Journal of Intelligent Manufacturing
https://doi.org/10.1007/s10845-023-02096-2

# HA$R^2$bot: a human-centered augmented reality robot programming method with the awareness of cognitive load

**Wenhao Yang[1] · Qinqin Xiao[2] · Yunbo Zhang[1,3]**

Received: 15 July 2022 / Accepted: 11 February 2023


**Abstract**
In the era of Industry 4.0, manufacturing enterprises are actively adopting collaborative robots (Cobots) in their productions. Current online and offline robot programming methods are difficult to use and require extensive experience or skills. On the other hand, the manufacturing industries are experiencing a labor shortage. An essential question, therefore, is: how would a new robot programming method help novice users complete complex tasks effectively, efficiently, and intuitively? To answer this question, we proposed HA$R^2$bot, a novel human-centered augmented reality programming interface with awareness of cognitive load. Using NASA's system design theory and the cognitive load theory, a set of guidelines for designing an AR-based human-robot interaction system is obtained through a human-centered design process. Based on these guidelines, we designed and implemented a human-in-the-loop workflow with features for cognitive load management. The effectiveness and efficiency of HA$R^2$bot are verified in two complex tasks compared with existing online programming methods. We also evaluated HA$R^2$bot quantitatively and qualitatively through a user study with 16 participants. According to the user study, compared with existing methods, HA$R^2$bot has higher efficiency, a lower overall cognitive load, lower cognitive loads for each type, and higher safety.

**Keywords** Augmented reality · Human–robot interaction · Collaborative robot · Cognitive load

## Introduction

With the transforming fourth industrial revolution (Industry 4.0), industrial robot-based automation plays an essential role in manufacturing industries. Among the various types of industrial robots, the collaborative robot (Cobot), a new type of robot, has a human-size scale and works alongside human workers in a shared, collaborative workspace (Association for Advancing Automation, 2023). The adoption of Cobot enables flexible production, a lower cost deployment, and a high return on investment (ROI) (Association for Advancing Automation, 2023), which supports the growth of automation in manufacturing industries, especially small and medium-sized enterprises. Because of its importance, the global market of the Cobot is projected to grow from $1.36 billion in 2021 to $16.4 billion in 2028, according to a newly published report from Fortune Business Insights (2023). One reason for adopting the Cobot is the improvement of flexibility and efficiency in production. For example, a 2015 paper (Nikolaidis et al., 2015) indicates that human-robot collaboration achieves better performance in assembly tasks than entirely automated assembly or conventional manual assembly. Another driving force for the adoption of Cobots is the labor shortage. A recent survey showed that 54% of companies globally report talent shortages in 2019, three times higher than a decade ago (ManpowerGroup, 2023), and the shortage became even worse in 2021 due to the COVID-19 pandemics (Robotics Adoption Survey, 2021). However, there are still barriers for manufacturing industries to adopt Cobots (Aaltonen & Salmi, 2019), such as the lack of knowledge about allocating tasks between Cobots and human operators, the need for new user interfaces (UIs) and new programming methods, the concerns about safety, etc.

✉ Yunbo Zhang
ywzeie@rit.edu

1 Department of Industrial & Systems Engineering, Rochester Institute of Technology, Rochester, NY 14623, USA

2 Warner School of Education, University of Rochester, Rochester 14627, NY, USA

3 School of Information (iSchool), Rochester Institute of Technology, Rochester, NY 14623, USA

Published online: 21 March 2023

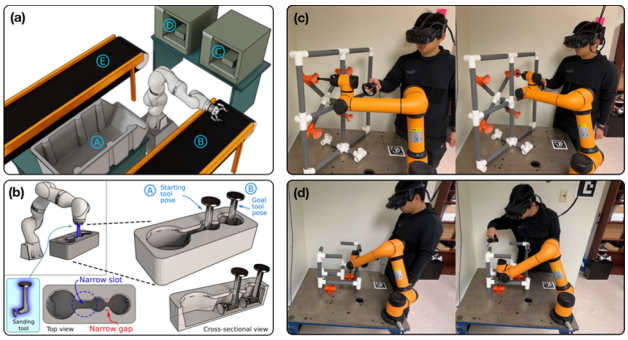


**Fig. 1** Automatic path planning methods fail to handle complex planning tasks (see **a** and **b** images courtesy of (Rajendran et al., 2019a)). Our proposed HA$R^2$bot system enabling a human-in-loop path planning procedure, successfully handles tasks with similar complexity (see **c** and **d**)

The current Cobot is operated either through offline programming or online programming (Pan et al., 2012) methods. The offline method requires the operator to input the codes and commands using an either text-based or graphical programming language. Then, motion planning is generated by planning algorithms, and verified in a simulation environment. There is a steep learning curve for operators to gain high computer programming skills. In addition, the offline method needs either Computer-aided Design (CAD) models or the 3D reconstruction of the workpiece and the environment, to generate collision-free paths. For complex planning tasks as shown in Fig. 1a and b, even with the 3D models, the offline method has a high failure rate to deal with since the complex obstacles limit the search space of the path planning algorithms. The online programming, including pendant teaching and manual leading methods (Todd, 2012), is conducted by the operator while the Cobot is actively running. Compared with offline programming, online programming keeps humans in the loop and therefore is flexible to handle complex cases, for example, in Fig. 1. Nevertheless, it still requires high shop-floor skills and extensive experience. The UIs and interactions of the online method are also required to be improved (Aaltonen & Salmi, 2019). Moreover, the safety concern is raised since the operator and the Cobot coexist in the same space without a fence or cage. Therefore, new robot programming methods are expected to address the aforementioned issues (Aaltonen & Salmi, 2019).

The Augmented Reality (AR) technique has been introduced in robot programming by augmenting human operators in the loop of robot programming. AR has the capacity to bridge the virtual and physical worlds, as it provides the perception of the physical environment and takes advantage of the richness and flexibility of virtual content. With humans in the loop, AR-based Human-Robot Interaction (HRI) has the potential to handle complex tasks (see Fig. 1). Leveraging AR, researchers have proposed a variety of interfaces for planning robot motion. For example, smartphone-based AR interfaces are developed for robot programming (Fuste et al., 2020; Thoo et al., 2021). The safety issue is emphasized in human-robot collaboration, where AR is utilized and has the potential to tackle this problem (Michalos et al., 2016; Ong et al., 2020; Tsamis et al., 2021). Researchers also noticed that human operators' perception and high-level decision-making abilities could possibly improve the efficacy and efficiency of robot programming tasks (Rajendran et al., 2019b, a). Although AR-based robot programming has become a raising research topic, there is no existing work that demonstrates the ability to deal with programming tasks with complex obstacles that mimic real-world industrial applications (as the ones shown in Fig. 1). In addition, it is unclear how the expectations from the industry stakeholders (Aaltonen & Salmi, 2019) should be addressed by an AR-based Cobot programming system. The following research question remains open—*RQ1: what are the design guidelines for the AR-based programming method from stakeholders' perspectives, and how would the AR-based method be realized based on these guidelines?*

To increase the adoption of Cobots, another recent focus is to lower the cognitive load (Sweller, 2011) in programming tasks using AR techniques (Yew et al., 2017; Cao et al., 2019). Cognitive load theory (CLT), proposed by Sweller (2011), represents the amount of working memory resources to be used when a person is in a learning process. For a complex process such as robot programming, it is essential to manage the cognitive loads of the operator, especially the novice operator. According to the CLT, there are three types of cognitive loads: *intrinsic*, *extraneous*, and *germane*. These cognitive loads are caused by different sources, and therefore, need to be managed differently (Sweller, 2011). In general, the intrinsic cognitive load, related to the learning content itself, cannot be eliminated, but only optimized. The extraneous cognitive load is related to the ways and methods of presenting the learning contents, and it needs to be reduced. The germane cognitive load, which is related to the working memory put into the learning process creating a permanent store of knowledge, or a schema benefits the information processing, is in many cases desired to increase (Sweller et al., 1998; Van Merriënboer & Sweller, 2010). Existing research works related to AR-based HRI (Yew et al., 2017; Cao et al., 2019) consider the cognitive load as a whole and lack an analysis of sources of different cognitive loads in the Cobot programming and corresponding strategies. Therefore, the following research question remains open—*RQ2: what are the sources of each type of cognitive load in the Cobot programming, and what are the strategies to manage these cognitive loads in the AR-based system?*

To answer these two research questions, we came up with a set of design guidelines based on the assessment of robot programming methods from the perspectives of stakeholders and

the cognitive load theory. With these guidelines, we proposed HA$R^2$bot, an AR-based human-robot interaction interface to keep humans in the loop of complex robot programming tasks while being aware of different types of cognitive loads. We verified the effectiveness and efficiency of HA$R^2$bot, and conducted a user study with both quantitative and qualitative evaluations of HA$R^2$bot. The main contributions can be summarized as follows:

(1) The design guidelines for the AR-based robot programming method for complex programming tasks come from both stakeholders' expectations and cognitive load theory.
(2) A novel AR-based robot programming system for supporting novice operators in performing complex tasks with the fulfillment of stakeholders' expectations and the management of different types of cognitive loads.
(3) A comprehensive user study consisting of quantitative and qualitative evaluations of HA$R^2$bot with the measurement of different types of cognitive loads.

The rest of the paper is organized as follows: Section "Related work" presents an overview of existing online robot programming methods and AR-based robot programming systems, along with an explanation of their shortcomings. Section "HA$R^2$bot system design" describes the design process and guidelines for the HA$R^2$bot. Section "System implementation" covers the system implementation and provides details about the system framework, robot programming methods, and interface design. The system verification is performed and discussed in Section "System verification", using two case studies involving two distinct complex tasks. Following that, Section "User study" presents a user study with 16 participants to assess the efficiency, usability, cognitive load, and user experience of HA$R^2$bot in comparison to existing online programming methods. Section "Conclusion and discussion" concludes the paper by summarizing the findings and highlighting several potential future directions.

