Utilizing Augmented Reality to Create and Control a Digital Twin of a Collaborative Robot

Mohamed-Kamel Koumenji; Marcus Soll; Louis Kobras; Jan Haase

2025 7th Experiment@ International Conference (exp.at'25), 2025/2026

IEEE Xplore: https://ieeexplore.ieee.org/document/11348279/

DOI: 10.1109/exp.at2565440.2025.11348279

Utilizing Augmented Reality to Create and Control a Digital Twin of a Collaborative Robot

Mohamed-Kamel Koumenji
NORDAKADEMIE gAG Hochschule der Wirtschaft
25337 Elmshorn, Germany
mohamed.koumenji.maise21o@nordakademie.org

Marcus Soll
NORDAKADEMIE gAG Hochschule der Wirtschaft
25337 Elmshorn, Germany
marcus.soll@nordakademie.de

Louis Kobras
NORDAKADEMIE gAG Hochschule der Wirtschaft
25337 Elmshorn, Germany
louis.kobras@nordakademie.de

Jan Haase
NORDAKADEMIE gAG Hochschule der Wirtschaft
25337 Elmshorn, Germany
jan.haase@nordakademie.de

***Abstract*—Robotic automation is an integral part of modern industry. It is thus imperative to prepare students of engineering and computer science to handle industrial robots. However, as stopping a single machine can impact the productivity of an entire facility, training on live machinery might not always be feasible. Likewise, setting up enough laboratory space and equipment for students to use during their studies requires much available space and funding and thus also might not always be feasible. This paper presents a prototype of a digital twin for a collaborative robot arm. The digital twin makes use of augmented reality for visualisation and connects to either a physical robot or a simulated robot controller inside a virtual machine. Several aspects of the implementation as well as limitations and lessons learned are described.**

***Index Terms*—Augmented Reality, Collaborative Robots, Digital Twin, Unreal Engine, Universal Robots UR5e, HoloLens**

I. Introduction

The use of automation technology is something without which modern production lines are unthinkable. With the spending on robotic process automation being expected to grow nearly exponentially by 2032 in one prognosis [2] and the global market revenue for industrial robots on an upwards trend at least until 2028 in another [3], it is safe to say that robotic automation is here to stay. As such, it is imperative to include the subject of robotics in the education of students of engineering and computer science to help them prepare to work in modern production environments.

Part of robotic automation is the notion of collaborative robots. A collaborative robot is an industrial robot that employs special safety measures to enable collaborative work alongside humans [4, Sec. 5.10]. An industrial robot is a machine that can be freely programmed in multiple axes to manipulate and interact with its environment [4]. In contrast to industrial robots, which may move with high velocity and large loads and are thus potentially dangerous to bystanders, collaborative robots (or *Cobots*) make use of one or more of several features to increase safety within the working area, namely: comparatively lighter loads and slower speeds than standard industrial machines, an automated emergency stop functionality, or hand guided movement [5]. Due to the special safety considerations, collaborative robots can share a working area with humans (so called *collaborative workspaces*; [4, 3.5]) which allows for automation to be weaved into human processes while still adhering to the same general principles of industrial automation (such as the method of programming, axis control, etc.).

While a collaborative robot in itself can be considered safe, a laboratory space may not necessarily be, due to environmental hazards like other laboratory equipment or other lab workers that need to be kept aware about a robot's movement. Additionally, setting up a work space to train working with a robot requires both (continuous) funds [6, p. 221] and space, both of which are usually limited, in turn limiting how many students at a time can interact with a robot. Furthermore, a work space that has been set up once may not be mobile or easily movable (cf. [6, p. 224]). By providing a digital twin (DT) that students can interact with using augmented reality (AR), all of these issues can be mitigated: an AR simulation cannot hurt bystanders[1] and can be taken somewhere without hazards; it is cheaper and more space efficient to purchase more AR devices than to set up more work spaces; and students can borrow an AR device to practise at home. An extensive literature survey conducted by Suzuki et al. demonstrates that AR applications for collaborative robotics are an active field of research with diverse foci and use cases [7], further cementing the viability of this study[2]. Incidentally, the AR device also makes for a good demonstration object for

This research was part of the project *Flexibel kombinierbare Cross-Reality Labore in der Hochschullehre: zukunftsfähige Kompetenzentwicklung für ein Lernen und Arbeiten 4.0* (CrossLab) [1], which is funded by the *Stiftung Innovation in der Hochschullehre*, Germany.

