applied sciences

MDPI

Article

Human-Robot Collaboration: An Augmented Reality Toolkit for Bi-Directional Interaction

Graziano Carriero †, Nicolas Calzone, Monica Sileo †, Francesco Pierri *,†, Fabrizio Caccavale † and Rocco Mozzillo *,†

School of Engineering, University of Basilicata, Via dell'Ateneo Lucano 10, 85100 Potenza, Italy; graziano.carriero@unibas.it (G.C.); nicolas.calzone@studenti.unibas.it (N.C.); monica.sileo@unibas.it (M.S.); fabrizio.caccavale@unibas.it (F.C.)

* Correspondence: francesco.pierri@unibas.it (F.P.); rocco.mozzillo@unibas.it (R.M.)

† These authors contributed equally to this work.

Abstract: This work proposes an Augmented Reality (AR) application designed for HoloLens 2 which allows human operators, without particular experience or knowledge of robotics, to easily interact with collaborative robots. Building on the application presented in a previous work of the authors, the novel contributions are focused on a bi-directional interaction that manages the exchange of data from the robot to the human operator and, in the meantime, the flow of commands in the opposite direction. More in detail, the application includes the reading of the robot state, in terms of joint positions, velocities and torques, the visualization of the workspace and the generation and manipulation of the end-effector trajectory by directly moving a set of way-points displayed in the AR environment. Finally, the trajectory feasibility is verified and notified to the user by taking into account the workspace limits. A usability study of the AR platform has been conducted involving 45 participants with different ages and expertise in robot programming and Extended Reality (XR) platforms, comparing two programming methods: a classical kinesthetic teaching interface, provided by the Franka Emika Panda cobot, and the presented AR platform. Participants have reported the effectiveness of the proposed platform, experiencing less physical demand and higher intuitiveness and usability.

Keywords: augmented reality; robotics; head-mounted display; human-robot interaction

Citation: Carriero, G.; Calzone, N.; Sileo, M.; Pierri, F.; Caccavale, F.; Mozzillo, R. Human-Robot Collaboration: An Augmented Reality Toolkit for Bi-Directional Interaction. *Appl. Sci.* **2023**, *13*, 11295. https://doi.org/10.3390/app132011295

Academic Editors: Zhijun Li and Jing Luo

Received: 19 July 2023
Revised: 5 October 2023
Accepted: 12 October 2023
Published: 14 October 2023

1. Introduction

The base concept of the Industry 4.0 paradigm is the development of autonomous technologies involving robots working in close collaboration with human operators [1,2]. Sheridan proposes a classification of the autonomy level of robots by assigning value 1 to the case of *computer offers no assistance* and value 10 to the totally autonomous case of *computer decides everything and acts autonomously, ignoring the human* [3]. At the intermediate levels, Human-Robot Collaboration (HRC) is involved, requiring the adoption of collaborative robots, called cobots [4]. A key point of collaborative robotics is the information exchange, which can be uni-directional if the cobot communicates with the human or vice versa, or bi-directional if a two-way stream is established. More in detail, the human shall be able to know the status of the cobot and predict in real time its intentions and, on the other side, she/he shall give instructions and commands to the cobot to proactively change and adapt the execution of its tasks.

Effective bi-directional HRC can be developed by introducing interfaces on Virtual Reality (VR) or Augmented Reality (AR) environments, which can allow human operators, without particular experience or knowledge in robotics, to easily interact with cobots. As reported in De Franco et al. [5], several studies have shown that the use of AR improves the task efficiency and the HRC [6].

This work proposes an AR application, designed for HoloLens 2 [7], which can be used in various research areas—industrial research in particular. This project is built on that proposed in [8], but, whilst in [8] the architecture provides only one-way communication between the robot Franka Emika Panda [9] and the human operator, here it has been extended by allowing the human operator to actively interact with the cobot, giving it inputs and commands. The development software for HoloLens 2 is Unity [10], while the data from the cobot and the command to it are managed by ROS (Robot Operating System) [11]. The communication between HoloLens 2 and ROS is achieved through the ROSSharp library [12].

The platform guarantees the exchange of data related to the status and the intention of the cobot in real time during the task execution within the Head-Mounted Display (HMD). Moreover, it is possible to assign a trajectory to the robot end-effector by grabbing and moving its hologram towards the final position in the AR environment. The user can show the robot workspace in the AR and verify that both the final position and the trajectory way-points are the expected ones. The cobot workspace can be also visualized giving feedback to the operator in terms of robot end-effector reachability. The hologram of the cobot workspace is displayed in red color if the operator places the end-effector hologram outside of the reachable area: a visible warning asks the user to move the end-effector to a reachable position. Conversely, the workspace hologram is colored green if the position of the end-effector hologram is inside the reachable area. Once the end-effector hologram is released in a feasible position, the desired trajectory is computed and then visualized in AR for the operator validation: she/he can further modify it dragging the displayed way-points in a different position, e.g., to avoid obstacle collisions. Hence, a ROS node interpolates the new way-points by using a spline and the computed trajectory is shown. Finally, the operator can command the cobot to track the desired trajectory.

The adoption of AR has many benefits for effective human-robot interaction. Following the taxonomy in [13], the purposes of the proposed application can be categorized as:

1. Facilitate Programming;
2. Improve Safety;
3. Communicate Intent.

In fact, the whole platform is designed for a human operator in strict contact with a cobot and represents a tool for tackling industrially relevant robotic tasks that require human-robot interaction. It can be viewed as an attempt to respond to real industrial needs, with the aim of facilitating the robot programming in the context of Industry 4.0. More in detail, it is designed in such a way:

- to make low-skill operators able to program the robot for complex collaborative tasks;
- to make the operator aware of the robot's intention by showing the planned trajectory in the AR environment and the robot workspace, in order to support a safe interaction in the absence of exteroceptive sensors;
- to improve the flexibility in trajectory planning in unstructured environments.

A usability test, conducted through an extensive experimental campaign involving 45 voluntary users with different levels of expertise, has assessed the effectiveness of the presented approach compared to a classical kinesthetic teaching task. The System Usability Scale survey [14] has been carried out in the experimental campaign and confirmed an improvement in the users' preference for the HoloLens platform compared to the classical programming interface. According to the results, the users' confidence in using the robot as well as the general usability have increased. In addition, the perceived workload, computed by resorting to the NASA Task Load Index method [15], indicated less physical demand, at the expense of a slightly higher cognitive demand, with respect to the classical programming interface.

After giving some background on the new operator roles in Industry 4.0, several methods to support human-robot interaction are analyzed: in Section 2, different AR systems are discussed and their benefits and drawbacks are evaluated. In Section 3,

the developed platform architecture and the experimental setup are described, while the application functionalities are explained in the following Section 4. Subsequently, AR features are outlined in Section 5: the robot calibration procedure, the robot state communication, the trajectory generation and the validation phase, assisted by the robot workspace visualization. A detailed analysis of the experimental results of the usability study of the AR platform is reported in Section 6. Finally, in Section 7, a review of the work is presented and future developments are discussed.

2. State-of-the-Art

Robots, sensors, devices and human resources are intelligent entities that can be integrated in a smart factory according to the paradigms of Industry 4.0 approach. Thanks to their cooperation and real-time data sharing, smart factories are able to reach high levels of flexibility and reconfigurability in order to dynamically adapt to current market changes.

The aim to improve flexibility of logistic and production systems is pursued not only with technological resources, e.g., smart devices and machines, but also with a new role of human operators, as the modern concept of the so-called *Operator 4.0* [16]. In smart factories, various augmentations of the original human capabilities are adopted to empower their "smart operators" with new skills and gadgets to fully capitalize on the opportunities being created by Industry 4.0 technologies. For that matter, the increasingly widespread introduction of cobots in industry provides an improvement of smart operator productivity; in addition to relieving her/him from non-ergonomic and repetitive tasks, cobots and humans can interact, establishing an information stream, e.g., by means of multi-modal interfaces [16].

In Human-Robot Interaction (HRI) systems, agency cannot be exclusively attributed to humans but it is distributed among humans and non-human resources [4]; interaction is exploited by assigning operators and robots different sub-tasks leading to a common goal (*cooperation*), or a sequence of tasks accomplished together toward a shared goal (*collaboration*). In the latter case, the envisioned goal is to make both human operator and cobot proactive, providing a bi-directional stream of exchanged information. For this purpose, hand gesture recognition could help humans to convey information and express intentions in a simple way. Relevant advantages of this solution with respect to traditional human-machines interaction, for instance mouse, keyboard and 2D display requiring a fixed operating space, are as follows: high degree of differentiation, strong flexibility and high efficiency of transmission, achievable through data gloves, ultrasound, vision or wearable devices based on electromyographic signals [17].

In [18], a novel AR-based teleoperation system is proposed, where the 3D virtual robot model dynamically tracks the current real robot state. The system architecture is composed of a multi-DoF industrial robot and two RGB-D sensors, connected to each other in the ROS framework and to the AR interface, built with head and hand gesture trackers, by means of ROSBridge server. The system's maneuverability is enhanced due to this setup which guarantees interaction with the environment by hand gestures, tracked using a Leap Motion controller mounted on the HMD. An "interaction proxy" manages hand gestures, making task performing easier and avoiding unintentional gesture disturbances. Moreover, two control modes (coarse movement mode and fine movement mode) are proposed to prevent operational faults related to each type of movement.

In [5], the adoption of the Microsoft HoloLens device allows for bi-directional communication with a cobot, both with visual and acoustic feedback. The visual ones make the operator aware of robot movements, increasing her/his trust in collaboration, whereas the acoustic feedback is useful when the 3D holograms are out of sight or in adverse lighting conditions. The development tool is Unity, together with Mixed Reality Toolkit [19], while the robot is controlled through ROS. The communication exploits the User Datagram Protocol (UDP) and it is designed aiming at information exchange reduction, also preventing communication overload. Indeed, the operator wearing the HoloLens receives information about the robot state both in terms of positions and contact forces. A light with a colour

code improves the operator safety: the color is green if the robot moves away from the operator; red if the robot approaches the operator; and yellow if the robot is waiting for user input. On the other side, the operator sends inputs to the robot through hand gestures, in order to select the control strategy and the task to be performed. A polishing task is executed comparing 2D display and AR systems: the AR interface provides better feedback, allowing the user to reduce his/her effort variances during the task execution.