## Related work

### Online robot programming

As defined by the American Occupational Safety and Health Administration, robot programming methods consist of online programming and offline programming (Occupational Safety and Health Administration, 2022). Especially for the Cobot, online methods including pendant teaching and manual leading are primarily adopted (Schraft & Meyer, 2006). With online programming, the operator is required to physically move the end-effector along a path and record several points on the path, namely waypoints. Then, these waypoints are fed to the path-planning algorithms to produce the path. The pendant teaching requires the operator to control the robot's motion and record the waypoints through the UI installed on the teaching pendant, while the manual leading relies on the operator manually moving each joint to achieve desirable locations. Although both methods are widely adopted, there are several drawbacks. First of all, using online methods in a complex environment with obstacles is very tedious and time-consuming (Pan et al., 2012). Online methods also suffer from a lack of flexibility and reusability. Even with a slight change, it is required to repeat the tedious programming process again (Pan et al., 2012). To address these shortcomings, Kohrt et al. (2013) proposed a novel cell-based Voronoi generation algorithm for online programming in industrial applications, emphasizing the programming system's maintainability and reusability. Brunete et al. (2016) presented an online walk-through path guidance using an external force/torque sensor for programming robotized machining tasks to reduce the completion time.

Recently, Gašpar et al. (2020) developed a Reconfigurable Manufacturing System (RMS) that realized robot programming by demonstrating practices to reconfigure passive hardware with robots. Gao and Huang (2019) proposed a Projection-based Augmented Table-top Interface (PATI) for programming the manipulator with common gestures and tools. Chen et al. (2019) used images to control an aerial robot by recognizing hand gestures. In order to accommodate the novice operator who lacks prior robotic experience and knowledge, the interface and programming method are evolving toward being more intuitive and assisted.

### AR-based robot programming

The Robot Programming using Augmented Reality (RPAR) techniques provides the affordance for utilizing offline programming algorithms without the need for CAD models of the workpiece and the environment (Pan et al., 2012). The PRAR also enables an in-situ simulation of the virtual robot in the real environment.

An AR interface (Thoo et al., 2021) for smartphones was developed, combining both online and offline robot programming methods. By manipulating a virtual robot in the end-effector space, that interface provides users the ability to control the robot in real-time as well as program robot tasks offline. Another earlier work (Fuste et al., 2020) also enabled robot programming on smartphones or tablets. However, these methods cannot handle the occlusion problems in the rendering since there is no depth information estimated and utilized. Besides, the interaction of these systems is limited because all four proposed control methods rely on drag-and-drop interaction on 2D planes in the AR scene. Due to the lack of occlusion rendering and limited interactions, these methods cannot handle programming tasks with com-

plex geometry and obstacles. A recent work (Pan et al., 2021) presents an AR-based teleoperation system with the RGB-D information provided by the Microsoft Kinect depth sensor. This system enables the robot's teleoperation through the handheld controller and the AR visual feedback. However, it does not demonstrate the ability to handle robot programming tasks with complex obstacles.

The safety issue is an essential limitation of prospering Cobots (Villani et al., 2018), since conventional safety standards separate humans and robots in different spaces with safety barriers. AR has the potential to enhance operators' awareness of safety by visualizing the safe zone of the robot workspace (Michalos et al., 2016). Therefore, safety is considered a paramount evaluation for AR-based robot programming systems, which has been integrated into the user experience survey as a Likert scale (Ong et al., 2020). In addition, the enhancement of user safety from AR technology contributes to the reduction of completion time and cost (Tsamis et al., 2021).

Spatial awareness techniques, integrated into AR-based robotics systems, provide different levels of spatial perception. For example, the AR-based robot programming interface proposed by Fang et al. (2014) displays the 3D AR scene on a 2D monitor, where the precise depth information is lost. As a result, systems like this are unsuitable for tasks requiring precision. Other automatic spatial perception methods based on object detection are limited to specific applications, such as empty plane (Gradmann et al., 2018), marker placement (Lin et al., 2017), and additional precise cameras (Bambuŝek et al., 2019). Additionally, automatic algorithms for object detection and registration are time-consuming and hard to perform in real-time (Makhataeva & Varol, 2020). Recently, Chen et al. (2020) presented a virtual-physical collision detection method applied in robot programming, but its performance depends on the appropriate placement of sensors in opposite of the user, a proper camera angle, and an unobstructed environment.

## HA$R^2$bot system design

To design HA$R^2$bot, we followed the NASA system design process (Aeronautics & Administration, 2019) with four steps, namely, developing stakeholder expectations, technical requirements, logical decompositions, and design solutions. We first figured out the key expectations from stakeholders and then assessed the existing programming methods using these expectations to obtain design guidelines. Then, we conducted another assessment from the perspective of cognitive load theory on the existing programming methods for more design guidelines. Finally, we came up with a design solution for HA$R^2$bot, following the previously obtained design guidelines.

### Assessment of exiting methods based on stakeholders' expectations

The definition of stakeholder expectations is the initial process that establishes the foundation for developing the HA$R^2$bot system. We reviewed the literature about the expectations of the robot programming method (Fang et al., 2014; Pan et al., 2012), especially from the industry perspective (Brogårdh, 2007; Aaltonen & Salmi, 2019). We summarized these expectations as: *Complex Task*, *Intuitiveness*, *Reusability*, *Generalization*, *Efficiency*, and *Safety*. We assessed the primary existing robot programming methods with these six expectations, and listed the results in Table. 1.

#### Complex task

In industrial applications, the environment for the robot programming could be complex with multiple obstacles (see Fig. 1a, b), and there could be different production tasks with a varied environment for the same robot. Therefore, a collision-free toolpath in a complex environment is expected. For the offline programming methods, the complexity of the tasks usually fails the automatic path planning algorithms (Rajendran et al., 2019a, b). The manual leading method allows a human operator to manually move the robot arm, set up configurations, and record the path. Therefore, it has the ability to handle complex tasks. The pendant teaching, enables human operators to pick up waypoints and set up configurations of the robot using the teaching pendant. The coordinate system in the pendant teaching is usually defined in the robot system, and therefore the operator is always required to trace the coordinate frame when the robot is moving (Pan et al., 2012). This way of interaction makes the pendant teaching difficult to handle complex programming tasks.

#### Intuitiveness

Industries have to recruit more robot operators due to the increasing adoption of robots. To support the novice robot operator, the robot programming method is expected to be intuitive (Brogårdh, 2007). The offline programming method has a steep learning curve and requires a substantial amount of effort to pick up, which causes difficulty for the novice robot operator. Using a teach pendant to jog a robot is not intuitive since the coordinate system is defined in the robot system (Pan et al., 2012), which is invisible to the user. The operator has to track the coordinate system when operating the robot. The manual leading method relies on the operator's manual to set up the configuration angle of each robot joint; therefore, it is relatively intuitive to use. Nevertheless, the efficiency and quality of robot motion highly depend on the understanding of robot arm kinetics and experience in con-

**Table 1** An assessment of exiting robot programming methods and proposed HA$R^2$bot, from the perspective of stakeholders' key expectations (Brogårdh, 2007; Fang et al., 2014), including Complex Task, Intuitiveness, Reusability, Generalization, Efficiency, and Safety

| | Online Programming | | Offline Programming | | HAR²bot |
|---|---|---|---|---|---|
| | Pendant teaching | Manual leading | Text-based programming languages | Graphical programming languages | |
| Complex Task | ✗ | ✓ | ✗ | ✗ | ✓ |
| Intuitiveness | ✗ | ✓ | ✗ | ✗ | ✓ |
| Reusability | ✗ | ✗ | ✓ | ✓ | ✓ |
| Generalization | ✓ | ✓ | ✗ | ✗ | ✓ |
| Efficiency | ✗ | ✗ | ✗ | ✗ | ✓ |
| Safety | ✗ | ✗ | ✓ | ✓ | ✓ |

figuring robot arms, which is also challenging for a novice operator to master (Aaltonen & Salmi, 2019).

### Reusability

Stakeholders also expect to change the whole or partial existing paths in different production scenarios (e.g., a new setup of robots or the same setup with partial paths to be adjusted). The offline methods provide modularized code blocks (either in text or graphics) associated with individual steps or configurations, which can be edited or reused separately. The online methods have no such flexibility to reuse the existing paths and always require the operator to restart the programming from the very beginning if any change or editing is required.

### Generalization

The programming method is expected to be general enough to handle most programming tasks (Brogårdh, 2007). The offline method can successfully handle programming tasks with relatively simple obstacles but usually fails to handle tasks with complex obstacles. The online method, relying on the operator's experience, could handle complex planning tasks if the operator is experienced. However, even the online method sometimes suffers from tedious operations and trial-and-error to avoid collisions.

### Efficiency

To increase the production rate, the programming method is expected to be efficient even in a complex environment. For complex path planning tasks (e.g., the ones shown in Fig. 1a, b), none of the existing programming methods are able to handle them efficiently. The offline method suffers from the increasing difficulty of the search-based planning algorithm due to the complexity of obstacles, while the online method requires tedious operations and an extensive amount of trial-and-error to complete complex path planning tasks.

### Safety

There are fewer concerns with the offline method since path planning and robot operation are asynchronous, and the robot and the operator usually do not co-exist in the same space. Using the online methods including both pendant teaching and manual leading to operating Cobots, on the other hand, receive concerns about the safety from stakeholders (Brogårdh, 2007; Aaltonen & Salmi, 2019). There are concerns partially due to the lack of technologies to ensure the safety of the Cobot operation (Aaltonen & Salmi, 2019), such as auditory alarms or visual notifications when the robot is in motion. Another reason for the concerns is that the operator lacks confidence in the Cobot's motion without seeing it.