[1]Obviously, there is still a risk of a rogue fist to consider. However as AR – in contrast to VR – doesn't entirely impede vision, one can assume that people using an AR device can still take care not to punch their lab partner.

[2]As it were, this study neatly slots into the categories of *Facilitate Programming*, *Support Real-time Control and Navigation* and *Improve Safety* of the different purposes Suzuki et al. [7, Sec. 5] identify.

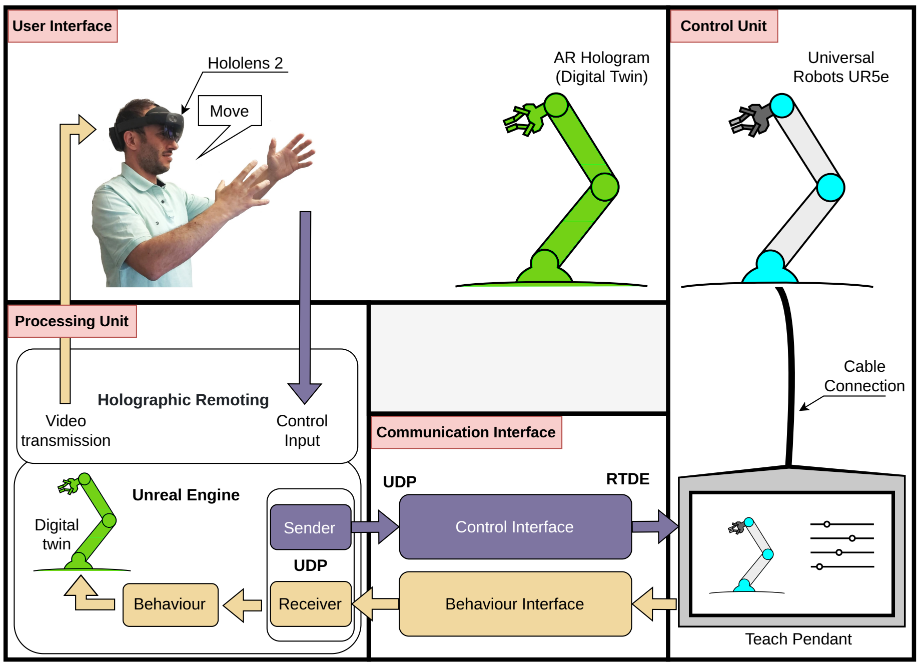

Figure 1. Schematic representation of the information flow between the DT, its AR representation, and the actual robot. The four principal components – User Interface, Processing Unit, Communication Interface, and Control Unit – are highlighted with separate borders and labels.

interested parties that cannot visit the laboratory. This paper presents a prototype of an AR DT of the *UR5e* collaborative robot developed by Universal Robots [14] which is controlled using a Microsoft HoloLens 2 [15]. The Cobot is equipped with a wrist-mounted camera [16] and 2F-85 2-finger [17] by Robotiq [18].

Compounding on the issues of safety, space, and up-front funding, stopping a robot – and thus possibly stopping an entire production line – can be expensive for companies, hence methods are needed through which a robot can be programmed with minimal downtime. One possible solution for this is programming a robot in a simulated environment before transferring the completed program to the live equipment. This work proposes the use of an AR DT to facilitate students learning how to engage with a collaborative robot 1) in a safe environment 2) through scalable means 3) that are also easily transferable to different locations. Ağca identifies a gap in research on DTs in an educational context and states that DTs, along with other technologies like artificial intelligence, may be used to address future educational needs, especially in the context of virtual or online learning and as stopgaps for societal inequality [8]. Considering this alongside the sudden surge of necessity concerning remote and hybrid teaching that arose during the COVID-19 pandemic [9], [10] grants a certain amount of gravitas to any effort of enabling mobility and scalability in education.