Rosen et al. propose an AR system that outperforms with respect to 2D display in [20], where the AR interface reduces the task completion time (about 38%), increases the accuracy of collision prediction by an average of 15% and precision by 11%. In order to map the environment in real time and collocate the operator inside the AR reconstruction, the system set-up consists of Microsoft HoloLens headset, an inertial measurement unit, four cameras and an infrared (IR) depth sensor. The user is immersed in a virtual space, perceiving the scene through the HoloLens device and interact with it through hand gestures (through 3D game engine Unity) and voice commands. One-hand and two-hand gestures are exploited in order to specify the desired position and orientation of robot end-effector and a voice command instructs the robot to plan the trajectory, which is validated by the operator through AR. If the operator notices that an object collision could occur, she/he replans the trajectory with a stochastic planner.

The same approach is proposed in [21], where an AR interface supporting robot programming is presented: pick-and-place tasks are designed using sequences of way-points. In this way, 3D trajectories are generated and eventually altered by moving, adding and deleting the way-points. Furthermore, the interface allows the users to visualize the assigned path that the robot would take from two consecutive way-points, which is planned by means of 'MoveIt!'. Robot programming through AR platforms is also investigated in [22], where two trajectory generation methods are proposed: free space trajectories and surface trajectories. In the first case, the user selects the pick-and-place locations in the physical workspace through speech and gesture inputs that are recognized by HoloLens. A virtual scenario allows the user to dynamically modify the path in free space during the simulation or even during the execution. On the other hand, virtual paths can be defined on a surface by using head tracking, speech and gesture recognition. The robot end-effector can then move along the path while keeping its orientation normal to the surface, due to an impedance-controlled mode, which ensures the application of a constant normal force.

Either in [21] or [22], the usability of proposed AR systems is evaluated and compared to classical approaches. In [21], volunteers have been instructed to program the robot to pick up and place a cube onto a platform, bringing it over walls of different heights. The task was completed using two different interfaces: a 2D monitor interface and the HoloLens AR application. In [22], participants were instructed to perform two tasks using the proposed AR platform and a kinesthetic teaching interface, i.e., a gamepad for storing way-points: a surface contact task, consisting in the erasure of lines drawn on a flat surface by means of a marker eraser fit in the robot end-effector, and a free space task, i.e., a pick-and-place in the presence of an obstacle. In both cases, significant reductions in the teaching time and the NASA Task Load Index [15] combined with increments of performance and usability supported the use of the AR systems for novice users.

In [23], Kästner et al. adopt the AR to improve the navigation of a mobile robot and visualize trajectory and spatial mapping. The setup consists of a Microsoft HoloLens as the AR device, and Kuka Youbot as the mobile robot. The development platform used for HoloLens is Unity 2018, while the robot is controlled through ROS Hydro. The communication between ROS and HoloLens is ensured by the ROSBridge protocol, which follows a specific JavaScript Object Notation (JSON) notation for messages. Furthermore, the ROSSharp library was used to allow HoloLens to follow the ROS publish and subscribe approach. ROSSharp is a set of open-source software libraries and tools in C# for communicating with ROS from .NET applications, in particular, Unity.

Furthermore, AR can be adopted to test program safety by a digital twin: real-time interaction between the real and a virtual world allows for programming by demonstration,

i.e., a widespread way to program an industrial robot, which is required to save the poses the robot has assumed in a virtual environment, without unsafe actions occurring. In the system developed by Ostanin et al. [24], basic and advanced functionalities of HRC are implemented:

- robot communicates its status, namely its joint positions and pose by AR interface;
- human plans a geometrical path specifying a sequence of way-points, i.e., end-effector position and orientation, by using the framework *MoveIt!* [25], and choose the 3D geometrical primitive which connects each goal point (e.g., an arc, a line); trajectory feasibility is then checked in joint space;
- it is possible to visualize robot workspace (WS) in AR, assisting human to assess point achievability, warning her/him in the presence of an obstacle and slowing down robot movements in case the operator shares WS with robot;
- for a correct virtual model of the robot placement, an algorithm based on point clouds processing is developed, which analyzes the scene, searches for robot-like objects and finally finds the position of the robot;
- during the trajectory generation phase, the shortest path to connect the sequence of goal points in joint and Cartesian space is automatically planned, while the Unity engine API provides object collision avoidance and path scaling.

3. Development Tools and Experimental Setup

The architecture of the developed platform is shown in Figure 1 and includes:

1. The HMD Microsoft HoloLens 2, produced by Microsoft and running the Windows Holographic operating system [7]. It is equipped with a Qualcomm Snapdragon 850 ARM64 CPU with 64 GB of internal storage and 4GB of RAM. Moreover, there is a dedicated processor for holograms visualization (HPU 2.0).
2. The collaborative robot Franka Emika Panda [9], characterized by seven rotational joints equipped with torque sensors and encoders to provide the joint positions and velocities. The cobot is controlled by using an external workstation PC connected via Ethernet to the controller. An open source C++ library, called `libfranka`, is used which allows for real-time commands to be sent and provides the current status of the robot at the frequency of 1 kHz.
3. A ROS node, allowing for commands to be sent to the cobot through the integration of `libfranka` with the entire ROS ecosystem. In particular, ROS Kinetic has been used in the experiments.
4. A Unity platform, version 2021.1.22 (64 bit) [10], with the Mixed Reality Toolkit (MRTK) library in version 2.7.2; this Microsoft library provides several additional components and features, which allows for the development of the cross-platform AR application in Unity. The code is written in C# through Visual Studio 2019 and perfectly integrated into the Unity development environment.
5. ROSSharp library [12], a Siemens Open-Source library designed for Unity, performing the data exchange between cobot and HMD or, in general, for communication of .NET applications with ROS. In this project, however, the version v1.2c of the ROSSharp library, designed exclusively for Universal Windows Platform (UWP) applications, is adopted [26].
6. ROSBridge, i.e., the protocol which enables messages exchange from C# to C++, by using the JSON format.

By the GitHub repository in [27], the Unified Robot Description Format (URDF) of the cobot is inherited and then imported in Unity [28]. In AR, the cobot hologram overlaps the real one by means of a calibration procedure exploiting the SampleQR code application [29], provided by Microsoft, which requires NuGet [30], a package manager designed for Unity. In Table 1, the used hardware and all the development tools are reported.

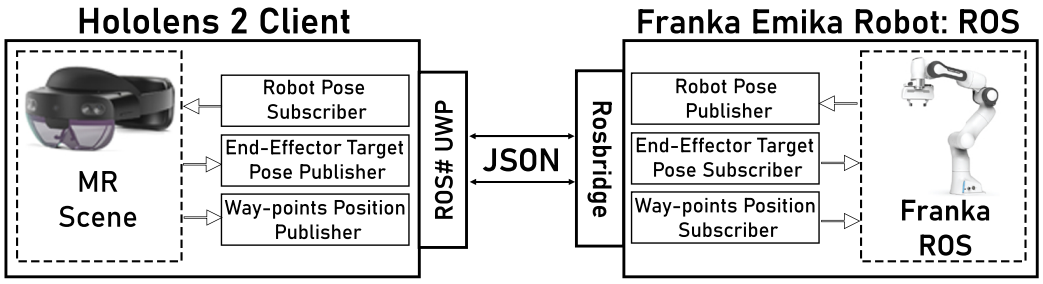


Figure 1. System architecture.

Table 1. Hardware and development tools.

Head-Mounted Display	Microsoft HoloLens 2
Robot	Franka Emika Panda
Unity	Unity 3D 2021.1.22 64 bit
Visual Studio	VS Community 2019
Mixed Reality Toolkit	MRTK v2.7.2
ROS Version	Kinetic
ROSSharp	ROSSharp UWP

4. Design of Augmented Reality Platform

Mixed Reality Toolkit is useful to generate the main menu and buttons (Figure 2): the menu consists of a panel where three default buttons are located on the first row dealing, with real time numerical data supply, namely positions, velocities and torques, after being collected by ROS system [8]. If the operator selects one among them, a new scene with its own panel is opened.

Figure 2. Application main menu.

Through the "*Human-Robot Interaction*" button on the main menu, the user can access the AR scene and the control panel, shown in Figure 3, with eight different buttons. More in detail:

- "*Calibrate*" starts the calibration procedure; as explained in Section 5.1, the HoloLens searches for a QR-code in the AR scene to place cobot hologram. All the other functionalities are enabled only after calibration;
- "*Move End-Effector*" sends the command to the ROS node to calculate a linear trajectory for the end-effector moving from the current position to the target one (see Section 5.4);
- "*Show Last Trajectory*" displays the trajectory previously generated. If a trajectory is already present on the scene, the button header becomes "*Hide Last Trajectory*" and allows the user to hide it;

- the same idea is below the "*Show Joint Spheres*" button, which can activate or deactivate the spheres placed in correspondence of the robot joints, which represent how far the joints are from their limits (see Section 5.2);
- "*Show Workspace*" button enables or disables the workspace visualization (see Section 5.3);
- "*Send Way-points*" button allows the human operator to send the ROS node the way-points of the previously computed trajectory after her/his visual validation (see Section 5.5);
- through the "*Move Robot*" button, the operator commands the robot to track the trajectory (see Section 5.6);
- "*Back to Main Menu*" button allows the user to return to the application main menu.