***Remark 1*** The assessment of the existing methods pointed out the features the proposed AR-based method should have. Firstly, the proposed method should keep humans in the loop of the robot programming to deal with complex tasks and different types of tasks. The proposed method is desirable to support the operator in validating the planned path for a collision-free solution. Secondly, the proposed method should provide the operator with simple, intuitive, and efficient interactions and UIs to complete the programming tasks. Thirdly, the proposed method is expected to be able to keep the planned path and reuse it with necessary editing and modification. Lastly, adequate safety technologies are desired to be proposed and implemented with the programming method.

## Assessment of exiting methods based on Cognitive Load Theory

### Different types of CLTs

The CLT, proposed by Sweller (1988) in 1988, is related to the amount of information that the working memory can hold at one time. CLT is built upon Baddeley and Hitch's work about human memory (Baddeley & Hitch, 1974), and it also views human cognitive architecture as working memory and long-term memory. The working memory has a limited capacity and is responsible for direct attention in the learning process, while the long-term memory has an endless capacity for storage and works with the working memory to retrieve information (Baddeley, 2003). Therefore, how to effectively present information to reduce the workload of working memory and promote the storage of information from working

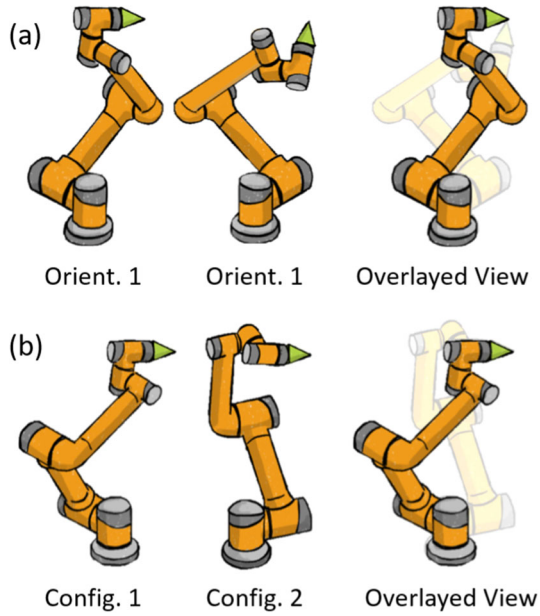


**Fig. 2** The end-effector could be located in the same position, but the orientations of the end-effector are totally different (**a**). The end-effector could have the same position and orientation, but the joint configuration could be totally different

memory into long-term memory is the focus of CLT (Sweller, 2011).

The CLT was further developed, and three different types of cognitive loads were introduced, including intrinsic, extraneous, and germane cognitive loads (Sweller et al., 1998). Intrinsic cognitive load is associated with the specific learning contents, which is the inherent difficulty of the learning objects itself (Sweller et al., 1998). Extraneous cognitive load describes the load generated by the ways and methods of presenting the information or tasks. Germane cognitive load refers to the work put into creating a permanent store of knowledge, or a schema. It is the elements that aid information processing and contribute to the development of "schemas" (Sweller et al., 1998; Van Merriënboer & Sweller, 2010). Based on the definition of different cognitive loads, the strategies associated with them should be different as well. Given the learning tasks are fixed, the intrinsic cognitive load can not be eliminated completely, but can be *optimized* and *simplified* (Sweller, 1999). The extraneous cognitive load, related to the ways and methods of presenting the learning content (e.g., instructional system design, UI and interaction design, etc.), should be *reduced*. The germane cognitive load, on the other hand, is desirable to be *increased*, through pre-training, feedback, etc.

### Assessment of exiting methods

To understand the different cognitive loads in existing robot programming methods, We conducted a pilot study with two robot operators, including one novice ($\sim$ 2 months experience) and an expert ($>$ 5 years experience). The pilot study is mainly a semi-structured interview that is centered on the question: what is the difficulty to program robots using existing programming methods? Each of the interviews lasted about 2 h and was scripted and coded by 3 individual researchers. The findings were summarized and categorized into three groups under each type of cognitive load.

**Intrinsic cognitive load** The intrinsic cognitive load for robot programming is relatively high for novices. For example, it is difficult for novices to understand that an end-effector can achieve the same position with a different orientation (see Fig. 2a), and an end-effector can achieve the same position with the same orientation but totally different configurations of the robot's joints (see Fig. 2b). The reason is that the position of the end-effector is in Cartesian space, but the orientation and configuration are in angle space, which is not intuitive to novices. The online method relies on the operator's input of the orientation and configuration information. The orientation and configuration are also essential to the search space of the path planning methods and the detection of collisions. Another challenge for the online method is the validation of the planned path. Using the online method, the operator struggles to validate the planned path since they cannot recall the information about the robot at each waypoint without a visualization. The offline method indeed has a visualization of the orientation and configuration information. However, it requires a time-consuming and complex 3D reconstruction of the physical environment to handle the obstacles.

**Extraneous cognitive load** Both online and offline methods have a high extraneous cognitive load. The manual leading method requires the operator to manually configure each of the joints and move the position of the end-effector. This way of programming heavily relies on the operator's understanding of the configuration of the robot and the limitations of each joint. A novice operator will have difficulty programming a robot using the manual leading method. The pendant teaching method uses a touch-screen pad or a joystick to allow the operator to control the end-effector's position, orientation, and joint configuration. The UI of the pendant is usually simple, but the operation of the UI is indirect to the physical robot's motion. The novice operator needs substantial effort to establish the correspondence between the operations with the UI and the motion of the physical robot. The offline method requires skills in text-based or graphics-based robot programming languages, which has a steep learning curve and becomes a major barrier for a novice operator. The existing programming methods also lack guidance for the operator to deal with problems during the programming tasks. For example, using the pendant teaching, when the joint angles reach the limit, the UI only displays a warning about failure to further move the robot, but

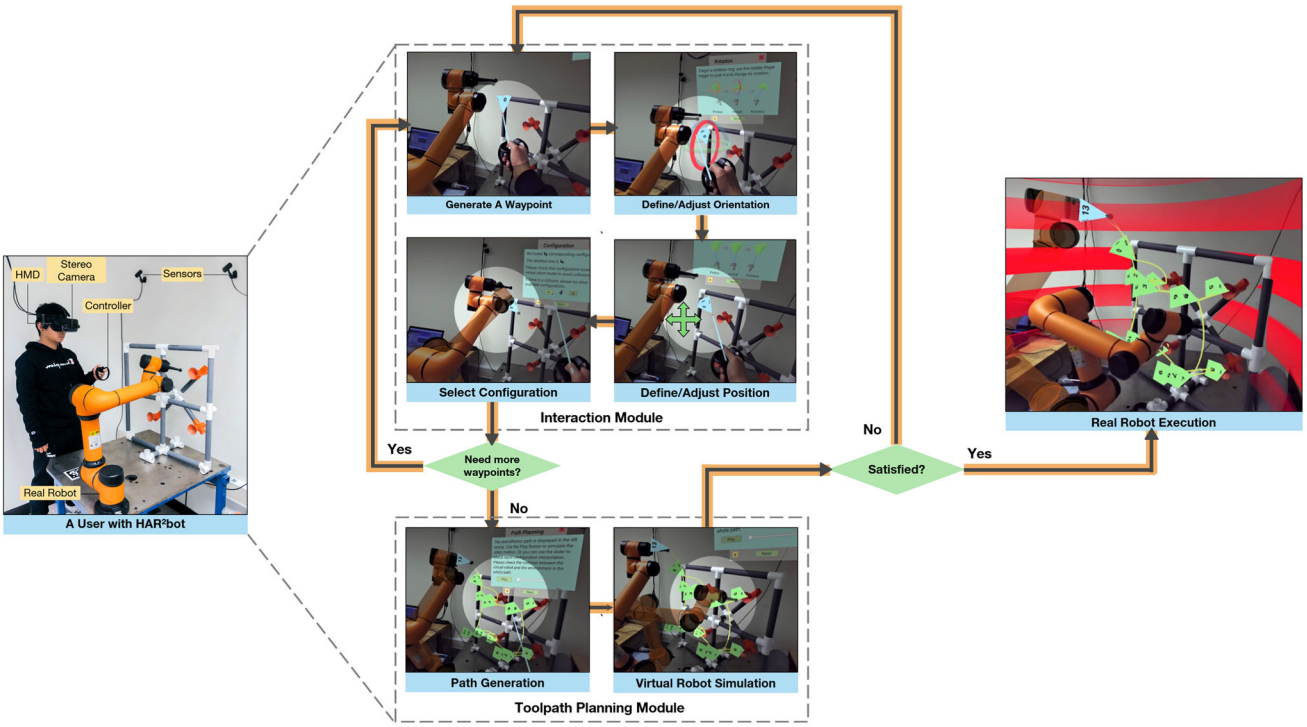


**Fig. 3** The Ha$R^2$bot system's workflow is as follows: an operator programs a Cobot in three steps. In step 1, several operations, including defining a waypoint's spatial pose, adjusting the waypoint, and selecting the configuration of the robot joints, will be repeated until enough waypoints are defined. In step 2, the robot configuration information is fed into the toolpath planning module, and the toolpath solutions are computed. In the last step, the operator will check and confirm the toolpath to ensure it is collision-free and satisfactory

there is no specific information for the operator. The novice operator, therefore, finds it difficult to deal with such cases.