Although an accepted general definition of the concept of a *digital twin* does not seem to exist yet [11], a coarse concept of DTs can be assumed to be a digital system that mirrors the state and operations of a physical system (thus, "twinning" a physical system) [12, p. 1017]. Kritzinger et al. differentiate between a *digital model*, a *digital shadow*, and a *digital twin*, identified by the degree of integration concerning the automated information exchange between the physical and digital system. The digital model is defined by completely manual information exchange, the digital shadow receives automated information updates by the physical object, and the DT features automated information exchange in both directions [12].

II. General System Architecture

The system architecture comprises four core components: the **User Interface**, the **Processing Unit**, the **Communication Interface**, and **the Control Unit**. The schematic information flow between the different components is displayed in Fig. 1.

The User Interface (Human-Machine Interface) is implemented through the HoloLens, enabling interaction between the user, the DT (virtual representation), and the Real Twin (physical robot). The HoloLens integrates real and virtual content by rendering the DT as a holographic object within the user's real-world environment. The real-time transmission of graphical content, computed by the Unreal Engine, is achieved via Holographic Remoting[3] to the HoloLens. The Interaction with the digital and real twins is realised through the user's hand gestures and voice commands, which are captured and interpreted by the HoloLens. These inputs are then transmitted in real time to the Processing Unit (Unreal Engine) via Holographic Remoting.

The Processing Unit is implemented using the Unreal Engine. It receives user inputs from the HoloLens and translates

[3]*Holographic Remoting* is a HoloLens app that allows a user to display a VR model streamed from another device.

them into commands for the Control Unit, which executes the physical movements. Additionally, the Unreal Engine receives behavioural data (e.g., joint positions) from the Real Twin via the Control Unit and transfers this data to the DT to update its state in real time. The Unreal Engine is also responsible for computing the graphical content of the DT, which is transmitted to the HoloLens via Holographic Remoting.

The Control Unit consists of the real twin and the Teach Pendant. It receives movement commands from the user via the HoloLens and the Unreal Engine and executes them as physical actions of the robot. Simultaneously, the Control Unit collects data (e.g., joint positions) and transmits this data to the Processing Unit via the Communication Interface. This data is used to update the state of the DT in real time, ensuring accurate feedback for the user. The Teach Pendant can either be simulated[4] or connected to a physical robot.

The Communication Interface enables the bidirectional transfer of control commands and behavioural data between the Processing Unit and the Control Unit. Data transmission is facilitated using protocols such as UDP (User Datagram Protocol) for low-latency control commands and RTDE (Real-Time Data Exchange) for sensor data feedback. This ensures precise and responsive communication between the system components.

III. Implementing the Model in the Game Engine

To ensure a precise geometric representation of the DT, the CAD models provided by the manufacturers of the different components (*Universal Robots* for the UR5e robot [19] and *Robotiq* for the 2F-85 gripper [20]) were utilized. Given that the Unreal Engine lacks native support for CAD files, the models were transformed through an intermediary process involving Autodesk Inventor [21] and Blender [22]. The CAD models, initially downloaded in STEP file format, were imported into Autodesk Inventor, where they were decomposed into their constituent movable components. This decomposition is essential, as not doing so would result in the models being interpreted as rigid, atomic entities in Blender and Unreal Engine, thereby complicating the subsequent implementation of rotational kinematics for their individual components.

Following decomposition, the components of each model were exported from Autodesk Inventor in Wavefront OBJ format and imported into Blender. Within Blender, each component underwent individual processing, which included the assignment and alignment of a local coordinate system.

The alignment of the local coordinate system for the UR5e components adheres to the Denavit-Hartenberg (D-H) convention: The Z-axis defines the rotational axis, the X-axis is derived as the cross product of adjacent Z-axes, and the Y-axis completes the right-handed coordinate system. For the gripper joints, the Z-axis was similarly designated as the rotational axis. This methodological approach ensures an accurate digital representation of the mechanical motion transmission across the joints and linkages of both the robot and the gripper, faithfully replicated in the DT within Unreal Engine.

Upon completion of processing, each component was exported from Blender to Unreal Engine in FBX file format. Due to the disparity between Blender's right-handed coordinate system and Unreal Engine's left-handed system, the Z-axis was oriented upward and the X-axis forward in Blender's export settings to preserve the correct alignment of local coordinate systems during the transfer process.