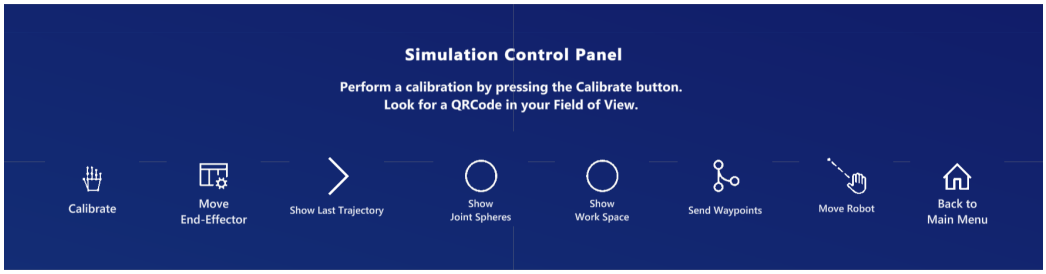


Figure 3. Simulation scene control panel.

5. Augmented Reality Features

In this section, each feature provided by the developed application will be detailed.

5.1. Calibration

The pose of the cobot virtual model in AR environment is assigned through a calibration procedure. More in detail, the calibration is needed to align the virtual features with the real environment. It consists of transferring a fixed frame, whose relative position and orientation with respect to the robot base frame is known, from the real to the augmented environment. A dedicated algorithm has been developed to evaluate the calibration accuracy. It is based on a quantitative evaluation of the position and orientation errors of the virtual hologram with respect to its real twin.

In Figure 4, two snapshots showing the two steps of the calibration procedure are reported: the robot hologram collocation phase (Figure 4a) and the subsequent validation phase (Figure 4b).

The calibration procedure requires the positioning of two QR codes in the user's field of view in a known position and orientation with respect to the real robot base frame $\mathcal{F}_r$. The first QR code allows the user to collocate the cobot hologram in the AR environment, while the second one permits the user to numerically evaluate the procedure success and compute the calibration errors in terms of position and orientation.

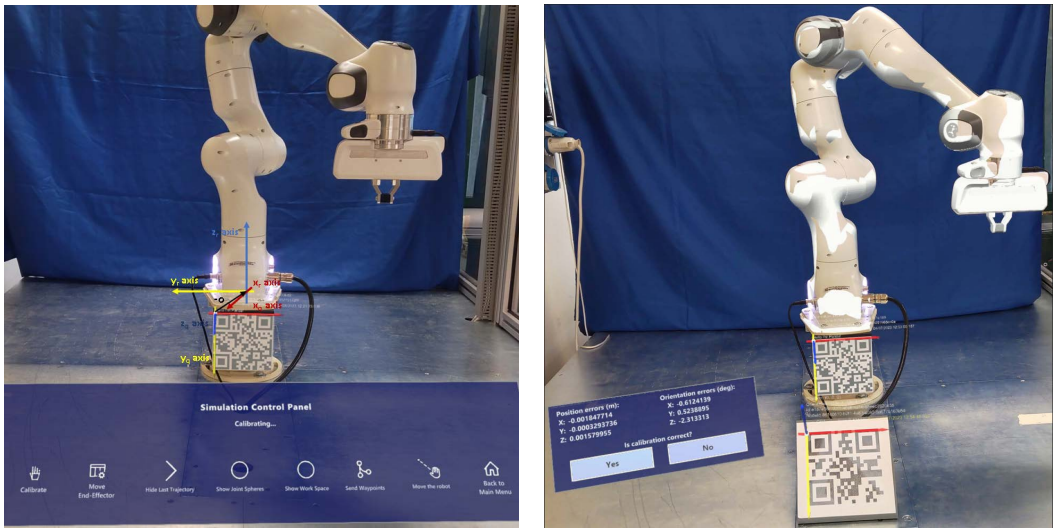
Denote with $\mathcal{F}_q = \{O_q, x_q y_q z_q\}$ and $\mathcal{F}_v = \{O_v, x_v y_v z_v\}$ the coordinate frames attached to the top left corner of the two QR codes, respectively. Based on the previous assumption, the relative positions ($\boldsymbol{p}_q^r$ and $\boldsymbol{p}_v^r$) and orientations (expressed in terms of rotation matrices [31] as $\boldsymbol{R}_q^r$ and $\boldsymbol{R}_v^r$) between the QR codes' frames and the robot base frame are constant and known in the real world. The superscript r indicates that all the quantities are expressed with respect to the robot base frame.

When the two QR codes are in the user's field of view, she/he presses the "*Calibrate*" button, shown in Figure 3, thus HoloLens begins to search for QR codes exploring the surrounding environment (Figure 4a) by exploiting the Unity QR code tracking algorithm [29]. If the search is successful, the QR tracking algorithm returns an estimation of the positions, $\hat{\boldsymbol{p}}_q$ and $\hat{\boldsymbol{p}}_v$, and the orientation matrices $\hat{\boldsymbol{R}}_q$ and $\hat{\boldsymbol{R}}_v$, of the QR frames, $\hat{\mathcal{F}}_q$ and $\hat{\mathcal{F}}_v$, in the HoloLens coordinate frame $\mathcal{F}_0$ defined by default at each start of the application (the superscript 0 is omitted for notation compactness). Then, the application notifies the operator with an acoustic signal and the robot hologram can be collocated in the AR environment.

Thus, the robot hologram is firstly positioned with its coordinate frame coincident with $\hat{\mathcal{F}}_q$; then, it is translated of the vector $-\boldsymbol{p}_q^r$ and rotated through the rotation matrix $\boldsymbol{R}_r^q = \boldsymbol{R}_q^{r\mathrm{T}}$. For example, in the setup developed in this work, shown in Figure 4b, the matrix $\boldsymbol{R}_r^q$ is given by:

$$\boldsymbol{R}_r^q = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ -1 & 0 & 0 \end{bmatrix}. \tag{1}$$

In this way, the coordinate frame of the hologram is expected to be coincident with the real robot base frame.



(a) Calibration button in the Control panel. (b) Validation of calibration procedure.

Figure 4. Calibration procedure.

The QR code pose detection could be affected by the perspective distortion due to the relative height of the human operator point of view with respect to the QR frame during the calibration phase. Hence, if HoloLens perceives the QR plane as tilted with respect to the vertical plane, a further rotation is added in order to compensate this effect and make the QR code frame always aligned with the robot base.

Once the hologram is collocated in the AR environment, the user has to verify the calibration accuracy. For this reason, she/he has to evaluate the position and the orientation of the robot hologram as well as the calibration errors which are displayed on a panel, as shown in Figure 4b. If the "*No*" option is chosen, then the procedure must be repeated and a message appears on the control panel.

Position errors are calculated by assessing the difference between the real and the AR reconstructed vector starting from O_q and pointing to O_v, i.e.,

$$\boldsymbol{e}_{pos}^q = \hat{\boldsymbol{R}}_q^{\mathrm{T}}(\hat{\boldsymbol{p}}_q - \hat{\boldsymbol{p}}_v) - \boldsymbol{R}_r^q(\boldsymbol{p}_q^r - \boldsymbol{p}_v^r), \tag{2}$$

where the vector $\boldsymbol{e}_{pos}^q$ is expressed in $\mathcal{F}_q$.

On the other side, the orientation error is conceived as the discrepancy between the (3×3) identity matrix $\boldsymbol{I}_3$ and the rotation matrix obtained by

$$(\hat{\boldsymbol{R}}_q^{\mathrm{T}}\hat{\boldsymbol{R}}_v)^{\mathrm{T}}\boldsymbol{R}_v^q, \tag{3}$$

where $\boldsymbol{R}_v^q$ is the rotation matrix expressing the relative orientation between $\mathcal{F}_v$ and $\mathcal{F}_q$, measured in the real world.

In the validation panel in Figure 4b the x, y and z axes position errors are displayed in meters, and the rotation errors about the same axes, expressed in Euler angles, in degrees. The calibration procedure must be repeated each time the application is started.

Tests have been carried out to validate the algorithm. In particular, out of a total of 25 calibration executions, the norm of average errors is 5.289 mm in position and 0.991° in orientation, while the standard deviation is 2.766 mm and 0.698° for position and orientation, respectively. These data can be fitted by Gaussian distributions, whose means and standard deviations are reported in Table 2. The proposed calibration procedure provides clear improvements with respect to the one mentioned in [24], which presents a positioning error of 9 mm and 3.1° in orientation, achieved in an average running time of 23.1 s, much greater than the 5 s average time of the aforementioned procedure.

Table 2. Average and standard deviation error on 25 calibration executions.

	Average	**Standard Deviation**
x axis [mm]	−1.094	0.475
y axis [mm]	−4.279	0.949
z axis [mm]	2.910	2.555
x axis [deg]	0.434	0.468
y axis [deg]	−0.381	0.403
z axis [deg]	−0.805	0.326

5.2. Robot State

Once the robot hologram has been properly positioned in the AR environment, the virtual joint positions must be updated in real time to trace the movements of the corresponding real twin. To this aim, a script fetching data from ROS by subscribing to the topic `/joint_states` has been developed. This topic provides the exchange of JointState messages [32], containing information about the position, $\boldsymbol{q}$, velocity, $\dot{\boldsymbol{q}}$, and torque, $\boldsymbol{\tau}$, of the joints. Based on the acquired data, the script updates the robot hologram for each frame through Unity's *Update()* trigger.

Consider the teaching-by-showing technique for robot programming: while the operator guides the cobot manually along the desired motion path, the data read by joint position transducers are stored and can be played back [31]. As this action could be performed also by an operator without technical knowledge, the teaching-by-showing technique is widespread in industrial field. During the training for the task, it should be useful to inform the operator about the status of the robot joints in terms of the distance for their physical limits ($\boldsymbol{q}_{min}$ and $\boldsymbol{q}_{max}$). In detail, a sphere per virtual joint is displayed in AR environment and aligned to the relative robot joint. As can be seen in Figure 5, the i-th sphere (with $i = 1, \ldots, 7$) is characterized by variable colors to inform the operator about the joint status:

- green, if the i-th joint position is away from the limit, i.e.,

$$q_{min_i} + 10^\circ < q_i < q_{max_i} - 10^\circ; \tag{4}$$

- yellow, if the i-th joint position approaches the limit, i.e.,

$$q_{min_i} + 5^\circ < q_i \leq q_{min_i} + 10^\circ \quad \text{OR} \quad q_{max_i} - 10^\circ \leq q_i < q_{max_i} - 5^\circ; \tag{5}$$

- red, if the i-th joint position is very close or equal to the limit

$$q_{min_i} \leq q_i \leq q_{min_i} + 5^\circ \quad \text{OR} \quad q_{max_i} - 5^\circ \leq q_i \leq q_{max_i}. \tag{6}$$

In other words, the color switches from green to yellow if the joint position is less than 10 degrees far from the physical limit, whilst from yellow to red when it is less than 5 degrees from it.