**Germane cognitive load** In contrast to intrinsic and extraneous cognitive load, it is anticipated that germane cognitive load will increase. Through the interviews with two robot operators, we noticed that they all motioned that the more they programmed with the robot, the more fluently they performed new programming tasks. We also noticed that two operators mentioned it would be helpful if a mentor could give them evaluative feedback about their performance. Another interesting point is that the experienced robot operator recalled his experience when he was a novice, and it is quite similar to that of the novice operator. With more knowledge of robotics and more experience in robot programming, experienced robot operators felt more confident and were much less challenged by the programming tasks.

***Remark 2*** The proposed method is expected to optimize the intrinsic cognitive load by making the invisible orientation and configuration information of the robot visible and provide an effective and efficient way for the operator to validate the planned paths collision-free. In order to reduce the extraneous cognitive load, the proposed method should be simple and able to control the robot intuitively and instantly. Some assistive instructions and notifications are also desirable to reduce the extraneous cognitive load. Last but not least, the germane cognitive load could be increased by a pre-training session and a post-feedback session.

## The proposed solution

Based on the guidelines included in Remarks 1 and 2, we came up with a novel AR-based robot programming approach with a set of cognitive load-aware features.

### AR-based human-in-loop approach

To include humans in the loop, a new workflow of HA$R^2$bot is proposed, as shown in Fig. 3. The operator wearing an AR headset is able to define waypoints with a virtual robot and the physical environment simultaneously. Waypoints can be added or removed and can be moved freely. Given the position of each waypoint, the operator can also specify the orientation of the end-effector and possible configurations based on the result obtained from the embedded inverse kinematic package. After the operator determines the positions and orientations of all points, as well as the configurations of robot joints at each waypoint, the path planning solver will compute a feasible solution. The solution is displayed to

the operator, and the physical environment with occlusions is also visualized. The operator, therefore, is able to validate the planned path and make sure there is no collision. Once the operator is satisfied with the planned path and presses on "execution", the physical robot will follow the planned path. This planned path can be stored, loaded, and further edited by the operator. When the physical robot is in motion, the working zone of the robot will be notified in red (see the rightmost of Fig. 3), and an auditory alarm will be played.

#### Cognitive load-aware features

HA$R^2$bot displays the essential information and optimize the intrinsic cognitive load. The information includes the position and orientation of the end-effector and the possible configurations of the robot at a given waypoint. HA$R^2$bot also displays the virtual robot, the planned path, and the physical environment with occlusion in real time. This will optimize the operator's intrinsic cognitive load on path validation. HA$R^2$bot's interactions are rather simple, and most of the operations can be completed through the physical hand-held controller and virtual UIs. A simple drag-and-drop interaction can realize a direct operation of the robot, and any change can be visualized instantly. These interfaces and interactions will reduce the extraneous cognitive load of the operator. HA$R^2$bot has the instruction and notification functions to notify the operator of an error when something goes wrong, prompt the next step when the one step is completed, or warn the operator of the workspace of the physical robot when the planned path will be executed. These instructions or notifications are in visual or auditory forms, which help reduce the extraneous cognitive load. The UI design of HA$R^2$bot follows the principle of flat hierarchies to ease the effort of the operator to find a function, which further reduces the extraneous cognitive load. To increase germane cognitive load, we set up a pre-training session to allow the operator to get familiar with HA$R^2$bot system and the task. A detailed user study is conducted to verify the effectiveness of these features, and it is presented in Sect. 6.

## System implementation

### Hardware and software setup

The HA$R^2$bot system is developed upon several hardware systems, including an AR headset and a collaborative robot arm. The AR headset consists of an Oculus Rift VR headset and a ZED mini stereo camera. The Oculus Rift headset and the associated hand-held controller, serving as the base of HA$R^2$bot, provide an immersive display, tracking, and interactions. The Zed mini camera captures the 3D information of the environment, and its SDKs enable the 3D data transmission as well as integration with the Oculus Rift headset. We chose the AUBO i5, an affordable Cobot, as the testing platform. The AUBO i5 has 6 degrees of freedom (DOFs), a payload of 5 kg, and a maximum reachable distance of 924 mm. The illustration of the full hardware setup can be found in Fig. 4.

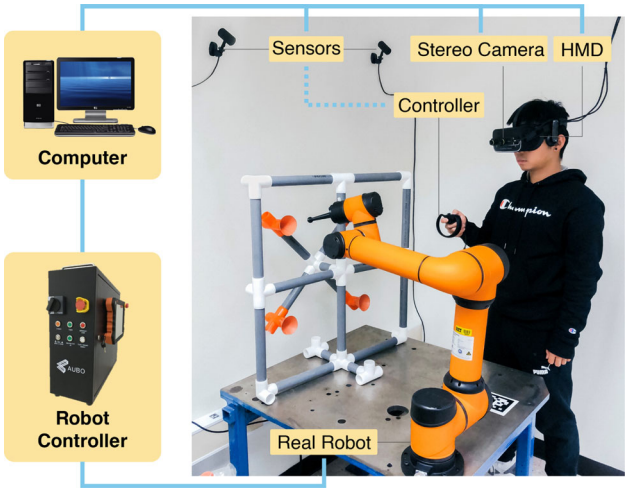


**Fig. 4** The overall hardware systems

The primary software environment is an integration of the Robot Operating System (ROS) with the Unity 3D engine. The ROS system has good compatibility and provides powerful and rich algorithm packages. The SDKs of AUBO i5 (Aubo Robot, 2023) connect the robot arm with the ROS system seamlessly. We also established a message communication protocol between ROS and Unity 3D (Manring et al., 2020), and therefore, the hardware and the software systems communicate through such a protocol.

### Robot programming method

HA$R^2$bot keeps the human operator in the loop of robot programming to deal with tasks with complex obstacles. The operator is able to define the positions of waypoints, which indicate the positions that the end-effector will pass through. Besides the positions of the end-effector, the orientation of the end-effector and the configuration of robot joints also affect the path planning results and collisions with the environment. Therefore, for two neighboring waypoints, HA$R^2$bot allows the operator to set up the orientations of the end-effector and the configuration at each waypoint, and then, all the information will be fed into the path planning library, the Open Motion Planning Library (OMPL) (Sucan et al., 2012), to compute motion plan in between the two neighboring waypoints. The flowchart of HA$R^2$bot's robot programming method is shown in Fig. 5.

The primary difference between HA$R^2$bot's programming method and existing offline methods is that HA$R^2$bot

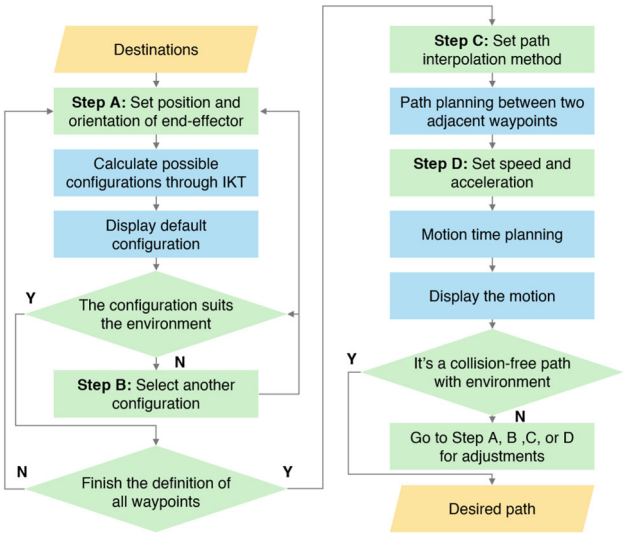


**Fig. 5** The flowchart of the robot programming method is integrated with both online and offline programming in HA$R^2$bot system. Green rectangular boxes are steps processed by human operators. Blue rectangular boxes mean processing steps managed by the computer. Green diamonds are decisions made by human operators

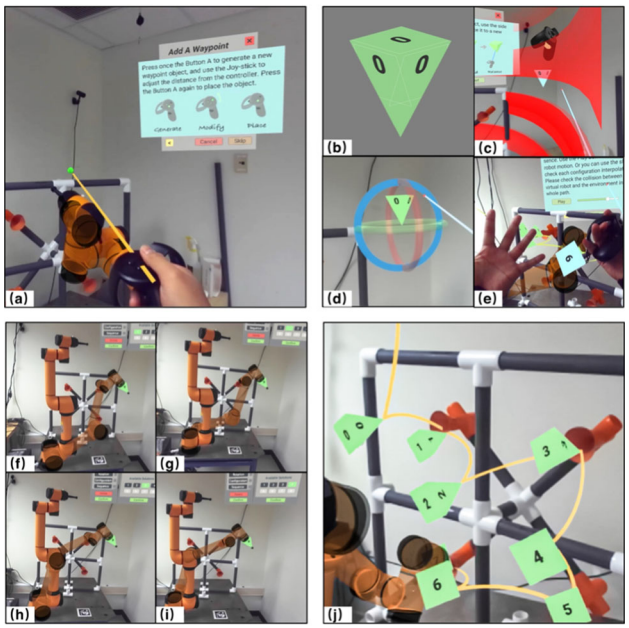


**Fig. 6** **a** The menu interface with instructions remains at a fixed position in the AR scene, and users apply a virtual raycast starting from the controller for the interaction; **b** an upside-down pyramid to represent the waypoint virtual object, where the top of the pyramid is the position of the waypoint; **c** the virtual boundary of the robot workspace is displayed for safety purposes; **d** a rotation gizmo as a stereo interactive tool to rotate the virtual object; **e** a virtual object can be occluded by a real object, i.e., the hand is in front of view; **f**–**i** the virtual robot with possible robot configurations is displayed at the same waypoint; **j** the final path defined through several waypoints is displayed

does not require the CAD model or the 3D reconstruction of the environment. The benefit is obvious: the operator's perception is able to figure out a feasible configuration that will lead to a collision-free path, which is more efficient. The existing offline method usually has a high failure rate when planning on a task with complex obstacles, which is because, without the human operator's perception information, the random search algorithms can easily get stuck trying to find a feasible solution. Indeed, online programming methods don't require a CAD model or 3D reconstruction of the environment as well. However, they require operators possessing extensive experience in manual robot operation, and are not able to leverage the automatic path planning algorithms as HA$R^2$bot does. To support the operator specifying configurations, we integrated a powerful inverse kinematics solver, IKFast (Smits et al., 2011), into HA$R^2$bot system. IKFast uses analytic methods that can find all possible solutions as well as ensure the consistency of solutions. For the 6-DOF robot arms, there will be at most 8 solutions for a given end-effector position and orientation. HA$R^2$bot visualizes all possible solutions for the operator to choose from. We adopted a configuration distance metric that measures the distance between a configuration at one waypoint and the configuration at the neighboring waypoint. This distance is defined as the sum of the angular variations of corresponding joints between two configurations. We rank all possible configurations in ascending order in terms of configuration distance, and visualize them also in such an order, to help the operator make decisions. Note that this function is just a suggestion, and the final decision will still be made by the operator.

