resulted in the robot's safety stop.

In the *offline* condition, participants instead controlled, via the AR interface, a virtual robot acting on a pre-stored virtual copy of the workspace of the *online* condition. In particular, the participants first placed the virtual workspace on a table different from the one of the real robot (see Fig. 6b) and programmed the disassembly of the same cylindrical peg from the board and its placement into a yellow box. The participants were given 5 minutes of time to define a robot trajectory by means of via-points, as shown in Fig. 5a. The robot trajectory was then sent to the real robot and enacted on the real workspace. If the peg was correctly disposed, the task was considered as successfully completed. In case of failure, the participants were given one extra minute to perform adjustments to the programmed trajectory in the virtual workspace, according to the visual feedback they observed from the real robot. This decision was made to provide a fair comparison with respect to the *online* condition where participants experienced real-time feedback while controlling the robot.

We adopted a within-subjects study design, with each participant operating the robot in both the *online* and the *offline* conditions. The order of the conditions was counterbalanced.

Prior to the experiment, the participants were provided with instructions and were given 1-2 minute to familiarise with the AR interface and the disassembly task.

3) Questionnaire and Logged Data: For each condition, we recorded whether the participants were successful in completing the task. Furthermore, the completion time t_c of the disassembly task was recorded (*i.e.*, seconds between the beginning of the task and its success/failure). At the end of the experiment, each participant filled a questionnaire with the following 6 Likert scale statements (*1 – completely disagree, 5 – completely agree*):

1) The interface is easy to understand,
2) I found the visualisation of the virtual robot useful,
3) I found it easy to control the robot in real time,
4) I found it easy to plan trajectories with the virtual robot,
5) I found the AR interface useful to program the robot,
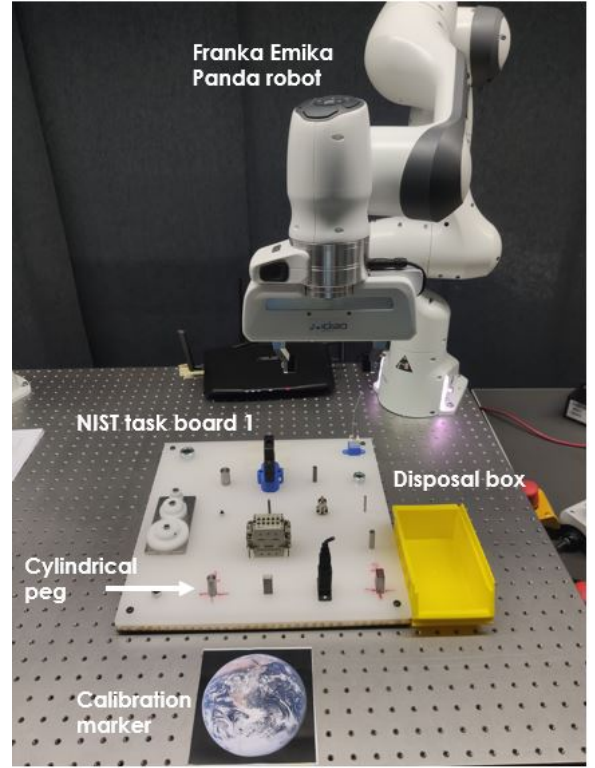6) I would use the AR interface for robot programming.

Fig. 7: Experimental setup with Franka Emika Panda, NIST board, calibration marker and disposal box.

Each statement included an optional comment section where participants could provide feedback about their experience and provide suggestions on what could be improved.

V. RESULTS

We hereafter present the analysis of the different metrics collected during the study. When relevant, the data is presented and the analysis is performed by separating the participants into two groups, with the 9 participants who experienced the *online* condition first and then the *offline* condition assigned to *group A*. The other 9 participants who experienced the conditions in the reverse order were assigned to *group B*.

We first computed the success rate of each group for each experiment as summarised in Table I. Our hypothesis entering the study was that the success rate in the *offline* condition for the participants in *group A* (that first operated the real-robot in the *online condition*) should be higher with respect to what observed for *group B*. The hypothesis was motivated by the fact that participants from *group A* have a better understanding of the application and the setup when performing this experiment

TABLE I: Success rate on the disassembly task, reported separately by groups and conditions.