All components were assigned *Unlit Materials* in the engine. This follows the recommendations by Microsoft due to concerns regarding the performance of the HoloLens [23], which we can confirm improved the performance noticeably[5]. Both the robot itself as well as the gripper were realised as classes containing a hierarchy of the different parts – this facilitated the handling of relative data each part holds with regards to is predecessor, such as position.

IV. Implementing the Behaviour of the DT

For an accurate representation of the physical robot's behaviour, the joint configurations and *Tool Center Point* (TCP) position of the physical robot are captured in real-time from the *PolyScope*[6] control software and transmitted to the Unreal Engine. The data is used to continuously update the joint positions and TCP of the digital twin. This approach relies on the precise geometric and kinematic replication of the physical robot, ensuring that the DT is fully synchronized to the real twin. The unified control through *PolyScope* allows seamless transfer of programmed sequences between the physical robot and its digital twin, significantly streamlining development and testing processes. Behavioural data acquisition and transmission to the Unreal Engine were implemented through a dedicated *Behaviour Interface*, utilizing the RTDE protocol, implemented via the `ur_rtde` library [24], to communicate with the robot controller. This library also enables access to Dashboard features (such as motor start) and the API of the gripper.

Using the `Receive Interface` provided by RTDE, the joint positions and gripper TCP coordinates of the robot can be retrieved and transmitted to the game engine. The data is transformed into JSON (*JavaScript Object Notation*). Initial attempts transmitted these data in a continuous loop which could be measured at about 1200 updates per second; however, as RTDE refreshes at a maximum of 500 Hz and the HoloLens is capped at 60 FPS, the transmit frequency was throttled to 128 Hz to reduce system load. As the robot moves with at maximum 1 m/s this translates to a movement of about 8 mm per update. 128 Hz allows for smooth movement both on the HoloLens as well as a commercially available PC monitor that runs with 120 Hz.

[4]Universal Robots offers a virtual machine which can emulate the robot and Teach Pendant. We used version 5.17.0, which is no longer available at the time of writing.

[5]With lit materials, we encountered issues in simulating the DT smoothly even on computers with strong GPUs.

[6]The controller and programming software of the UR cobots.

The data received by the game engine[7] is then passed to actors implementing a specific interface. This facilitates the addition or removal of specific actors since a new actor just needs to implement said interface. The interface then triggers an update-function in an actor; e.g., the robot updates its joint and TCP positions in accordance with the received data.

One notable difference between the robot itself and the gripper is that the gripper is a third-party add-on to the robot, not a part of the robot itself; thus, while a *PolyScope* plugin to use the gripper as part of a robot program exists, the gripper is itself not simulated in *PolyScope*, leading to the need to implement a separate control mechanism for the DT. This is done by utilising the digital I/O interface of the Cobot that can be and written to and read from within *PolyScope*. Thus, a corresponding signal both for opening and closing can be programmed into the digital I/O ports that can be interacted with by the DT. Implementing the opening and closing of the gripper for the DT makes use of the same data handling mechanisms as controlling the robot itself. As the opening or closing is just a rotation of the fingers in opposing directions, most functionality can be trivially adapted from the robot. The gripper actor loops over an increment or decrement of its joint positions until the desired position has been reached. However, the first attempt of this loop ran too fast for the visualisation to keep up, so a delay had to be introduced.

The gripper features a *Sphere Collision* component for work pieces to interact with. Virtual work pieces[8] were also implemented as actors with a *Sphere Collision* component[9]. To identify when a work piece actor should be considered gripped, the intersection between the work piece collision and the gripper collision is detected. If such an intersection is detected during a closing action of the gripper, the work piece actor is anchored to the gripper to simulate moving it[10], disabling the work piece's own physics to avoid artefacts in the simulation (such as jittering). To let go of a work piece the gripper model is visually opened and the anchoring is cancelled. A small delay is interjected so that the work piece is visually only let go by the gripper once the fingers have started opening. Physics are then re-enabled for the work piece so that it can fall down and collide with the virtual table.