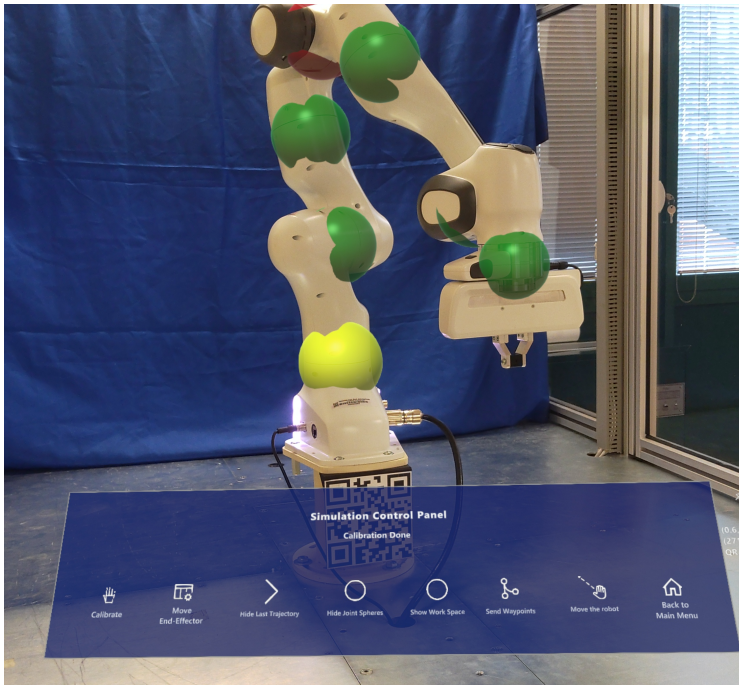
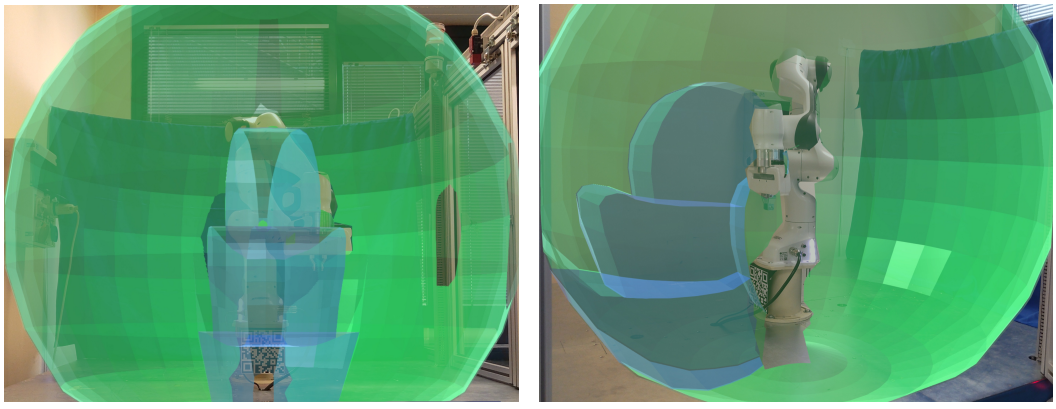


Figure 5. Joint spheres displayed after calibration.

The joints spheres can be activated or deactivated through the *Show/Hide Joint Spheres* button.

5.3. Robot Workspace Visualization

The adoption of AR allows to visualize the workspace of the cobot. Starting from the information related to joints' limits in Franka Emika Panda datasheet [9], the reachable space has been reconstructed in a 3D model through CAD software, obtaining an envelope surface which represents the border points achievable by the robot end-effector (see Figure 6). Then, the 3D surface has been discretized adopting a triangle mesh: the maximum and minimum size of each element has been set as a trade-off between the accuracy and the computational burden. The user can decide whether to show the robot workspace in AR environment by selecting *Show/Hide Workspace* on the control panel.



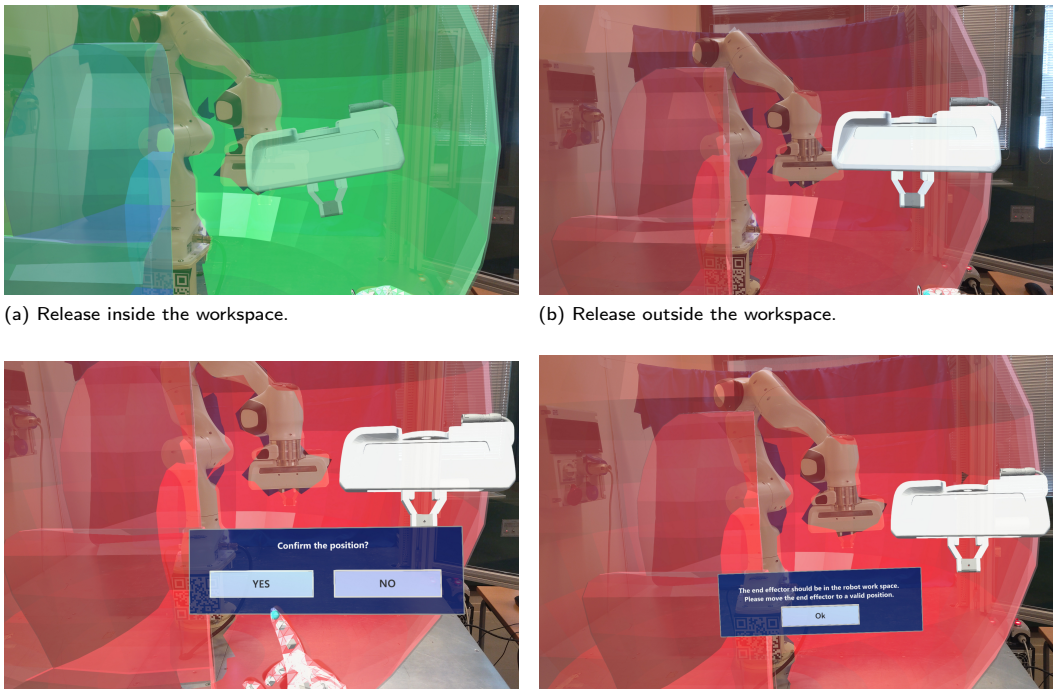
(a) Rear view of the robot and its workspace (b) Lateral view of the robot and its workspace

Figure 6. Robot workspace from two different point of view: the reachable space is shown in green.

The visualization of the workspace offers a visual help to the user to locate the goal position of the end-effector. More in detail, the user can grab the end-effector hologram and move it in any position of the AR environment to assign a new trajectory to the robot.

Regardless of whether the robot workspace is visible, a check over trajectory feasibility is always executed. Indeed, if the operator releases the end-effector hologram outside

the reachable space (Figure 7b), the entire workspace is displayed in red in order to warn her/him about the uncorrected position. If the user accepts an unfeasible position of the end-effector, even in the presence of the red workspace (Figure 7c), a panel appears advising her/him to modify the goal point (Figure 7d).



(a) Release inside the workspace.

(b) Release outside the workspace.

(c) Unfeasible release acceptance.

(d) Warning panel

Figure 7. End-effector desired position check.

5.4. Trajectory Generation

Once the goal position of the end-effector is correctly assigned to the cobot, the path and its time law have to be generated. First of all, the trajectory is computed in a ROS node and then it is sent to Unity in order to be visualized in the AR environment. When the operator presses the "*Generate Trajectory*" button, HoloLens sends a message on the topic `/event_start` and a linear point-to-point motion in the operational space is planned.

$$\boldsymbol{x}_d(s) = \boldsymbol{x}_0 + \frac{s(t)}{\|\boldsymbol{x}_f - \boldsymbol{x}_0\|}(\boldsymbol{x}_f - \boldsymbol{x}_0). \tag{7}$$

The (6×1) vector $\boldsymbol{x}_d$ represents the desired value of the end-effector pose $\boldsymbol{x}$ (i.e., position and orientation):

$$\boldsymbol{x} = [p_x, \quad p_y, \quad p_z, \quad \phi, \quad \theta, \quad \psi]^{\mathrm{T}}, \tag{8}$$

where the position components are expressed in the robot base coordinate frame and the orientation is expressed via a triple of Euler angles (e.g., roll-pitch-yaw).

In (7), $\boldsymbol{x}_0$ and $\boldsymbol{x}_f$ are the initial and final end-effector pose, respectively, and $s(t)$ is the abscissa which represents the time law of the path, planned with a fifth-order polynomial function, as

$$s(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5. \tag{9}$$

The desired final pose x_f corresponds to the end-effector pose defined by the user. Figure 8 shows the displayed trajectory and the end-effector hologram at the final point.