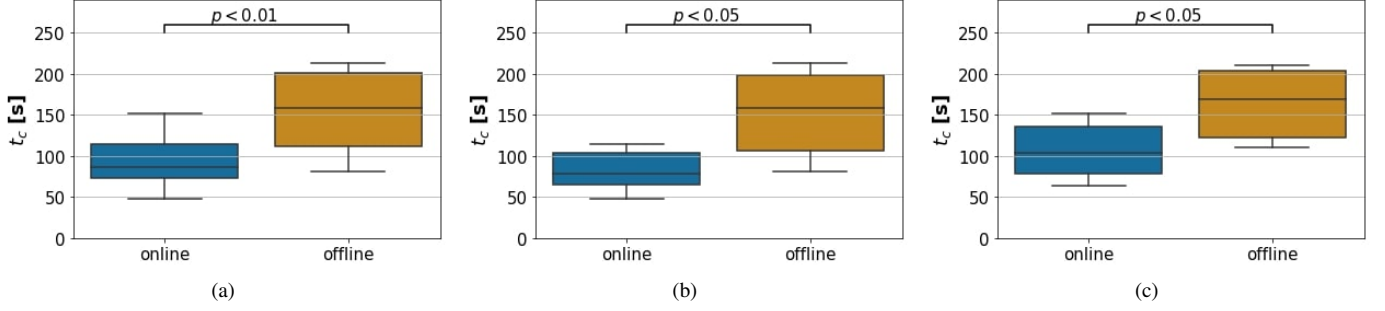|	*Online* condition	*Offline* condition	Pop. size
Group A	100%	78%	9
Group B	100%	67%	9
Overall	100%	72%	18

Fig. 8: Boxplots of the completion time $\mathbf{t_c}$ acquired during the successful experiments for (a) entire population, (b) *group A* and (c) *group B*.

compared to those of *group B*. Although we observed a 11% difference in success rate, no statistically significant difference was found (test on the Agresti-Coull interval, $p > .05$).

For the completion time $\mathbf{t_c}$, the descriptive statistics are presented in Fig. 8 as boxplots. We tested the data for normality with the Shapiro-Wilk test, rejecting the null hypothesis ($p < .05$). For each subset, we therefore ran a non-parametric Wilcoxon signed-rank test for differences between conditions. Statistically significant differences were found, both for the overall population ($p < .01$), and for *groups A* and *B* ($p < .05$).

To test for potential effects of the ordering of conditions, we compared the completion times $\mathbf{t_c}$ of *groups A* and *B* for each condition. We therefore run a non-parametric Mann-Whitney U test and found the difference between the two groups for the *online* condition to be statistically significant ($p < .05$), as shown in Fig. 9. A possible explanation for this difference may be due to the discrepancy between the control of the virtual robot in comparison with the dynamics of the real robot. As participants from *group B* had performed the *offline* condition first, we noticed that they took more time to adjust to the new dynamics than their counterparts. Finally, no statistically significant difference between groups was observed for the *offline* condition ($p > .05$).
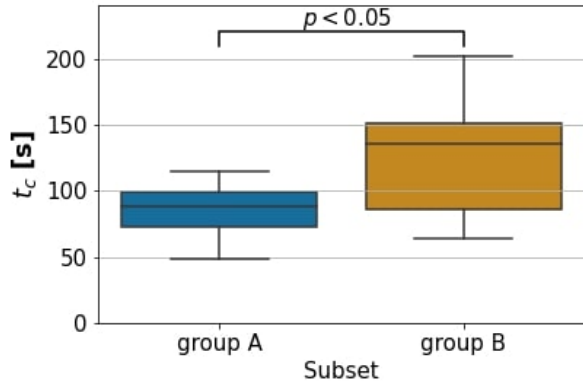

Fig. 9: Boxplots of the completion time $\mathbf{t_c}$ for *group A* and *B* for the *online* condition.

A. User feedback

The scores of the Likert scale statements presented in Section IV-3 are visualised in Fig. 10 for the whole study population. As for the completion time $\mathbf{t_c}$, we looked for differences of questionnaire scores between the two groups with a Mann-Whitney U test, finding however no statistically significant differences ($p > .05$).