## The interaction and UI design

The interaction of HA$R^2$bot is realized by a virtual ray casting from a hand-held controller (see Fig. 6a). This virtual ray, serving as a pointer, enables multiple interactions, including the interaction with virtual menus and buttons (Fig. 6a), the interaction with the interface for adjusting waypoints and the end-effector's orientation (Fig. 6b, d), and the interaction with the interface for choosing the configuration of the robot (see Fig. 6f–i).

An upside-down pyramid is displayed to represent the orientation of the end-effector at each waypoint (shown in Fig. 6b). A rotation gizmo helps the operator adjust the orientation of the end-effector, as illustrated in Fig. 6d. Both the position and orientation of the end-effector are fed into the inverse kinematics solver to calculate the configurations. The IKFast Kinematics Solver (OpenRAVE, 2018) from MoveIt Package is integrated into HA$R^2$bot as the inverse kinematics module. The IKFast can analytically solve the kinematics equations of complex kinematics chains, which provides sta-

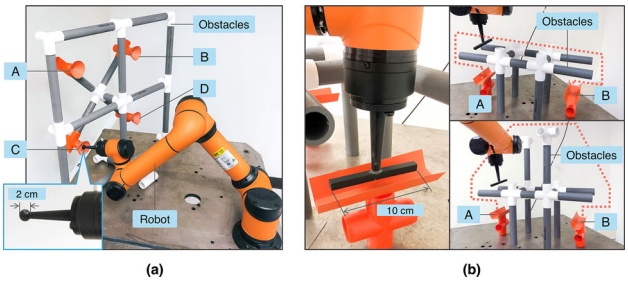


**Fig. 7** Two scenarios were implemented in system verification. **a** The robot is desired to travel through four destinations (A–D) in a space truss structured environment. **b** The robot needs to move a T-shaped end-effector from destination A to destination B and avoid collision within the space truss structure

ble solutions, including all possible configurations of a 6 DOFs robot manipulator as fast as 5 microseconds on recent computers. The related robot configurations are visualized for the operator to choose from, as shown in Fig. 6f–i. After the path is planned, it is visualized for the operator to check its feasibility, as shown in Fig. 6j.

HA$R^2$bot provides the instructional information to prompt the operator to get to the next step when the current step is completed. The operator will be notified with the specific information displayed if the path planning cannot be completed, such that the operator is not able to make adjustments accordingly. During the planned path being executed by the physical robot, there will be a visualization of the workspace of the robot as well as an auditory alarm played.

## System verification

We verified HA$R^2$bot through the experiments with two programming tasks (Task 1 & 2), as shown in Fig. 7. The setup of these two programming tasks is inspired by Rajendran et al. (2019a) (see Fig. 1a, b). The first scenario (Task 1) simulating the application in the automobile industry (Fig. 7a), has four destinations (A, B, C, and D) located in window-like spaces. The end-effector (a sphere with a 2-centimeter diameter), mimicking the welding tool in automobile manufacturing, is expected to move from one destination to another in a certain sequence. The second scenario (Task 2) simulates a robot moving a T-shaped tool (a 10-centimeter long stick) from destination A to destination B through a narrow slot (Fig. 7b).

For each task, there are three steps: the waypoint definition, the path verification, and the path adjustment. The waypoint definition requires the operator to define several waypoints to complete the task. Afterward, it is required to verify the planned path and make sure there is no collision of the robot arm with the environment. Lastly, the operator is asked to adjust the defined path. For Task 1, the operator is required to modify the originally defined path in the order "A-B-D-C" to "A-C-D-B". Similarly, for Task 2, the operator originally plans the path in the order "A-B" and needs to adjust the path due to the presence of newly added obstacles.

The case study, consisting of two complex tasks, is conducted by an expert user with advanced-level knowledge of both AR and robot programming. The same set of tasks are completed by the user using the manual leading, the pendant teaching, and HA$R^2$bot. The offline method, requiring the expensive 3D reconstruction of the environment and having a high failure rate with a complex environment, has an obvious disadvantage compared with other methods. Therefore, we did not include the offline method in the verification and following user study. Both the manual leading method and the pendant teaching method are supported by AUBO's tools (Aubo Robot, 2023). To eliminate the randomness in the experiments, for each task, we did 10 trials (including an attempted trial of the task and 3 time-recorded trials for each condition) and calculated the average time for each step (see Table 2).

From Table 2, we found that the efficiency of HA$R^2$bot is significantly higher than the other two methods in each step of the two tasks. For Task 1, using HA$R^2$bot saves 58.2% more time than manual leading and 48.1% more time than pendant teaching. Similarly, for Task 2, there is a 54.6% time reduction from the one using manual leading and a 61.6% reduction from the one using pendant teaching. We also analyzed the time for each of the steps for both tasks. In the waypoint definition step, HA$R^2$bot achieved the most time reduction over the pendant teaching (46.8% in Task 1 and 64.4% in Task 2). Both the two existing methods spent much more time than HA$R^2$bot spent in the verification step, where HA$R^2$bot achieved the most time reduction over manual leading in Task 1 (77.3%), and the most time reduction over pendant teaching in Task 2 (50.5%). In the adjustment step, HA$R^2$bot achieved the most time reduction over manual leading in Task 1 (78.4%), and the most time reduction over pendant teaching in Task 2 (60.8%). In conclusion, HA$R^2$bot effectively supported the operator to complete the two complex programming tasks, and it achieved much higher efficiency than the two existing online methods, especially for the verification and adjustment steps.

## User study

### Experiment setup

To further evaluate HA$R^2$bot system, we conducted a user study with 16 participants. This study and the previously described pilot study have been approved by the Institutional Review Boards of RIT. Since HA$R^2$bot is expected to support novice users, we intentionally invited participants without

**Table 2** Completion time among three programming methods for two scenarios

| Programming method | Task 1 time (s) | | | | Task 2 time (s) | | | |
|---|---|---|---|---|---|---|---|---|
| | Definition | Verification | Adjustment | Overall | Definition | Verification | Adjustment | Overall |
| Manual leading | 306 | 365 | 417 | 1088 | 377 | 79 | 289 | 745 |
| Pendant teaching | 530 | 127 | 219 | 876 | 472 | 97 | 311 | 880 |
| HA$R^2$bot | 282 | 83 | 90 | 455 | 168 | 48 | 122 | 338 |

prior robot programming experience. We asked each participant to complete a pre-study survey with questions about their demographic information before the study. Among the 16 participants, there are 9 males and 7 females, and they are college science and engineering students. Six participants have prior VR experience, and 4 participants have AR experience. These participants were randomly divided into 2 groups (8 in each group): Group A and Group B. Each participant was asked to complete programming Task 1 (shown in Fig. 7) twice, using HA$R^2$bot and existing online programming methods (ONP), respectively. The participants can choose either one of the online methods, including manual leading and pendant teaching, or even a combination of the two. To eliminate any possible bias due to the order of the experiment, participants in group A were asked to complete the programming task using HA$R^2$bot first and then ONP, whereas participants in group B completed tasks using ONP first and then HA$R^2$bot. Prior to the experiment, all participants were provided with a pre-training session with 10 min of introduction and operation instruction for each method to familiarize them with these methods and the task goals. The participant has 3 trials to complete the task using each method. Using a certain method, if the participant cannot finish the task in 3 tries, the task is considered a failure for this method.

## Metrics

During the study, the completion time, the first-trial success rate of the task for each method, and the number of waypoints each participant defined were recorded. The time and the number of waypoints indicate the efficiency of a specific method, and the first trial success rate reflects its effectiveness. After the study, we evaluated HA$R^2$bot, using the System Usability Scale (SUS) (Brooke, 1996) and the UX survey (Fowler & Cosenza, 2009). SUS measures usability, and the UX survey assesses the participants' satisfaction with the UIs and interaction. To compare the cognitive load for HA$R^2$bot and ONP, we asked participants to complete the NASA Task Load Index (TLX) (Hart & Staveland, 1988) and the Cognitive Load Component (CLC) questionnaire (Naismith et al., 2015). The NASA TLX measures the overall cognitive load, while the modified CLC measures the cognitive load of each type. Since there are no standard questions

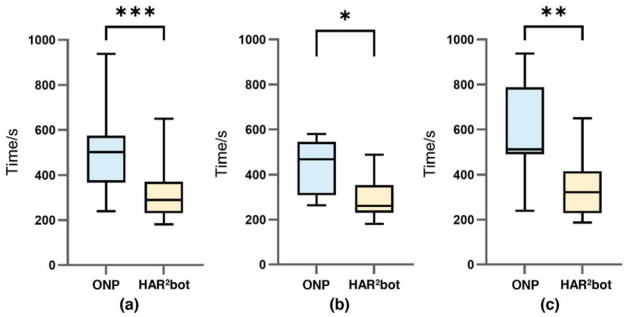


**Fig. 8** Box and whisker plot of completion time for **a** the whole population, **b** group A, and **c** group B

for the UX survey and CLC questionnaire, we developed our own questions, which are listed in Appendices A (for the UX survey) and B (for the CLC questionnaire). We also conducted a post-study interview with each participant for a qualitative evaluation. The interview questions can be found in Appendix C.