V. User Control

The control of both the physical robot and its digital twin is based on the teach method, which allows the user to manually manipulate the end effector within the workspace. Traditionally, this is achieved by directly applying force to guide the end effector, resulting in the displacement of the TCP position in the working space and generating a joint angle configuration that defines the robot's joint positions.

In this work, however, the displacement of the end effector is not induced by the application of direct force but rather by the user's hand movements. When the user moves their hand, the HoloLens captures the 3D positions of the hand during the motion and transmits them sequentially as input events to the Unreal Engine via Holographic Remoting. The data is then forwarded to the Control Interface, which calculates the displacement between each pair of consecutive positions (the latest and the newest one), transforms it, and transmits it to the Polyscope control software. The software applies this information to the robot's TCP coordinates, generating a new joint configuration. This updated configuration is simultaneously transmitted to the DT via the behaviour Interface and to the physical twin via a wired connection, ensuring real-time synchronization between the two.

The user can use both hands to manipulate the gripper TCP of the DT. After using a pinch gesture [11], the right hand of the user can be used to control the TCP position of the virtual robot. Its spatial orientation can be controlled using the left hand. The hand controls are mutually exclusive, i.e. only one hand can be used at a time. While this may seem inconvenient at first – since it forces the user to continuously switch between hands – it allows for separation of the position and orientation manipulations, thus preventing accidental orientation changes during position changes and vice versa, while also allowing for easier rotation control. This control mechanism utilizes the pre-defined input methods `UxtRightGrab` and `UxtLeftGrab`, respectively. As the actor class for the Cobot cannot directly interact with the hand positions, another actor was created at both palms which is moved in relation to the hands. The orientation of the palm actors in the three spatial axes is then relayed to the Cobot controller, where it is mapped to the corresponding gripper TCP data. The movement vector is calculated by getting the TCP position from the robot, calculating the difference to the position that had been passed, and adding the difference to the current position[12]. A check is then performed using the RTDE interface whether an inverse kinematics solution for the target position exists and if the target position is within the safety limits of the robot before it is sent to the robot controller.

Variable thresholds and a parameter for impact were included that modify how strongly the robot reacts to hand gestures, with the thresholds cutting of miniscule and very large

[7]The `SocketIOClient-Unreal`-Plugin [25] was used to facilitate UDP and JSON handling.

[8]In our laboratory, students are tasked with programming the Cobot to stack cubes. To keep the DT similar to the physical laboratory, the implemented work pieces were also cubes of the same size as the physical ones.

[9]In addition, a *Static Mesh* component is used for the visualisation.

[10]The virtual work piece object is visually snapped to the gripper and its width is used to interpolate the opening angle of the fingers, widening or closing the fingers as necessary.

[11]The HoloLens, equipped with an advanced hand-tracking system, can recognize spatial gestures. The pinch gesture was selected as the primary input method.

[12]Note that the function `getActualTCPPose()` the RDTE interface provides returns the TCP position as a rotation vector, while the hand movement is described as a vector containing the magnitude and direction of the hand displacement in the respective dimension. To adjust the TCP position according to the hand movement, both vectors are converted into quaternions, added together, and then transformed into a rotation vector. The resulting vector is subsequently transmitted to *Polyscope* to update the TCP position. Quaternions ensure a unique solution, in contrast to rotational matrices that can yield multiple valid solutions due to their inherent ambiguities. Before sending the target position to the robot controller.

movements and the impact defining how strongly movement of a hand is scaled up for the robot. These parameters, among others, are displayed in a virtual floating panel and can be manipulated at will (within certain limits) using sliders. Other sliders include the gripping force and opening and closing speed for the gripper as well as a limiting threshold for the maximum gripper opening. The floating panel also features buttons to open, close, and initialise[13] the gripper. The size of a newly created virtual work piece can also be manipulated using a slider.

The DT also accepts voice commands. To actually be able to move the robot via hand gestures, it first needs to be put into *freedrive*[14]. This is achieved using the voice command *"start freedrive"*[15]. Further voice commands were implemented that manipulate the freedrive speed. A new work piece can be created, picked, and placed using the voice commands *"cube"*, *"pick"*, and *"place"*, respectively. The Cobot can be moved into predetermined positions using *"move start"*, *"move home"*, and *"move zero"*, where *start* corresponds to the starting place of a test application (see Sec. VI); *zero* sets all joint positions to 0°, and *home* moves the cobot in the 90° angled position that corresponds to the home position in *PolyScope*. Further voice commands exist to create a new programme, add waypoints to the programme, and manipulate the gripper and work pieces as part of a programme. A defined programme can be launched using the command *"program start"*.