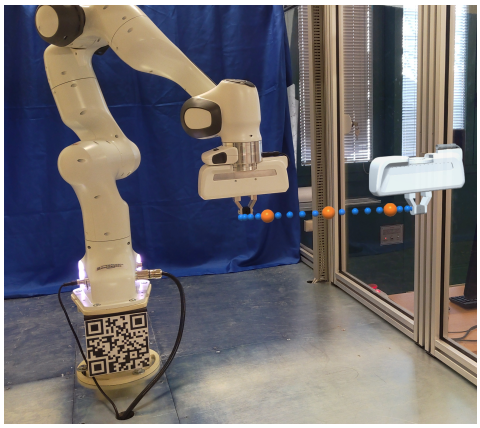


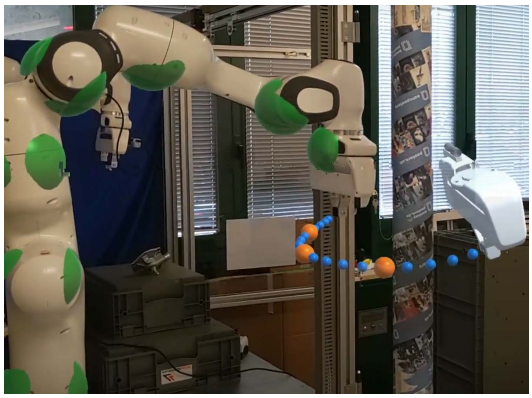
Figure 8. Linear point-to-point trajectory generation in AR.

Finally, the trajectory can be disabled/enabled by the "*Show/Hide Last Trajectory*" button.

5.5. Trajectory Validation

After the generation of the linear point-to-point trajectory, the user can suitably modify the end-effector path so as, e.g., to avoid collisions with obstacles (see Figure 9a). To this aim, among the way-points visualized in the AR environment, the orange ones can be dragged by the user in order to design a new path by keeping constant the initial and final poses. The new positions of the orange way-points are sent to the ROS node, which computes a cubic interpolating spline designing the new path, as shown in Figure 9b, by pressing the "*Send Way-points*" button.

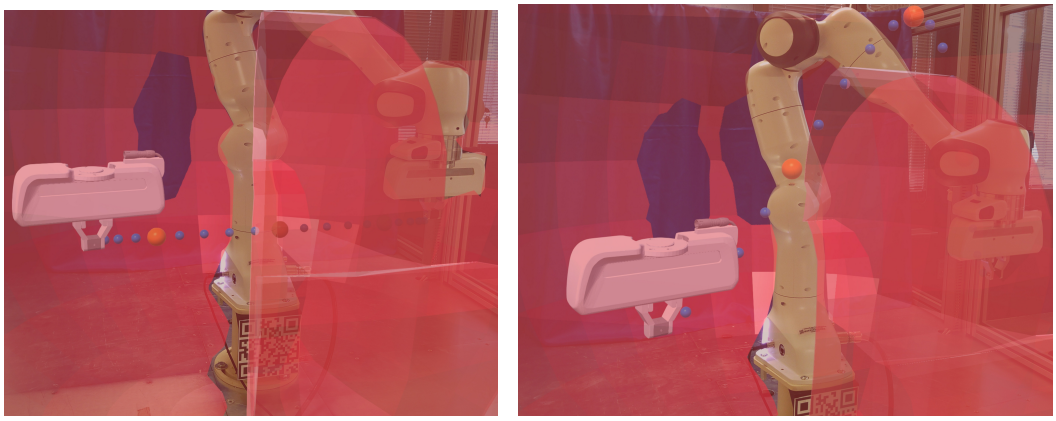
(a) Possible obstacle collision.



(b) Modified trajectory around the obstacle.

Figure 9. Trajectory modification in presence of an obstacle.

This feature becomes crucial when, despite the desired final position of the end-effector has been chosen inside the reachable space, some way-points of the computed trajectory are located outside the cobot workspace, as visualized in Figure 10.



(a) Several way-points outside the WS. (b) Only one way-point outside the WS.

Figure 10. Uncorrected trajectory generation in AR.

5.6. Trajectory Execution

Once the user checks over the trajectory feasibility, as in Figure 11, a closed loop inverse kinematics (CLIK) algorithm [31] is implemented in order to compute the motion in terms of reference velocity of each joint

$$\dot{\boldsymbol{q}}_r = \boldsymbol{J}^{\dagger}(\boldsymbol{q})(\dot{\boldsymbol{x}}_d + \boldsymbol{K}(\boldsymbol{x}_d - \boldsymbol{x})), \tag{10}$$

where $\boldsymbol{J}^{\dagger}(\boldsymbol{q})$ is the right pseudo-inverse of the robot Jacobian matrix [31] and $\boldsymbol{K}$ is a (6×6) matrix of positive gains.

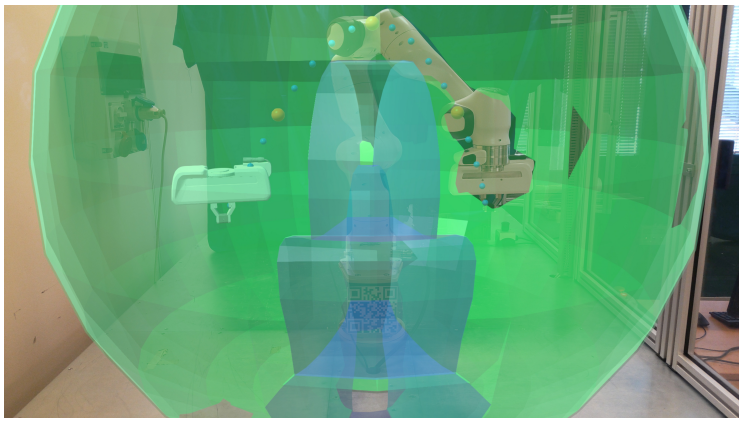
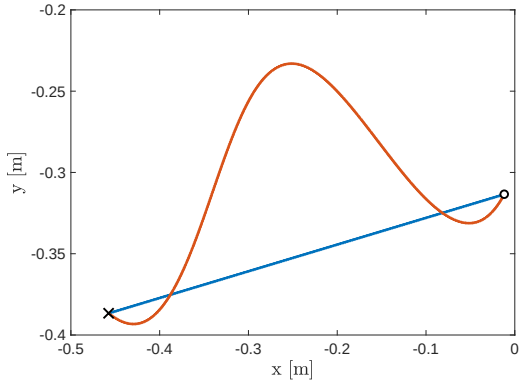
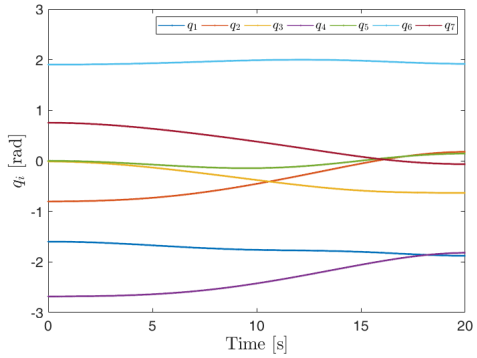


Figure 11. Trajectory validation with WS visualization.

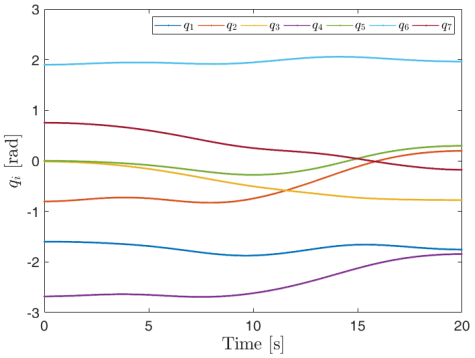
Finally, if the user presses the button "*Move robot*", the robot moves along the desired path. An example of the robot path, in terms of the end-effector position in the plane xy (under the assumption of z coordinate constant), is shown in Figure 12a. In detail, the blue line represents the linear point-to-point trajectory, while the red line represents the modified trajectory where it is assumed the presence of an obstacle. Figure 12b,c show the joints' positions for the linear and modified trajectories, respectively. The joints' velocities for the two cases are depicted in Figure 12d,e.



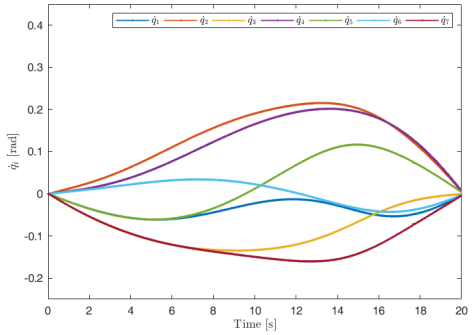
(a) Linear point-to-point trajectory (blue line) compared with the modified trajectory (red line). The circle represents the starting point, while the cross represents the final point.



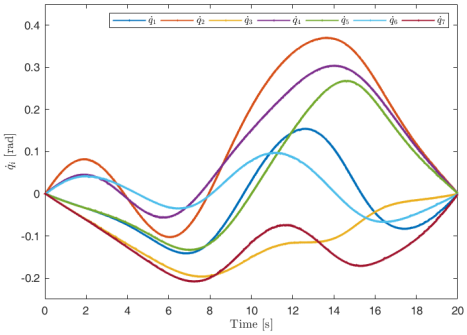
(b) Joints' positions for the linear point-to-point trajectory.



(c) Joints' positions for the modified trajectory.



(d) Joints' velocities for the linear point-to-point trajectory.



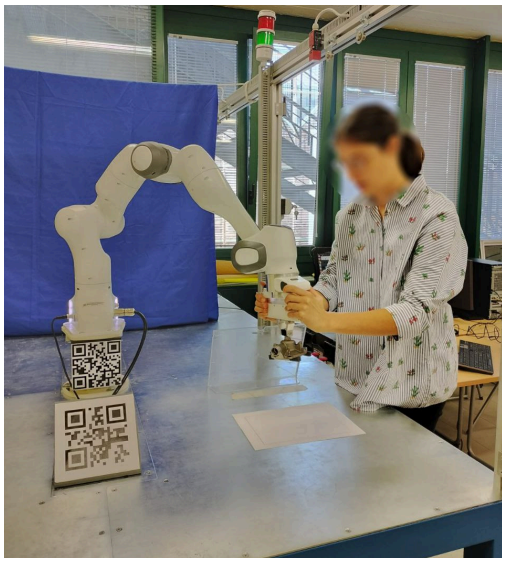
(e) Joints' velocities for the modified trajectory.

Figure 12. Example of trajectories executed by the robot.