## Statistics

### Completion time

We first compared the time for successful completion of the task using HA$R^2$bot and ONP. We hypothesized that the HA$R^2$bot system speeds up the programming process for novice users. The ONP ($KSdistance = 0.1441$, $p > 0.1$) and HA$R^2$bot set ($KSdistance = 0.1542$, $p > 0.1$) both passed the Kolmogorov-Smirnov normality test, indicating that they follow a normal distribution. Therefore, we conducted a t-test for differences between datasets. The descriptive statistics are presented in Fig. 8a. All 16 participants spent significantly less time ($t(15) = 4.637$, $p = 0.0003$) using the HA$R^2$bot (M = 508, SD = 181) than the one using the ONP ($M = 318$, $SD = 119$). The overall completion time is reduced by 37% on average using HA$R^2$bot.

For each group (A and B), we ran parametric paired t-tests, rejecting the null hypothesis ($p < 0.05$) respectively. For both groups A ($p = 0.0311$) and B ($p = 0.0062$), the participants spent significantly less time using HA$R^2$bot. To test the effect of the order, we used a t-test to compare the completion time of the HA$R^2$bot between the two groups and found no significant difference ($p = 0.03826$). Similarly, we

**Table 3** Success rate of the first trial of the task for groups and methods

| | ONP (%) | HA$R^2$bot (%) | Pop. size |
|---|---|---|---|
| Group A | 87.5 | 75 | 8 |
| Group B | 75 | 87.5 | 8 |
| Overall | 81.25 | 81.25 | 16 |

discovered no significant difference in the time spent using ONP between the two groups ($p = 0.1226$).

### First-trial success rate

We counted how many trials participants needed to complete the task. If the physical robot collides with obstacles either during the planning phase or execution phase, the task is considered a failure. If the participants fail in the first trial, they need to conduct the next trial until they complete the task or reach the maximum number of trials (3 trials). The first-trial success rate is defined as the ratio of the number of people who completed the first trial to the total population size, which is summarized in Table 3. In group A, participants achieved a higher first trial success rate using ONP, while the case for group B is the opposite: participants using HA$R^2$bot achieved a higher first trial success rate. This phenomenon can be explained by the fact that participants have a better understanding of the task and programming logic when they are using the second programming method. Although there are slight differences between the two groups, the overall first success rates for the two methods are the same, which indicates that HA$R^2$bot is as effective as ONP.

### Number of waypoints

We hypothesized that using HA$R^2$bot, participants would define fewer waypoints. The number of waypoints each participant defined using each method is presented in Fig. 9a. The two sets of data for two methods passed the Kolmogorov-Smirnov test, and the t-test was applied to find the differences between the two sets of data (shown in Fig. 9b). The result ($p(15) = 4.392$, $p = 0.0005$) indicates that participants defined fewer waypoints using HA$R^2$bot ($M = 5.188$, $SD = 0.9106$) than they defined using the ONP method ($M = 6.875$, $SD = 1.746$).

## System usability and user experience

### SUS

We applied standard SUS to verify the usability of the HA$R^2$bot. The scores of SUS of participants are presented in

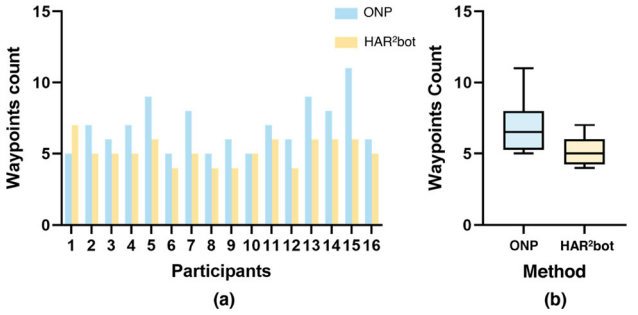


**Fig. 9** **a** Number of waypoints participants defined using HA$R^2$bot and ONP method; and **b** box and whisker plots

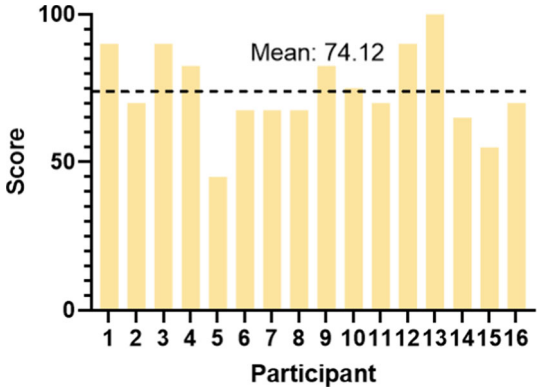


**Fig. 10** The SUS scores of HA$R^2$bot with a mean of 74.12

Fig. 10. We observed that the mean SUS score ($M = 74.21$) is above average 68 (Brooke, 2013).

### User experience

The UX survey (see Appendix A and C) consists of 8 Likert scale questions and 5 open-ended questions. The statements and scores of the Likert scale questions are presented in Fig. 11. Generally, most participants showed a positive opinion of the HA$R^2$bot system, considering HA$R^2$bot as efficient and comprehensible for robot programming. 13 out of 16 participants perceived HA$R^2$bot as easy to understand. The majority of participants (13 participants) approved the advantages of the waypoint definition using HA$R^2$bot. 12 out of 16 participants gave positive feedback about the visualization of the simulation, which is intuitive and easy to understand for collision avoidance.

## Cognitive load

### NASA-TLX

The NASA TLX measures the overall cognitive load of the participants. Both the datasets for ONP ($KSdistance = 0.1710$, $p > 0.1$) and HA$R^2$bot ($KSdistance = 0.1480$,

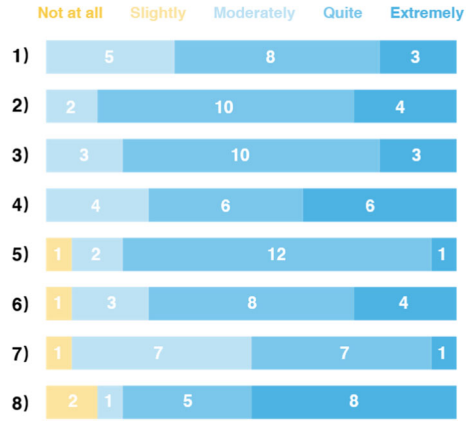


**Fig. 11** The stacked bar chart of the user experience survey responses of participants

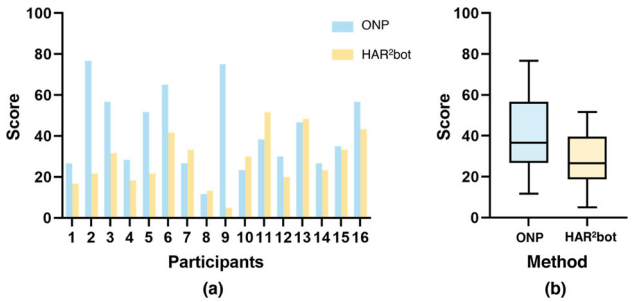


**Fig. 12** **a** Summary TLX score of participants and **b** box and whisker plot of total TLX score between conditions

$p > 0.1$) passed the Kolmogorov-Smirnov test, and a t-test was applied. According to Fig. 12, the 16 participants had a significantly lower TLX score when using the HA$R^2$bot ($M = 28.33$, $SD = 13.17$) compared with the one when using ONP (M = 42.19, SD =19.55), where $t(15) = 2.448$, and $p = 0.0271$. This indicates that HA$R^2$bot effectively reduced the cognitive load compared with ONP.

### CLC

We applied a modified version of the CLC questionnaire with our questions, inspired by Naismith et al. (2015), to measure 3 types of cognitive loads (see Fig. 13). From the total score of the result, 16 participants have significantly lower cognitive load using HA$R^2$bot ($M = 11.31$, $SD = 3.114$) compared with the one using ONP ($M = 13.63$, $SD = 0.649$), where $t(15) = 2.151$, and $p = 0.0482$. For each type of cognitive load, the ones associated with HA$R^2$bot are also lower than the ones associated with ONP. Note that we processed the scores of germane cognitive load using the full score (5) to substrate the original germane score in order to have a consistent visualization in Fig. 13 and to calculate the overall cognitive load.

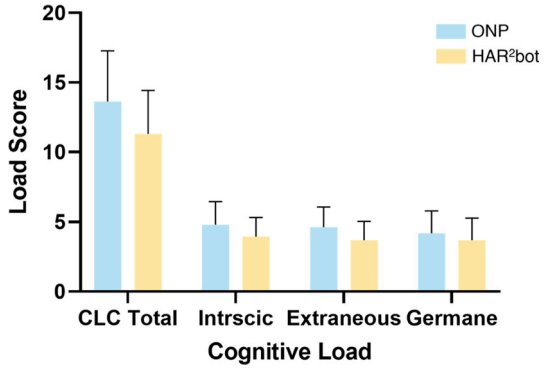


**Fig. 13** Summary of the CLC total score and partial cognitive load of participants using two methods

## Qualitative measurement

We conducted a post-study interview with each participant to further evaluate HA$R^2$bot qualitatively. Particularly, we focus on the management of the three types of cognitive load. The safety of programming tasks is also assessed.