Overall, the results indicate that the participants perceived the proposed AR interface as easy to understand and useful for robot programming. However, out of the 18 participants, only 7 participants stated that they would use the interface for programming industrial robots. While praising the easiness and quickness of use of the interface, the rest of the participants raised concerns about the interface's lack of accuracy, especially for manufacturing tasks such as insertion. Most participants mentioned how it was difficult to perform precise robot motions by means of the dragging motion on the smartphone screen.

On the other hand, almost all participants stated that the virtual workspace provides a good representation of the real workspace and that the interface offers an intuitive way of planning trajectories as errors can easily be visualised and corrected.

VI. DISCUSSION

The results presented in Section V indicate a promising success rate for both *online* and *offline* programming with
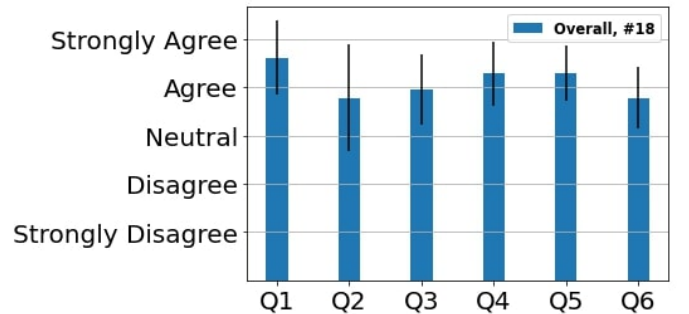

Fig. 10: Average user ratings for the assertions provided in Section IV-3.

completion times $\mathbf{t_c}$ ranging from 1 to 4 minutes. Nevertheless, during the study we observed how certain aspects of the interface could have hindered the participants in accomplishing the task.

A general problem of AR interfaces is their weakness in the estimation and visualisation of depth [17], [18]; a limitation that requires the users to adopt coping strategies such as *e.g.*, change their location to obtain different views of the scene and, consequently, a better perception of depth. In our study, we indeed observed how the participants who adopted such strategies performed the disassembly tasks faster and more accurately than their counterparts. This was especially true for the *offline* condition, where the issue of the AR interface with depth was particularly relevant.

Another issue indicated by the study's participants was in the perceived lack of accuracy during dragging motions across the screen. This led to inaccurate or unexpected motions on the real robot, especially when the translation axis being manipulated was perpendicular to the surface of the device. A potential solution to this issue would be to automatically disable the control of the robot along the axis being employed when the aforementioned condition is met. While avoiding the problem of unexpected robot motions, this solution would also encourage the users to move around the workspace, in order to regain control of a certain axis, indirectly addressing the aforementioned issue of depth perception.

Finally, the user feedback also provided insights into various potential areas of research such as using the interface to control the real robot remotely, in a similar fashion to teleoperation. Additionally, some participants stated that they would consider using the AR interface not only as a control tool but also as a visualisation/monitoring tool. For example, in a learning from demonstration [1] scenario, the AR interface could be used to inspect the quality of the demonstrated trajectories or to visualise their variability with aptly placed Gaussians, in a way similar to [4].

VII. Conclusion & Future work

In this paper, we presented an augmented reality interface for smartphones and/or tablets, enabling users to control a robot in real-time, to program it *offline* as well as to model a workspace by means of virtual objects. The proposed interface aims to provide operators programming industrial robots with an alternative to common *online* programming methodologies, for which the presence of a real robot is not required and pre-stored workspaces and trajectories can be adapted to re-program a robot.

Future work will address the challenge of performing more accurate motions. We will investigate the option of adapting the sensitivity of the dragging action on the interface, giving the required precision to the user when felt necessary. Also, we will test the option of switching from the use of dragging motions to a button based system where users can affect the displacement along each axis individually. This would enable users to perform rapid actions via the current dragging motions

on the screen and then switch to the button based system when more accuracy is required.

Additionally, the ARCore software development kit currently lacks the ability to detect 3D objects. Having such a feature would allow to automatically place virtual objects or obstacles, resulting in a faster creation of the virtual workspace. We plan to integrate a 3D object detection pipeline in the interface to provide this capability. Motivated by the promising results and the participants' feedback presented in Sections V and VI, we also plan to investigate possible extensions of the approach to other robot applications beyond manufacturing.