VI. Evaluation

A simple test application was programmed and run in *PolyScope*, with the resulting behavioural data being transmitted to the DT. It was then observed whether the simulated robot within *PolyScope* and the DT in Unreal behaved identically. It could be confirmed visually that the simulated Cobot and the DT moved virtually identically (with a small delay between the two models). The same routine also successfully tested the creation, picking, and placing of virtual work pieces.

A first formal usability evaluation of the prototype has been conducted using a usability checklist. Said usability checklist [13] is developed by the CrossLab project[16] and rates the usability of a laboratory or experiment on eleven different scales. According to the authors of the checklist, it is based on research on both usability in general as well as usability for cross reality laboratories. Applying the checklist to the prototype yields the result displayed in Fig. 2.

[13]When the Cobot is booted up, the gripper is in an uninitialised state. Only after running the initialisation the gripper accepts commands.

[14]In freedrive mode, the motor controls of the robot are released and the robot can be freely moved around. For a physical robot, this is essential for manual teaching, however it also disengages some safety mechanisms. The physical robot only enables freedrive via a dead man's switch connected to the robot via cable to ensure someone is around when the robot moves freely.

[15]Analogously, *"stop freedrive"* exits freedrive mode.

[16]The current version of the checklist can be accessed under a Creative Commons licence. However, since it is currently an early release, papers about it are in progress. Still, we believe that it is in a state we feel comfortable to apply it to our DT.

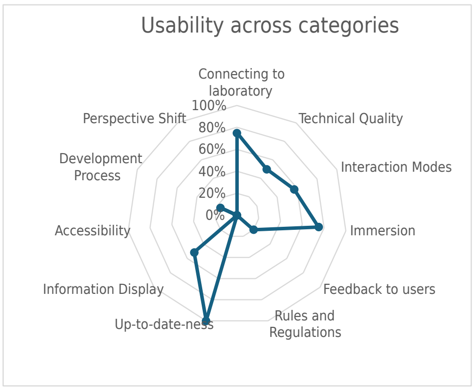

Figure 2. Usability diagram of the application

The exact percentage scores are not especially relevant as the checklist was designed with making general observations and giving general pointers in mind. That said, some things are immediately evident. The prototype reaches an excellent score in *Up-to-date-ness* (measuring the topicality and relevance of the investigated technology), with additional good scores in *Connecting to laboratory* (measuring the technical setup required) and *Immersion*. At the same time, glaring issues are made obvious with regards to *Rules and Regulations* (e.g., concerning a transparent privacy policy), *Accessibility*, and *Perspective Shift* (which takes the evaluative perspective away from the laboratory developer or educator and instead focuses on learners or administrative personnel). All these issues can be interpreted as "To-Dos" for the further development of the prototype.

VII. Caveats and Lessons Learned

While the prototype in itself works as intended, there are still some caveats that need to be noted as well as some stumbling blocks one needs to be aware of before trying to replicate this project.

a) Technical Reliability: For its price tag, we found the capabilities of the HoloLens to detect motion controls to be somewhat lacking. While this issue also correlates with computational capacity, even with a decently strong machine we still occasionally experienced problems with detecting hand gestures.

b) Model Availability: While both Universal Robots and Robotiq provide STEP files for their products, using them in an Unreal Engine AR application does not seem to be natively possible. Instead, the provided STEP files had to be pre-processed first in a CAD programme (to separate a singular model of the whole technological artefact into components for its several parts), after which the export of the CAD programme had to piped through a 3D modelling programme to convert it into a format Unreal could work with – while having to deal with the different programmes utilizing different coordinate systems.

c) Sustainability: Shortly after we acquired several HoloLenses to use with our laboratories, an official notice about them being discontinued was issued [26]. Thus, even while we now have a working prototype, once a HoloLens breaks, it is gone for good, and porting the software to another type of AR display may well be more effort than to programme it from scratch for the new platform. As such, we have to assume that this software may fall out of use even sooner than might be expected. Even before this, sustainability issues were expected, as support for the HoloLens was dropped from Unreal Engine after version 5.0 (with the current version during development being 5.3), meaning that any further development needs to rely on outdated software.

d) Computational Power: The prototype presented here may seem like a light-weight application. In fact, it was possible to launch both a virtual robot controller as well as this prototype on a Surface Pro 7. However, considerable computational capacity is required for it to actually run smoothly and stably.