### Intrinsic cognitive load

Our strategy to optimize the intrinsic cognitive load is making the invisible information visible and providing instant visual feedback and simulation with occlusion. The feedback from the participants proves the effectiveness of these strategies. One participant (P8) appreciated the immersive visualization of the robot position, orientation, and joint configurations, and thought this information was "useful information to help the planning". P14 and P16 found the instant simulation of HA$R^2$bot was quite helpful. P14 thought the simulation was "easy to understand", while P16 considered it "convenient, time-saving, and clear". Particularly, P16 believed that she could "gain an understanding of the total movement of the robot when completing the path planning".

### Extraneous cognitive load

The interaction of HA$R^2$bot received positive feedback from the participants. P9 felt the interaction with the virtual robot "sped up the planning task". P14 also thought that HA$R^2$bot is "easier to use and friendly for users". The UIs of HA$R^2$bot were praised by P14, and he "can easily find what to do for the next step". This proved the assistive instructions and notifications of HA$R^2$bot supported users effectively. On the contrary, the ONP is considered by P7 to have "a complex UI", "it does not provide guidance" and "causes a lot of troubles". P8 found that "it was hard to jog the robot manually because the robot is so heavy for her", and she preferred HA$R^2$bot because it allowed her to "move the virtual robot easily".

### Germane cognitive load

To increase the germane cognitive load, we provided a simple solution by having each participant take a 10-minute session of instruction about the system and the programming task. Even this primitive implementation received positive feedback from the participants. P4 and P10 both mentioned that the instruction from the instructor and the communication with the instructor were helpful for them "to understand the programming task", which agrees with the quantitative evaluation of CLC scores shown in Fig. 13.

### Safety

From the feedback of participants, we also noticed that HA$R^2$bot improved the safety of the robot programming. There are 14 out of 16 participants reported they felt quiet or extremely safe using HA$R^2$bot. P8 mentioned that the existing online method made her "feel unsafe", and the HA$R^2$bot allowing her to move the virtual robot made her feel safer. P12 expressed her appreciation for HA$R^2$bot's visual notification of the workspace of the robot and the auditory warning when the physical is executing the planned path.

During the user study, when users were asked to use existing online methods, we observed that most users switched between different methods (manual leading and pendant teaching) frequently. They usually prefer to use manual leading for the robot's large movements in the work cell and apply pendant teaching for small but more precise movements. We believe this observation reflects that a single online programming method makes it hard to achieve all expectations of stakeholders, which is consistent with our assessment in Sect. 3.1.

## Conclusion and discussion

In this research, we proposed and developed HA$R^2$bot, an AR-based robot programming system, to support novice users in performing robot programming in a complex environment. A human-centered design process of human-robot interaction systems is presented, and a set of design guidelines is received through the assessment of existing programming methods from the perspectives of stakeholders and the cognitive load theory. With these guidelines, we came up with the human-in-loop workflow of HA$R^2$bot, and the strategies to optimize the intrinsic cognitive load, reduce the extraneous cognitive load, and increase the germane cognitive load. HA$R^2$bot system was implemented and verified with two complex programming tasks. The verification results indicated the effectiveness and efficiency of HA$R^2$bot. Compared with existing online methods, HA$R^2$bot can effectively support the operator to complete complex tasks with much higher efficiency. A user study with 16 participants was conducted to evaluate HA$R^2$bot quantitatively and qualitatively. The study result suggested that HA$R^2$bot has higher efficiency, lower overall cognitive load, lower cognitive loads for each type, and higher safety.

There are drawbacks mentioned by the participants. The visualization accuracy is not satisfactory and could lead to difficulty for the operator in precisely performing the programming tasks and detecting collisions. In the near future, we plan to investigate the sources of the errors in the visualization and propose a potential solution to eliminate these errors. Some participants complained about the interaction using the hand-held controllers, and they preferred the interaction with their bare hands. Developing intuitive interactions would be a possible future direction to address this problem. Other participants suggested a more detailed and intuitive pre-training session and an assistive system during the path planning process. Our current pre-training implementation is rather simple and can be further improved. We believe a more sophisticated pre-training system based on AI would be a potential solution to support operators and increase their germane cognitive load.

**Acknowledgements** This work has been partially supported by the startup fund and the Bootcamp fund of RIT. Qinqin Xiao is partially supported by the National Science Foundation (NSF) under Grant DGE-1922591.

## Appendix A

### *AR Robotic Path Planner System User Study*

*Share your opinions about using the HAR²Bot system for path planning task...*

1) **How do you rate your interaction with the robot by using the HAR2Bot system?**

☐ not at all satisfactory ☐ slightly satisfactory ☐ mildly satisfactory ☐ quite satisfactory ☐ very satisfactory

2) **How safe do you feel during the robot programming task by using the HAR2Bot system?**

☐ not at all safe ☐ slightly safe ☐ moderately safe ☐ quite safe ☐ extremely safe

3) **How well did the robot execute your desired path?**

☐ never follow ☐ general direction ☐ almost ☐ very with small error ☐ go all the way

4) **How intuitive was the robot simulation?**

☐ not at all intuitive ☐ slightly intuitive ☐ moderately intuitive ☐ quite intuitive ☐ extremely intuitive

5) **How easy was it to put critical points in the end-effector space?**

☐ not at all easy ☐ slightly easy ☐ moderately easy ☐ quite easy ☐ extremely easy

6) **How helpful was the motion simulation and visualization for collision-check?**

☐ not at all helpful ☐ Slightly helpful ☐ Moderately helpful ☐ Quite helpful ☐ extremely helpful

7) **How easy was it to estimate the relative position with the depth rendering?**

☐ not at all easy ☐ slightly easy ☐ moderately easy ☐ quite easy ☐ extremely easy

8) **How easy was the interface to understand?**

☐ very difficult ☐ slightly difficult ☐ neutral ☐ quite easy ☐ very easy

## Appendix B

### *AR Robotic Path Planner System User Study*

*Task:*_____ *Condition:*_____

***Tell us about your detailed cognitive workload after using this system...***

**Tips :** For each question, please circle the option that is most applicable to you.

- **How difficult did you find the programming method?**

☐ not at all difficult ☐ slightly difficult ☐ moderately difficult ☐ quite difficult ☐ extremely difficult

- **How complex was the content covered in the robot simulation session?**

☐ not at all complex ☐ slightly complex ☐ moderately complex ☐ quite complex ☐ extremely complex

- **How clear was the instructions for the interaction method?**

☐ not at all clear ☐ slightly clear ☐ moderately clear ☐ quite clear ☐ extremely clear

- **How relevant was the simulation session for real execution?**

☐ not at all relevant ☐ slightly relevant ☐ moderately relevant ☐ quite relevant ☐ extremely relevant

- **How focused were you during the programming session?**

☐ not at all focused ☐ slightly focused ☐ moderately focused ☐ quite focused ☐ extremely focused

- **How well do you remember each step after the programming task?**

☐ remember nothing ☐ a little bit ☐ half process ☐ almost steps ☐ remember all

## Appendix C

### *AR Robotic Path Planner System User Study*

*Talk more about your using experience...*

- **When you use this AR device, what other information or instructions should be displayed for a better experience?**

- **Were there any functions that were difficult to remember, like buttons or processes of implementation?**

- **Were there any parts of the system that felt inefficient?**

- **Do you have any evaluations about the motion simulation part?**

- **What does a better training process of the AR system look like?**

## References


Aaltonen, I., & Salmi, T. (2019). Experiences and expectations of collaborative robots in industry and academia: Barriers and development needs. *Procedia Manufacturing, 38*, 1151–1158. https://doi.org/10.1016/j.promfg.2020.01.204

Aeronautics, N., & Administration, S. (2019). System design processes. Retrieved February 06, 2023, from https://www.nasa.gov/seh/4-design-process.

Association for Advancing Automation. What are collaborative robots? Retrieved February 06, 2023, from https://www.automate.org/a3-content/what-are-collaborative-robots.

Aubo Robot. AUBO-i5 collaborative robot. Retrieved February 06, 2023, from https://www.aubo-cobot.com/public/i5product3?CPID=i5.

Baddeley, A. D., & Hitch, G. (1974). Working memory. In *Psychology of learning and motivation* (Vol. 8, pp. 47–89). Elsevier

Baddeley, A. (2003). Working memory: Looking back and looking forward. *Nature Reviews Neuroscience, 4*(10), 829–839. https://doi.org/10.1038/nrn1201

Bambusek, D., Materna, Z., Kapinus, M., Beran, V., & Smrž, P. (2019). Combining interactive spatial augmented reality with head-mounted display for end-user collaborative robot programming. In *2019 28th IEEE international conference on robot and human interactive communication (RO-MAN)*, pp. 1–8. IEEE. https://doi.org/10.1109/RO-MAN46459.2019.8956315

Brogårdh, T. (2007). Present and future robot control development-an industrial perspective. *Annual Reviews in Control, 31*(1), 69–79. https://doi.org/10.1016/j.arcontrol.2007.01.002

Brooke, J. (1996). Sus: A "quick and dirty" usability scale. *Usability Evaluation in Industry*. https://doi.org/10.1201/9781498710411-35

Brooke, J. (2013). Sus: A retrospective. *Journal of Usability Studies, 8*(2), 29–40. https://doi.org/10.5555/2817912.2817913

Brunete, A., Mateo, C., Gambao, E., Hernando, M., Koskinen, J., Ahola, J. M., Seppälä, T., & Heikkila, T. (2016). User-friendly task level programming based on an online walk-through teaching approach. *Industrial Robot: An International Journal*. https://doi.org/10.1108/IR-05-2015-0103

Cao, Y., Xu, Z., Li, F., Zhong, W., Huo, K., & Ramani, K. (2019). V. ra: An in-situ visual authoring system for robot-iot task planning with augmented reality. In *Proceedings of the 2019 on designing interactive systems conference*, pp. 1059–1070. https://doi.org/10.1145/3322276.3322278

Chen, B., Hua, C., Dai, B., He, Y., & Han, J. (2019). Online control programming algorithm for human-robot interaction system with a novel real-time human gesture recognition method. *International Journal of Advanced Robotic Systems, 16*(4), 1729881419861764. https://doi.org/10.1177/1729881419861764

Chen, C., Pan, Y., Li, D., Zhang, S., Zhao, Z., & Hong, J. (2020). A virtual-physical collision detection interface for ar-based interactive teaching of robot. *Robotics and Computer-Integrated Manufacturing, 64*, 101948. https://doi.org/10.1016/j.rcim.2020.101948

Fang, H., Ong, S., & Nee, A. (2014). A novel augmented reality-based interface for robot path planning. *International Journal on Interactive Design and Manufacturing (IJIDeM), 8*(1), 33–42. https://doi.org/10.1007/s12008-013-0191-2

Fortune Business Insights. Collaborative robots market. Retrieved February 06, 2023, fromhttps://www.fortunebusinessinsights.com/industry-reports/collaborative-robots-market-101692.