References

[1] S. Calinon, "Learning from Demonstration (Programming by Demonstration)," Ang, M.H., Khatib, O. and Siciliano, B. (eds), *Encyclopedia of Robotics*, Springer, 2018.

[2] F. Zhang, C.Y. Lai, M. Simic and S. Ding, "Augmented reality in robot programming," *Procedia Computer Science*, vol. 176, pp. 1221-1230, 2020.

[3] S. Blankemeyer, R. Wiemann, L. Posniak, C. Pregizer and A. Raatz, "Intuitive Robot Programming Using Augmented Reality," *Procedia CIRP*, vol. 76, pp. 155-160, 2018.

[4] M. Gradmann, E. M. Orendt, E. Schmidt, S. Schweizer and D. Henrich, "Augmented Reality Robot Operation Interface with Google Tango," *Intl Symposium on Robotics (ISR)*, 2018, pp. 1-8.

[5] S. M. Chacko, A. Granado, V. Kapila, "An Augmented Reality Framework for Robotic Tool-path Teaching," *Procedia CIRP*, vol. 93, pp. 1218-1223, 2020.

[6] S. M. Abbas, S. Hassan and J. Yun, "Augmented reality based teaching pendant for industrial robot," *Intl Conf. on Control, Automation and Systems*, 2012, pp. 2210-2213.

[7] S.K. Ong, A.W.W. Yew, N.K. Thanigaivel, A.Y.C. Nee, "Augmented reality-assisted robot programming system for industrial applications," *Robotics and Computer-Integrated Manufacturing*, vol. 61, 2020.

[8] M. Ostanin, S. Mikhel, A. Evlampiev, V. Skvortsova and A. Klimchik, "Human-robot interaction for robotic manipulator programming in Mixed Reality," *IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2020, pp. 2805-2811.

[9] H.-L. Chi, S.-C. Kang, X. Wang, "Research trends and opportunities of augmented reality applications in architecture, engineering, and construction," *Automation in Construction*, vol. 33, 2013, pp. 116-122.

[10] Z. Makhataeva and H. Varol, "Augmented Reality for Robotics: A Review", *Robotics*, vol. 9, no. 2, p. 21, 2020.

[11] 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," *IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, 2018, pp. 1838-1844.

[12] M. Ostanin, A. Klimchik, "Interactive Robot Programming Using Mixed Reality," *IFAC-PapersOnLine*, vol. 51, issue 22, 2018, pp. 50-55.

[13] S. Y. Gadre, E. Rosen, G. Chien, E. Phillips, S. Tellex and G. Konidaris, "End-User Robot Programming Using Mixed Reality," *Intl Conf. on Robotics and Automation (ICRA)*, 2019, pp. 2707-2713.

[14] S. M. Chacko and V. Kapila, "Augmented Reality as a Medium for Human-Robot Collaborative Tasks," *IEEE Intl Conf. on Robot and Human Interactive Communication (RO-MAN)*, 2019, pp. 1-8.

[15] W. Li and E. Todorov, "Iterative Linear Quadratic Regulator Design for Nonlinear Biological Movement Systems," *Intl Conf. on Informatics in Control, Automation and Robotics (ICINCO)*, 2004, pp. 222-229.

[16] K. Kimble, K. Van Wyk, J. Falco, E. Messina, Y. Sun, M. Shibata, W. Uemura, and Y. Yokokohji, "Benchmarking protocols for evaluating small parts robotic assembly systems," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 883–889, 2020.

[17] F. Heinrich, K. Bornemann, K. Lawonn and C. Hansen, "Depth Perception in Projective Augmented Reality: An Evaluation of Advanced Visualization Techniques," *ACM Symposium on Virtual Reality Software and Technology (VRST)*, art. 26, 2019, pp. 1-11.

[18] F. E. Jamiy and R. Marsh, "Distance Estimation In Virtual Reality And Augmented Reality: A Survey," *IEEE Intl Conf. on Electro Information Technology (EIT)*, 2019, pp. 63-68.