VIII. Conclusion

This paper presents a prototype application that implements a virtual reality DT of a collaborative robot. The bulk of the paper (Sec.s II–V) consists of descriptions of the system architecture and noteworthy implementation details one might need to know in order to replicate this project. A sketch of the data flow in the prototype is displayed in Fig. 1. Using UDP and RTDE to transfer data between the HoloLens, the game engine, and the robot controller, a DT could be implemented that featured a bi-directional real-time mirroring of the physical robot. Due to the nature of the system architecture, a physical robot can be exchanged for a simulated one using the virtual machine simulator provided by Universal Robots by simply changing an IP address.

References

[1] I. Aubel, S. Zug, A. Dietrich, J. Nau, K. Henke, P. Helbing, D. Streitferdt, C. Terkowsky, K. Boettcher, T. R. Ortelt, M. Schade, N. Kockmann, T. Haertel, U. Wilkesmann, M. Finck, J. Haase, F. Herrmann, L. Kobras, B. Meussen, M. Soll, and D. Versick, “Adaptable digital labs - motivation and vision of the crosslab project,” in *2022 IEEE German Education Conference (GeCon)*, 2022, pp. 1–6.

[2] Precedence Research and GlobeNewswire, “Spending on robotic process automation (RPA) software worldwide from 2020 to 2023 (in billion u.s. dollars) [graph],” Available: https://www.statista.com/statistics/1309384/worldwide-rpa-software-market-size/, 2023.

[3] Inkwood Research, “Size of the global market for industrial robots from 2018 to 2020, with a forecast for 2021 to 2028 [graph],” Available: https://www.statista.com/statistics/760190/worldwide-robotics-market-revenue/, 2021.

[4] ISO 10218-1:2011, “Robots and robotic devices – Safety requirements for industrial robots – Part 1: Robots,” International Organization for Standardization, Geneva, CH, Standard, 2011. [Online]. Available: https://www.iso.org/standard/51330.html

[5] ISO/TS 15066:2016, “Robots and robotic devices – Collaborative robots,” International Organization for Standardization, Geneva, CH, Technical Specification, 2016. [Online]. Available: https://www.iso.org/standard/62996.html

[6] M. Soll, L. Kobras, I. Aubel, S. Zug, C. Terkowsky, K. Boettcher, T. R. Ortelt, N. Kaufhold, M. Schade, R. Sritharan, J. Steinert, U. Wilkesmann, P. Helbing, J. Nau, D. Streitferdt, A. Baum, A. Bock, J. Haase, F. Herrmann, B. Meussen, and D. Versick, “It’s a marathon, not a sprint: Challenges yet to overcome for digital laboratories in education,” in *Smart Technologies for a Sustainable Future*, M. E. Auer, R. Langmann, D. May, and K. Roos, Eds. Cham: Springer Nature Switzerland, 2024, pp. 220–231. [Online]. Available: https://doi.org/10.1007/978-3-031-61905-2_22

[7] R. Suzuki, A. Karim, T. Xia, H. Hedayati, and N. Marquardt, “Augmented reality and robotics: A survey and taxonomy for ar-enhanced human-robot interaction and robotic interfaces,” in *CHI Conference on Human Factors in Computing Systems*. New Orleans LA USA: ACM, 2022, p. 1–33. [Online]. Available: https://dl.acm.org/doi/10.1145/3491102.3517719

[8] R. K. Ağca, “Using digital twins in education from an innovative perspective: Potential and application areas,” *Education Mind*, vol. 2, no. 2, p. 65–74, Dec. 2023. [Online]. Available: https://doi.org/10.58583/Pedapub.EM2306