Fowler, F. J., Jr., & Cosenza, C. (2009). Design and evaluation of survey questions. *The SAGE Handbook of Applied Social Research Methods, 2*, 375–412.

Fuste, A., Reynolds, B., Hobin, J., & Heun, V. (2020). Kinetic ar: A framework for robotic motion systems in spatial computing. In *Extended abstracts of the 2020 CHI conference on human factors in computing systems*, pp. 1–8. https://doi.org/10.1145/3334480.3382814

Gao, Y., & Huang, C.-M. (2019). Pati: A projection-based augmented table-top interface for robot programming. In *Proceedings of the 24th international conference on intelligent user interfaces*, pp. 345–355. https://doi.org/10.1145/3301275.3302326

Gašpar, T., Deniša, M., Radanovič, P., Ridge, B., Savarimuthu, T. R., Kramberger, A., Priggemeyer, M., Roßmann, J., Wörgötter, F., Ivanovska, T., et al. (2020). Smart hardware integration with advanced robot programming technologies for efficient reconfiguration of robot workcells. *Robotics and Computer-Integrated Manufacturing, 66*, 101979. https://doi.org/10.1016/j.rcim.2020.101979

Gradmann, M., Orendt, E. M., Schmidt, E., Schweizer, S., & Henrich, D. (2018). Augmented reality robot operation interface with google tango. In *ISR 2018; 50th international symposium on robotics*, pp. 1–8. VDE. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8470593&isnumber=8470559

Hart, S. G., & Staveland, L. E. (1988). Development of nasa-tlx (task load index): Results of empirical and theoretical research. *Advances in Psychology, 52*, 139–183. https://doi.org/10.1016/S0166-4115(08)62386-9

Kohrt, C., Stamp, R., Pipe, A., Kiely, J., & Schiedermeier, G. (2013). An online robot trajectory planning and programming support system for industrial use. *Robotics and Computer-Integrated Manufacturing, 29*(1), 71–79. https://doi.org/10.1016/j.rcim.2012.07.010

Lin, K., Rojas, J., & Guan, Y. (2017). A vision-based scheme for kinematic model construction of re-configurable modular robots. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 2751–2757. IEEE. https://doi.org/10.1109/IROS.2017.8206103

Makhataeva, Z., & Varol, H. A. (2020). Augmented reality for robotics: A review. *Robotics, 9*(2), 21. https://doi.org/10.3390/robotics9020021

ManpowerGroup: ManpowerGroup's Talent Shortage survey. Retrieved February 06, 2023, from https://go.manpowergroup.com/talent-shortage#read-report.

Manring, L., Pederson, J., Potts, D., Boardman, B., Mascarenas, D., Harden, T., & Cattaneo, A. (2020). Augmented reality for interactive robot control. In *Special topics in structural dynamics & experimental techniques* (Vol. 5, pp. 11–18). Springer. https://doi.org/10.1007/978-3-030-12243-0_2

Michalos, G., Karagiannis, P., Makris, S., Tokçalar, Ö., & Chryssolouris, G. (2016). Augmented reality (ar) applications for supporting human-robot interactive cooperation. *Procedia CIRP, 41*, 370–375. https://doi.org/10.1016/j.procir.2015.12.005

Naismith, L. M., Cheung, J. J., Ringsted, C., & Cavalcanti, R. B. (2015). Limitations of subjective cognitive load measures in simulation-based procedural training. *Medical Education, 49*(8), 805–814. https://doi.org/10.1111/medu.12732

Nikolaidis, S., Ramakrishnan, R., Gu, K., & Shah, J. (2015). Efficient model learning from joint-action demonstrations for human-robot collaborative tasks. In *2015 10th ACM/IEEE international conference on human-robot interaction (HRI)*, pp. 189–196. IEEE. https://doi.org/10.1145/2696454.2696455

Occupational Safety and Health Administration. (2022). OSHA technical manual TED 01-00-015. US Department of Labor, Washington, DC, USA. Retrieved from https://www.osha.gov/enforcement/directives/ted-01-00-015-5

Ong, S., Yew, A., Thanigaivel, N., & Nee, A. (2020). Augmented reality-assisted robot programming system for industrial applications. *Robotics and Computer-Integrated Manufacturing, 61*, 101820. https://doi.org/10.1016/j.rcim.2019.101820

OpenRAVE. (2018). IKFast kinematics solver. Retrieved February 06, 2023, from https://en.wikipedia.org/wiki/OpenRAVE#IKFast.

Pan, Y., Chen, C., Li, D., Zhao, Z., & Hong, J. (2021). Augmented reality-based robot teleoperation system using rgb-d imaging and attitude teaching device. *Robotics and Computer-Integrated Manufacturing, 71*, 102167. https://doi.org/10.1016/j.rcim.2021.102167

Pan, Z., Polden, J., Larkin, N., Van Duin, S., & Norrish, J. (2012). Recent progress on programming methods for industrial robots. *Robotics and Computer-Integrated Manufacturing, 28*(2), 87–94. https://doi.org/10.1016/j.rcim.2011.08.004

Rajendran, P., Thakar, S., & Gupta, S. K. (2019). User-guided path planning for redundant manipulators in highly constrained work environments. In *2019 IEEE 15th international conference on automation science and engineering (CASE)*, pp. 1212–1217. IEEE. https://doi.org/10.1109/COASE.2019.8843126

Rajendran, P., Thakar, S., Kabir, A. M., Shah, B. C., & Gupta, S. K. (2019). Context-dependent search for generating paths for redundant manipulators in cluttered environments. In *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 5573–5579. https://doi.org/10.1109/IROS40897.2019.8967865. IEEE

Robotics adoption survey finds ups, downs, and a few surprises. Retrieved November 11, 2021, from https://www.automationworld.com/factory/robotics/article/21307161/robotics-adoption-survey-finds-ups-downs-and-a-few-surprises.

Schraft, R. D., & Meyer, C. (2006). The need for an intuitive teaching method for small and medium enterprises. *Vdi Berichte, 1956*, 95. https://doi.org/10.1016/B978-008045157-2/50099-7

Smits, R., Bruyninckx, H., & Aertbeliën, E. (2011). IKFast: The robot kinematics compiler. Retrieved from http://openrave.org/docs/0.8.0/openravepy/ikfast/

Sucan, I. A., Moll, M., & Kavraki, L. E. (2012). The open motion planning library. *IEEE Robotics & Automation Magazine, 19*(4), 72–82. https://doi.org/10.1109/MRA.2012.2205651

Sweller, J. (1999). Instructional design. In *Australian educational review*. Citeseer

Sweller, J. (2011). Cognitive load theory. In *Psychology of learning and motivation* (Vol. 55, pp. 37–76). Elsevier Academic Press. https://doi.org/10.1016/B978-0-12-387691-1.00002-8

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science, 12*(2), 257–285. https://doi.org/10.1016/0364-0213(88)90023-7

Sweller, J., Van Merrienboer, J. J., & Paas, F. G. (1998). Cognitive architecture and instructional design. *Educational Psychology Review, 10*(3), 251–296.

Thoo, Y. J., Maceiras, J., Abbet, P., Racca, M., Girgin, H., & Calinon, S. (2021). Online and offline robot programming via augmented reality workspaces. arXiv:2107.01884. https://doi.org/10.48550/arXiv.2107.01884

Todd, D. J. (2012). *Fundamentals of robot technology: An introduction to industrial robots, teleoperators and robot vehicles*. Springer Business Media.

Tsamis, G., Chantziaras, G., Giakoumis, D., Kostavelis, I., Kargakos, A., Tsakiris, A., & Tzovaras, D. (2021). Intuitive and safe interaction in multi-user human robot collaboration environments through augmented reality displays. In *2021 30th IEEE international conference on robot & human interactive communication (RO-MAN)*, pp. 520–526. https://doi.org/10.1109/RO-MAN50785.2021.9515474. IEEE

Van Merriënboer, J. J., & Sweller, J. (2010). Cognitive load theory in health professional education: Design principles and strategies. *Medical Education, 44*(1), 85–93. https://doi.org/10.1111/j.1365-2923.2009.03498.x

Villani, V., Pini, F., Leali, F., & Secchi, C. (2018). Survey on human-robot collaboration in industrial settings: Safety, intuitive interfaces and applications. *Mechatronics, 55*, 248–266. https://doi.org/10.1016/j.mechatronics.2018.02.009

Yew, A., Ong, S., & Nee, A. (2017). Immersive augmented reality environment for the teleoperation of maintenance robots. *Procedia Cirp, 61*, 305–310. https://doi.org/10.1016/j.procir.2016.11.183