6. Experiments

A usability study of the AR platform involving 45 participants with different ages and expertise in robot programming and Extended Reality (XR) platforms has been conducted. Two programming methods have been proposed to the user: a kinesthetic teaching interface, provided by the Franka Emika Panda cobot, and the presented AR platform. More in detail, in the first case, users were asked to interact with the cobot in the handling guide mode by physically moving and positioning it, whereas in the latter case, participants wore HoloLens 2 and interacted with the AR environment (see Figure 13).

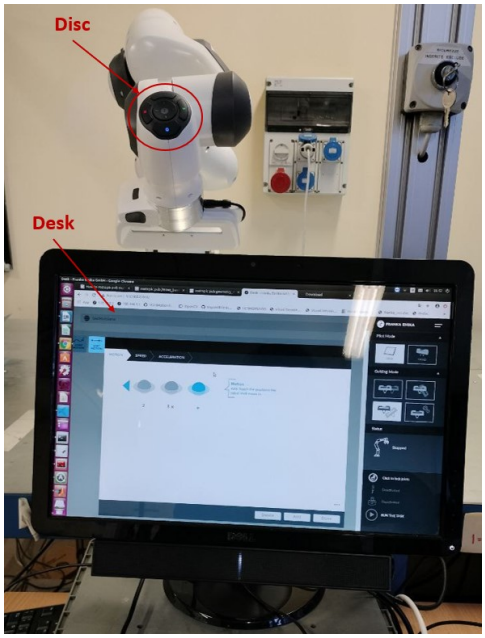
(a) Setup of the AR interface.



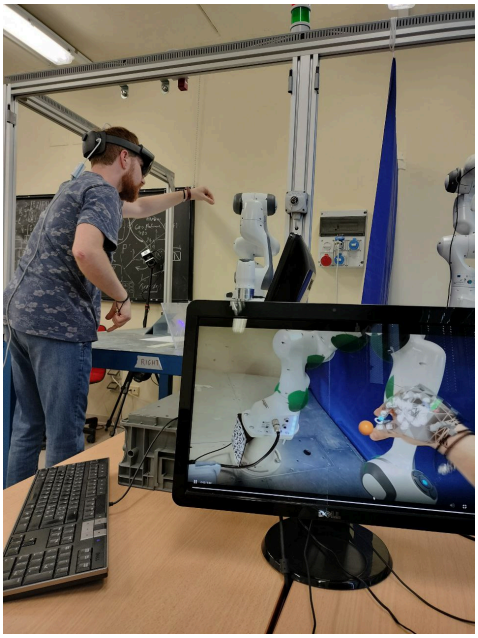
(b) Setup of the kinesthetic interface.

Figure 13. Setup of both the AR and the kinesthetic interface.

Prior to carrying out the experiments, each participant signed a consent form and she/he underwent a preliminary training phase on how to safely operate the two interfaces. In the kinesthetic teaching scenario, the participant was instructed on how to physically move the robot in hand-guiding mode using the guiding buttons on the last joint of the cobot, save way-points using the cobot Disc and check their storage on the Desk (see Figure 14a).



(a) The kinesthetic interface: the Desk is the web-Based programming interface, the Disc is used to interact with both the cobot and the Desk.



(b) The AR interface: user's point of view within HoloLens is displayed in the monitor.

Figure 14. The AR and the kinesthetic interfaces.

Conversely, in the AR-robotic interface, the participant was instructed on how to interact with the Simulation Control panel, visualize the cobot workspace, assign and modify a trajectory, and, finally, move the cobot, as displayed in Figure 14b.

6.1. Task

The task can be divided in two steps:

1. firstly, each participant was asked to position a blank sheet, with the placement target area drawn on it, on the working table, on the opposite side of the pick position with respect to a vertical obstacle;
2. secondly, starting from the pick pose with the object (a component used in the automotive industry) already gripped, the robot had to be moved by generating a trajectory so that it could finally release the object onto the target area, overcoming or bypassing the obstacle.

The presence of the obstacle is intended to force the user to generate a curved trajectory to avoid collision with it. The two tasks have been executed in a fixed sequence: firstly adopting the kinesthetic interface and, then, with the AR platform, since the latter would have provided participants the information about the robot workspace, invalidating the first step of the task.

After completing the task with both the kinesthetic and the AR methods, participants filled out the "raw" NASA Task Load Index [33] and the System Usability Scale [14] surveys: regardless of the accomplishment of the task, users were asked to assess the proposed methods used for locating the target area and generating the desired trajectory.

6.2. Surveys

The NASA Task Load Index (NASA TLX) is a tool for determining a subjective mental workload (MWL) assessment of a participant after completing a task. The overall workload index is measured after rating performance across six dimensions: mental demand, physical demand, temporal demand, effort, performance and frustration. Each subscale is accompanied by a corresponding question for clarification purposes. Participants are asked to self-rate their score on an interval scale ranging from 0 to 100, i.e., from very low to very high demand, respectively. The TLX also should employ a paired comparisons procedure: users should determinate the most effect on the workload during the task under analysis by 15 pairwise combinations between each subscale. Nevertheless, for this experimental campaign, it was preferred not to include these pairwise comparisons in order to reduce the burden on the questionnaire and to utilize the raw data, which was deemed more sensitive [33].

The System Usability Scale (SUS) is a reliable tool that can be used for global assessments of systems usability [14]. It consists of a questionnaire including ten items, employing a five-point Likert-type scale [34] ranging from "Strongly Disagree" to "Strongly Agree", which correspond to the values 1 to 5, respectively. The statements are arranged in such a manner that the odd ones are pointing towards a greater usability of the proposed system, whilst the even statements suggest the opposite. They encompass different aspects, including the effectiveness and efficiency of the system, as well as the satisfaction of the users. The SUS total rating is then multiplied by an appropriate factor to convert it to a range of 0 to 100.

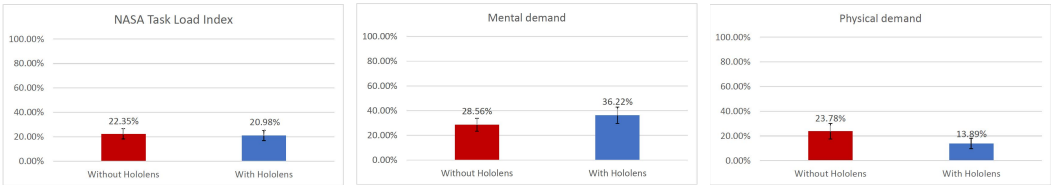
6.3. Analysis of the Results

Paired-sample t-tests have been conducted on participants' reported data obtained from the NASA TLX and the System Usability Scale. In this section, the results of these tests are reported in terms of $t(44)$ and p, which are the t-statistic value of a paired 44-samples t-test and the p-value, respectively, along with the effect size expressed in terms of Cohen's d [35]. A correction factor for samples with a cardinality of less than 50 has been taken into account in the calculation of the effect size.

6.3.1. NASA TLX Results

In Figure 15 the main results of the NASA TLX survey are reported. Statistically significant differences have been found in the mental and physical demands between the two interfaces. The kinesthetic system appears to require less mental effort ($Mean = 28.56\%$,

Standard Deviation = 17.44%) than the use of the AR system ($M = 36.22\%$, $SD = 22.19\%$), $t(44) = 2.8350$, $p = 0.0069$, $d = 0.3690$, as depicted in Figure 15b.



(a) NASA TLX for kinesthetic and AR interfaces.

(b) Mental demand for kinesthetic and AR interfaces.

(c) Physical demand for kinesthetic and AR interfaces.

Figure 15. NASA TLX, mental demand and physical demand for kinesthetic and AR interfaces; error bars represent 95% confidence intervals (CIs).

The mental workload required by the AR platform is strictly dependent on the interaction skills with holograms, as well as related to the ability to remember the specific sequence of buttons; on the other hand, the kinesthetic teaching interface provided by the cobot is extremely intuitive and user-friendly. It is not hard to imagine that an XR novice would tackle challenges when interacting with the virtual reality features. An appropriate evaluation should consider the different background of the participants in relation to XR devices. To this aim, Figure 16 shows the comparison between upper-intermediate users' assessments on mental workload (14 among 45 participants, depicted in grey) compared to those of participants with lower levels of AR expertise (31 among 45, in blue). The difference between the two samples in the case of the kinesthetic interface is almost negligible, while, in the case of the AR platform, a great discrepancy is experienced: the mental workload points out a sensible increment for the non-expert users, whilst for the expert users it is comparable with the other interface. Thus, one could argue that the mental load requested by the AR platform can be easily reduced with a more intensive training phase on the HoloLens device, as it is rather due to the participants' inexperience than to the complexity of the proposed application.

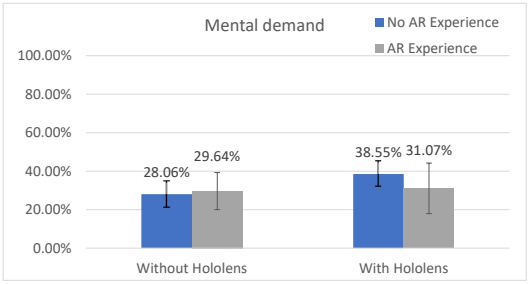


Figure 16. Mental demand for kinesthetic and AR interfaces, with comparison between different levels of AR expertise; error bars represent 95% CIs.

On the other hand, as reported in Figure 15c, the kinesthetic interface necessitates more physical demands ($M = 23.78\%$, $SD = 20.84\%$) than the AR interface ($M = 13.89\%$, $SD = 13.94\%$), $t(44) = 3.7025$, $p = 0.0006$, $d = 0.5358$, as evidenced by a larger effect size.

It is an expected result since the kinesthetic teaching method requires users to physically move the robot to each way-point of the desired path, whereas the AR-robotic interface does not need physical interaction.

There are no other noteworthy variances in the other dimensions. Nonetheless, the adoption of the AR platform results in decreased temporal demand and effort while yielding better performance levels (85.89% without the HoloLens platform, 88.44% with the HoloLens). Although it does not represent a statistically significant difference, the overall NASA TLX, depicted in Figure 15a is slightly decreased with the AR interface (−6.13%).