[9] M. P. A. Murphy, “Covid-19 and emergency elearning: Consequences of the securitization of higher education for post-pandemic pedagogy,” *Contemporary Security Policy*, vol. 41, no. 3, p. 492–505, Jul. 2020. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/13523260.2020.1761749

[10] K. A. A. Gamage, D. I. Wijesuriya, S. Y. Ekanayake, A. E. W. Rennie, C. G. Lambert, and N. Gunawardhana, “Online delivery of teaching and laboratory practices: Continuity of university programmes during covid-19 pandemic,” *Education Sciences*, vol. 10, no. 10, p. 291, Oct. 2020. [Online]. Available: https://www.mdpi.com/2227-7102/10/10/291

[11] A. Berisha-Gawlowski, C. Caruso, and C. Harteis, *The Concept of a Digital Twin and Its Potential for Learning Organizations*. Cham: Springer International Publishing, 2021, p. 95–114. [Online]. Available: https://link.springer.com/10.1007/978-3-030-55878-9_6

[12] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, “Digital twin in manufacturing: A categorical literature review and classification,” *IFAC-PapersOnLine*, vol. 51, no. 11, p. 1016–1022, 2018. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2405896318316021

[13] F. Herrmann, M. Soll, L. Kobras, M. Schade, K. Boettcher, D. Kaiser, I. A. Aubel, N. Kaufhold, and P. Helbing, “Usability checklist for cross reality laboratories,” Jan. 2025. [Online]. Available: https://zenodo.org/doi/10.5281/zenodo.14329173

Web Links

[14] Universal Robots A/S, “UR5e lightweight, versatile cobot,” https://www.universal-robots.com/products/ur5-robot/, last accessed 2025-05-07.

[15] Microsoft, “Microsoft hololens | microsoft learn,” https://www.microsoft.com/en-us/hololens, last accessed 2025-05-07.

[16] Robotiq, “Wrist camera | robotiq,” https://robotiq.com/products/wrist-camera, last accessed 2025-05-07.

[17] ——, “Adaptive grippers | robotiq,” https://robotiq.com/products/adaptive-grippers#Two-Finger-Gripper, last accessed 2025-05-07.

[18] ——, “Empower people, boost productivity, enhance adaptability | robotiq,” https://robotiq.com/, last accessed 2025-05-07.

[19] Universal Robots A/S, “Universal robots - robot step file - ur5e - e-series,” https://www.universal-robots.com/download/mechanical-e-series/ur5e/robot-step-file-ur5e-e-series/, last accessed 2025-05-07 (needs login).

[20] Robotiq, “Support | robotiq,” https://robotiq.com/support, last accessed 2025-05-07.

[21] Autodesk Inc., “Autodesk inventor software | get prices & buy official inventor 2025,” https://www.autodesk.com/eu/products/inventor/overview, last accessed 2025-05-07.

[22] Blender, “blender.org - home of the blender project - free and open 3d creation software,” https://www.blender.org/, last accessed 2025-05-07.

[23] Cameron-Micka, “Mixedreality-graphicstools-unreal/docs/lighting.md at main · microsoft/mixedreality-graphicstools-unreal · github,” https://github.com/microsoft/MixedReality-GraphicsTools-Unreal/blob/main/Docs/Lighting.md, last accessed 2025-05-07.

[24] “Introduction — ur_rtde 1.6.1 documentation,” https://sdurobotics.gitlab.io/ur_rtde/introduction/introduction.html, last accessed 2025-05-07.

[25] getnamo, lehuan5062, tamaynard, mikeseese, LordNed, gmpreussner, bapin93, Deams51, tlightsky, KneshiHH, namse, anadin, iambeeblebrox, seon geun, finger563, dobby5, staskjs, rwinright, Lootheo, knapsu, and ASpookieGhost, “Github - getnamo/socketioclient-unreal: Socket.io client plugin for the unreal engine.” https://github.com/getnamo/SocketIOClient-Unreal, last accessed 2025-05-07.

[26] T. Warren, “Microsoft is discontinuing its hololens headsets | the verge,” https://www.theverge.com/2024/10/1/24259369/microsoft-hololens-2-discontinuation-support, last accessed 2025-05-07.