6.3.2. SUS Results

The effectiveness of the proposed platform can be distinctly assessed through the analysis of the SUS, whose main results are reported in Figures 17 and 18.

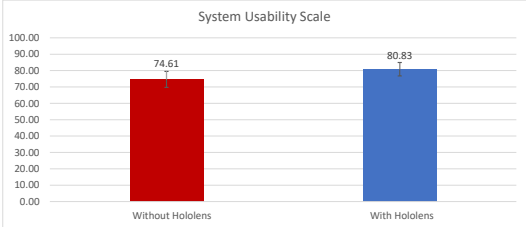
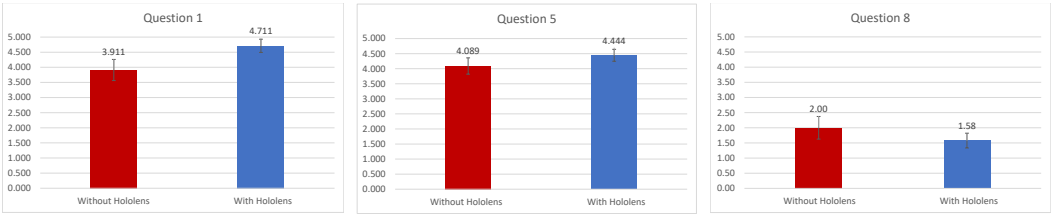


Figure 17. SUS scores for kinesthetic and AR interfaces; error bars represent 95% CIs.



(a) SUS first question scores for kinesthetic and AR interfaces.

(b) SUS fifth question scores for kinesthetic and AR interfaces.

(c) SUS eighth question scores for kinesthetic and AR interfaces.

Figure 18. Main statistically significant results of the SUS survey; error bars represent 95% CIs.

Statistically significant differences have been found in the SUS overall score between the two interfaces: the AR system ($M = 80.83\%$, $SD = 13.65\%$) surpasses the kinesthetic system ($M = 74.61\%$, $SD = 16.24\%$) by 6.22% in terms of usability, $t(44) = 2.09$, $p = 0.043$, $d = 0.3984$, as depicted in Figure 17. The overall SUS score has corroborated that the Franka Emika Panda robot is designed to be a collaborative robot with a user-friendly kinesthetic teaching interface, which is much more intuitive than the industrial robots' teach pendant. Hence, it is noteworthy that the improvement in usability achieved by the proposed AR platform could be further increased in industrial scenarios.

In this regard, the results concerning the first SUS statement (*"I think that I would like to use this system frequently"*), shown in Figure 18a, prove that users' preference for the HoloLens platform increases by about 16% compared to the classical programming interface ($M = 4.71$, $SD = 2.17$ for the AR system, $M = 3.91$, $SD = 1.98$ for kinesthetic interface, $t(44) = 3.769$, $p = 0.0005$), with a large effect size of $d = 0.7916$.

Furthermore, the proposed AR interface satisfies the requirements of efficiency and consistency. This is evidenced by the scores of statements 5 (*"I found the various functions in this system were well integrated"*), Figure 18b, and 8 (*"I found the system very cumbersome to use"*), Figure 18c. Indeed, statistically significant differences have been found for the statement 5 ($M = 4.44$, $SD = 0.659$ for the AR system, $M = 4.089$, $SD = 0.90$ for kinesthetic interface, $t(44) = 2.701$, $p = 0.01$, $d = 0.4329$) and the statement 8 ($M = 2.00$, $SD = 1.24$ for the AR system, $M = 1.58$, $SD = 0.81$ for kinesthetic interface, $t(44) = 2.06$, $p = 0.045$, $d = 0.3863$).

Although it does not represent a statistically significant difference, the statement 9, concerning users' confidence during task execution, reveals that each participant felt safer by the HoloLens platform of about 4.4%. To this purpose, it has been experienced that visualizing the trajectory in the AR environment as a set of way-points significantly boosts users' confidence.

7. Conclusions

This work deals with the development of an AR platform aimed at allowing the bi-directional interaction between a human operator and a collaborative robot through a HoloLens 2 device, supplied by the Automation, Robotics and Applied Electromagnetism Laboratory (AREA) and the Mechanical Design and Advanced Engineering Methods

(MEDEA) of University of Basilicata. The work builds on the application presented in [8], whose main features consisted of communicating the robot status via a visual feedback in the AR environment and the robot intent by showing the planned end-effector trajectory to the operator. The novel contributions allow for a human-to-robot communication to be established by implementing the visualization of the robot workspace and the possibility to manipulate the trajectory by moving some way-points in the AR environment. Moreover, a safety system checks over the trajectory feasibility by taking into account the workspace limits. According to the authors' best knowledge, an AR platform that implements exactly the same features is not available in the literature. To this aim, a detailed comparative analysis cannot be performed. However, we have conducted a qualitative comparison with similar methods (see Table 3) taking into account the most relevant functionalities.

Table 3. Comparative analysis of the proposed platform with other approaches in the literature.

	Proposed Platform	**Ostanin et al. [24]**	**De Franco et al. [5]**	**Rosen et al. [21]**	**Kastner et al. [23]**	**Sun et al. [18]**	**Quintero et al. [18]**
Show joints' limit	Yes	Yes	Yes	No	No	No	No
Show WS	Yes	Yes	No	No	N.A.	Yes	No
Show Trajectory	Yes	Yes	No	Yes	Yes	No	Yes
Modify Trajectory	Yes	No	No	Yes	No	No	Yes
Traj. planning	Yes	Yes	No	Yes	Yes	Yes	Yes
Show forces	No	No	Yes	No	No	Yes	No
Obstacle avoid.	Yes	Yes	No	Yes	Yes	No	Yes
Show calibr. errors	Yes	Yes	No	No	No	No	No

A usability test has been conducted through a wide experimental campaign involving 45 voluntary users with different expertise levels, in which the proposed application is compared to a classical kinesthetic teaching task. In this campaign, the NASA TLX survey has showed a lower physical workload for the AR platform despite of the mental one. In case of XR-skilled users, the cognitive workload exhibited similar values for both kinesthetic and AR interfaces. Moreover, the System Usability Scale Questionnaire showed that users prefer the AR interface with respect to the kinestetic one. Moreover, the users felt safer using the AR platform by about 4.4%. Furthermore, it has been discovered that visualizing the trajectory in the AR environment as a series of way-points was a key element in increasing user confidence.

Future works will be devoted to enhancing safety and users' confidence during HRI through the real-time construction of virtual barriers that demarcate the boundaries of human and robot workspaces: if the robot attempts to cross this barrier, its speed is expected to decrease. Further research efforts will be focused on supporting complex tasks as the cooperative human-robot object transportation, managing information about the exchanged forces between the robot and the object. Other studies will be needed to reduce the mental workload by mean of optimization the HMD field of view. Moreover, collision avoidance algorithms in the AR environment will be implemented. At the end of the process, a second usability assessment will be carried out to address potential hardware and software issues.

Author Contributions: Conceptualization, G.C., N.C., M.S., F.P., F.C. and R.M.; Methodology, G.C., N.C., M.S., F.P., F.C. and R.M.; Validation, G.C., N.C. and M.S.; Writing—original draft, G.C., N.C. and M.S.; Writing—review & editing, F.P., F.C. and R.M.; Supervision, F.P., F.C. and R.M.; Project administration, F.P. and R.M.; Funding acquisition, F.P., F.C. and R.M.; Experimental validation: G.C. and M.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been supported by the Italian Ministry of University and Research under the grant PRIN 2022 COM^3 (COoperative Mobile Manipulators for Manufacturing) n.2022ZLYBF5.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Informed consent was obtained from all participants involved in the experimental study.

Acknowledgments: The authors would like to thank Rocco Pio Cortese, Simone Basso and Donato Gabriele Carretta for their help with software implementation. Moreover, authors wish to thank all the participants of the experiment.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

API	Application Programming Interface
AR	Augmented Reality
CAD	Computer Aided Design
CI	Confidence Interval
CLIK	Closed Loop Inverse Kinematics
HMD	Head-Mounted Display
HPU	Holographic Processing Unit
HRC	Human-Robot Collaboration
HRI	Human-Robot Interaction
IR	Infrared
JSON	JavaScript Object Notation
M	Mean
MRTK	Mixed Reality Toolkit
QR code	Quick Response code
RGB-D	Red, Green, Blue and Depth
ROS	Robot Operating System
SD	Standard Deviation
SUS	System Usability Scale
TLX	Task Load Index
UDP	User Datagram Protocol
URDF	Unified Robot Description Format
UWP	Universal Windows Platform
VR	Virtual Reality
WS	Workspace
XR	Extended Reality

List of Symbols

The following symbols are used in this manuscript:

$\mathcal{F}_r$	Robot base coordinate frame
$\mathcal{F}_0$	HoloLens coordinate frame
$\mathcal{F}_q$	Coordinate frame of the first QR code
$\mathcal{F}_v$	Coordinate frame of the second QR code
O_q, O_v	Origins of the coordinates frames $\mathcal{F}_q$ and $\mathcal{F}_v$
x_q, y_q, z_q	Unit vectors of the coordinate frame $\mathcal{F}_q$
x_v, y_v, z_v	Unit vectors of the coordinate frame $\mathcal{F}_v$
$\hat{\mathcal{F}}_q$	Estimated coordinate frame of the first QR code
$\hat{\mathcal{F}}_v$	Estimated coordinate frame of the second QR code
$\boldsymbol{p}_q^r$	Relative position between $\mathcal{F}_q$ and $\mathcal{F}_r$
$\boldsymbol{R}_q^r$	Relative orientation between $\mathcal{F}_q$ and $\mathcal{F}_r$
$\boldsymbol{R}_r^q$	Relative orientation between $\mathcal{F}_r$ and $\mathcal{F}_q$
$\boldsymbol{p}_v^r$	Relative position between $\mathcal{F}_v$ and $\mathcal{F}_r$
$\boldsymbol{R}_v^r$	Relative orientation between $\mathcal{F}_v$ and $\mathcal{F}_r$
$\boldsymbol{R}_v^q$	Relative orientation between $\mathcal{F}_v$ and $\mathcal{F}_q$
$\hat{\boldsymbol{p}}_q$	Estimated position of $\mathcal{F}_q$ expressed in $\mathcal{F}_0$
$\hat{\boldsymbol{R}}_q$	Estimated orientation of $\mathcal{F}_q$ expressed in $\mathcal{F}_0$
$\hat{\boldsymbol{p}}_v$	Estimated position of $\mathcal{F}_v$ expressed in $\mathcal{F}_0$
$\hat{\boldsymbol{R}}_v$	Estimated orientation of $\mathcal{F}_v$ expressed in $\mathcal{F}_0$
$\boldsymbol{I}_3$	Identity matrix of dimension (3×3)
$\boldsymbol{e}_{pos}^q$	Vector of the position errors expressed in $\mathcal{F}_q$
$\boldsymbol{q}$	Vector of the joint positions

$\dot{\boldsymbol{q}}$	Vector of the joint velocities
$\boldsymbol{\tau}$	Vector of the joint torques
$\boldsymbol{q}_{min}$, $\boldsymbol{q}_{max}$	Vector of the joint limits
q_i	The i-th joint
q_{min_i}, q_{max_i}	Limits of the i-th joint
$s(t)$	Abscissa function representing the time law of the path
$\boldsymbol{x}$	End-effector pose
$\boldsymbol{x}_d$	Desired end-effector pose
$\dot{\boldsymbol{x}}_d$	Desired end-effector velocity
$\boldsymbol{x}_0$	Initial end-effector pose
$\boldsymbol{x}_f$	Final end-effector pose
p_x, p_y, p_z	Position components of the end-effector pose
ϕ, θ, ψ	Orientation components of the end-effector pose expressed in Euler angles
a_i	The i-th coefficient of the polynomial function
$\dot{\boldsymbol{q}}_r$	Vector of the reference velocities of the joints
$\boldsymbol{K}$	Positive definite matrix gain
$\boldsymbol{J}^{\dagger}(\boldsymbol{q})$	Right pseudo-inverse of the robot Jacobian matrix
$t(44)$	t-statistic value of a paired 44-samples t-test
p	p-value
d	Cohen's d effect size measure

References

1. Salvato, R.; Marra, G.; Scardamaglia, P.; Di Gironimo, G.; Marzullo, D.; Mozzillo, R. Design and integration of automation systems with manual operation: small and medium enterprises issues. In Proceedings of the Second International Conference on Design Tools and Methods in Industrial Engineering, ADM 2021, Rome, Italy, 9–10 September 2021; Springer: Berlin/Heidelberg, Germany, 2022; pp. 298–307.
2. Iaccarino, P.; Inserra, S.; Cerreta, P.; Mozzillo, R. Determinant assembly approach for flat-shaped airframe components. *Int. J. Adv. Manuf. Technol.* **2020**, *108*, 2433–2443. [CrossRef]
3. Sheridan, T.B. *Humans and Automation: System Design and Research Issues*; J. Wiley ans Sons: Hoboken, NJ, USA, 2002; Volume 280.
4. Weiss, A.; Wortmeier, A.K.; Kubicek, B. Cobots in industry 4.0: A roadmap for future practice studies on human–robot collaboration. *IEEE Trans. Hum. Mach. Syst.* **2021**, *51*, 335–345. [CrossRef]
5. De Franco, A.; Lamon, E.; Balatti, P.; De Momi, E.; Ajoudani, A. An Intuitive augmented reality interface for task scheduling, monitoring, and work performance improvement in human-robot collaboration. In Proceedings of the 2019 IEEE International Work Conference on Bioinspired Intelligence (IWOBI), Budapest, Hungary, 3–5 July 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 75–80.
6. Green, S.A.; Billinghurst, M.; Chen, X.; Chase, J.G. Human-robot collaboration: A literature review and augmented reality approach in design. *Int. J. Adv. Robot. Syst.* **2008**, *5*, 1. [CrossRef]
7. HoloLens 2. Available online: https://www.microsoft.com/en-us/hololens/hardware (accessed on 5 July 2023).
8. Calzone, N.; Sileo, M.; Mozzillo, R.; Pierri, F.; Caccavale, F. Mixed Reality Platform Supporting Human-Robot Interaction. In Proceedings of the International Joint Conference on Mechanics, Design Engineering & Advanced Manufacturing, JCM 2022, Ischia, Italy, 1–3 June 2022; Springer: Berlin/Heidelberg, Germany, 2022; pp. 1172–1182.
9. Franka Emika Panda. Available online: https://www.franka.de/ (accessed on 11 July 2023).
10. Unity. Available online: https://unity.com/ (accessed on 29 June 2023).
11. Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A.Y. ROS: An open-source Robot Operating System. In Proceedings of the ICRA Workshop on Open Source Software, Kobe, Japan, 12–17 May 2009; Volume 3, p. 5.
12. GitHub. ROS#. Available online: https://github.com/siemens/ros-sharp (accessed on 6 July 2023).
13. Suzuki, R.; Karim, A.; Xia, T.; Hedayati, H.; Marquardt, N. Augmented Reality and Robotics: A Survey and Taxonomy for AR-Enhanced Human-Robot Interaction and Robotic Interfaces. In Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems, CHI '22, New York, NY, USA, 29 April–5 May 2022. [CrossRef]
14. Brooke, J. SUS: A quick and dirty usability scale. In *Usability Evaluation in Industry*; CRC Press: Boca Raton, FL, USA, 1996; Volume 189, pp. 189–194.
15. Hart, S.G.; Staveland, L.E. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In *Advances in Psychology*; Elsevier: Amsterdam, The Netherlands, 1988; Volume 52, pp. 139–183.
16. Romero, D.; Stahre, J.; Wuest, T.; Noran, O.; Bernus, P.; Fast-Berglund, Å.; Gorecky, D. Towards an operator 4.0 typology: A human-centric perspective on the fourth industrial revolution technologies. In Proceedings of the International Conference on Computers and Industrial Engineering (CIE46), Tianjin, China, 29–31 October 2016; pp. 29–31.
17. Guo, L.; Lu, Z.; Yao, L. Human-machine interaction sensing technology based on hand gesture recognition: A review. *IEEE Trans. Hum. Mach. Syst.* **2021**, *51*, 300–309. [CrossRef]

18. Sun, D.; Kiselev, A.; Liao, Q.; Stoyanov, T.; Loutfi, A. A new mixed-reality-based teleoperation system for telepresence and maneuverability enhancement. *IEEE Trans. Hum. Mach. Syst.* **2020**, *50*, 55–67. [CrossRef]
19. Microsoft Mixed Reality Toolkit. Available online: https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/?view=mrtkunity-2021-05 (accessed on 5 July 2023).
20. Rosen, E.; Whitney, D.; Phillips, E.; Chien, G.; Tompkin, J.; Konidaris, G.; Tellex, S. Communicating and controlling robot arm motion intent through mixed-reality head-mounted displays. *Int. J. Robot. Res.* **2019**, *38*, 1513–1526. [CrossRef]
21. Gadre, S.Y.; Rosen, E.; Chien, G.; Phillips, E.; Tellex, S.; Konidaris, G. End-User Robot Programming Using Mixed Reality. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 2707–2713. [CrossRef]
22. Quintero, C.P.; Li, S.; Pan, M.K.; Chan, W.P.; Machiel Van der Loos, H.; Croft, E. Robot Programming Through Augmented Trajectories in Augmented Reality. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 1838–1844. [CrossRef]
23. Kästner, L.; Lambrecht, J. Augmented-reality-based visualization of navigation data of mobile robots on the microsoft hololens-possibilities and limitations. In Proceedings of the 2019 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM), Bangkok, Thailand, 18–20 November 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 344–349.
24. Ostanin, M.; Mikhel, S.; Evlampiev, A.; Skvortsova, V.; Klimchik, A. Human-robot interaction for robotic manipulator programming in Mixed Reality. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 2805–2811.
25. MoveIt. Available online: https://moveit.ros.org/ (accessed on 30 June 2023).
26. GitHub. ROS# UWP. Available online: https://github.com/EricVoll/ros-sharp (accessed on 28 June 2023).
27. Andrej Orsula, Panda_ign. Available online: https://github.com/AndrejOrsula/panda_ign (accessed on 11 July 2023).
28. ROS Wiki. URDF. Available online: http://wiki.ros.org/urdf (accessed on 28 June 2023).
29. Microsoft. QR Tracking Unity Sample for HoloLens 2 Using OpenXR. Available online: https://github.com/yl-msft/QRTracking (accessed on 5 July 2023).
30. GitHub. NuGet. Available online: https://github.com/GlitchEnzo/NuGetForUnity (accessed on 12 July 2023).
31. Siciliano, B.; Sciavicco, L.; Villani, L.; Oriolo, G. *Robotics–Modelling, Planning and Control*; Springer: London, UK, 2009.
32. JointState Message. Available online: http://docs.ros.org/en/noetic/api/sensor_msgs/html/msg/JointState.html (accessed on 12 July 2023).
33. Hart, S.G. Nasa-Task Load Index (NASA-TLX); 20 Years Later. *Proc. Hum. Factors Ergon. Soc. Annu. Meet.* **2006**, *50*, 904–908. [CrossRef]
34. Sullivan, G.M.; Artino, A.R., Jr. Analyzing and interpreting data from Likert-type scales. *J. Grad. Med. Educ.* **2013**, *5*, 541–542. [CrossRef] [PubMed]
35. Cohen, J. *Statistical Power Analysis for the Behavioral Sciences*, 2nd ed.; Lawrence Erlbaum Associates, Publishers: Hillsdale, NJ, USA, 1988.