Construction Robotics (2022) 6:57–73
https://doi.org/10.1007/s41693-022-00068-1

ORIGINAL PAPER

Real-time state synchronization between physical construction robots and process-level digital twins

Ci-Jyun Liang[1] · Wes McGee[2] · Carol C. Menassa[1] · Vineet R. Kamat[1]

Received: 19 July 2021 / Accepted: 4 March 2022 / Published online: 22 March 2022

Abstract

This research focuses on developing a robot digital twin (DT) and the communication methods to connect it with the corresponding physical robot in collaborative human–robot construction work. Robots are being increasingly deployed on construction sites to assist human workers with physically demanding work tasks. Robot simulations in a process-level DT can be used to extend design models, such as building information modeling, to the construction phase for real-time monitoring of robot motion planning and control. Robots can be enabled to plan work tasks and execute them in the DT simulations. Once simulated tasks and trajectories are approved by human workers, commands can be sent to the physical robots to perform the tasks. However, a system to bridge a virtual DT and a physical robot and allow for such communication to occur is a capability that has not been readily available thus far, primarily due to the complexity involved in physical robot operations. This paper discusses the development of a system to bridge robot simulations and physical robots in construction and digital fabrication. The Gazebo robot simulator is used for DT, and the robot operating system is leveraged as the primary framework for bi-directional communication with the physical robots. The virtual robots in Gazebo receive planned trajectories from motion planners and then send the commands to the physical robots for execution. Two different robot control modes, i.e., joint angle control mode and Cartesian path control mode, are developed to accommodate various construction strategies. The system is implemented in a digital fabrication case study with a full-scale KUKA KR120 six-degrees-of-freedom robotic arm mounted on a track system. We evaluated the system by comparing the data transmission time, joint angles, and end-effector pose between the virtual and physical robot using several planned trajectories and calculated the average and maximum mean square errors. The results showed that the proposed real-time process-level robot DT system can plan the robot trajectory inside the virtual environment and execute it in the physical environment with high accuracy and real-time performance, offering the opportunity for further development and deployment of the collaborative human–robot work paradigm on real construction sites.

Keywords Digital twin · Co-robots · Robot operating system · Human–robot collaboration

1 Introduction

The deployment of collaborative robots on construction sites is believed to have the potential to relieve safety issues and chronic occupational disease (Lundeen et al. 2019) and further improve the efficiency of the construction process (Eversmann et al. 2017). For instance, the construction robot can partner with human workers on job-sites to assist with physically demanding tasks, while human workers focus on the robot control or work process plan (Liang et al. 2020, 2022). This is similar to the assembly line in the manufacturing industry, where the robots focus on repetitive and precise motion control tasks, and humans focus on planning and quality-checking functions. Human–robot interaction is

✉ Ci-Jyun Liang
cjliang@umich.edu

Wes McGee
wesmcgee@umich.edu

Carol C. Menassa
menassa@umich.edu

Vineet R. Kamat
vkamat@umich.edu

[1] Civil and Environmental Engineering, University of Michigan, 2350 Hayward Street, 2340 G.G. Brown Building, Ann Arbor, MI 48109, USA

[2] Taubman College of Architecture and Urban Planning, University of Michigan, 2000 Bonisteel Boulevard, Ann Arbor, MI 48109, USA

defined as humans and robots working in a shared environment with all types of interactions (Schmidtler et al. 2015). Wang et al. (2019) classified the relationships between humans and robots into four categories: coexistence, interaction, cooperation, and collaboration. Human–robot collaboration (HRC) is among the most active interactions between humans and robots, where humans and robots are sharing the workspace and coordinating on the same task synchronously (Musić and Hirche 2017; Liang et al. 2021).

Symbiotic human–robot collaboration is one of the HRC methods applicable in solving complex tasks (Farrell et al. 2016; Nikolakis et al. 2019) by combining the expertise and complementing the proficiencies of humans and robots, which typically requires significant computational effort and training data. This type of HRC is among the most suitable for construction robots and human workers. For example, the human worker has cognitive skills, decision-making ability, and the ability to react reasonably to unexpected situations that might arise on a construction site, whereas the construction robot has the advantage of high precision, strength, and repeatability (Hentout et al. 2019).

Since the symbiotic HRC consistently engages the human and robot with each other during the process, bi-directional communication is required to minimize the interruption and ensure safety (Wang et al. 2019). In the human-to-robot direction, communication can be achieved with direct commands through the user interface to determine the robot goal or using sensors to observe human movement, such as hand gestures, and interpret the intended commands (Mohammed et al. 2017). The robot can parse the received commands and execute the work plan. In the robot-to-human direction, the human has to be informed of the robot's work plan before execution. This can be achieved by providing a virtual representation, i.e., simulation or digital twin (DT), of the robot and the environment. The robot's work plan can be demonstrated in the DT to the human in real-time and with high precision (Wang et al. 2019), allowing the human workers to make decisions based on the information.

1.1 Process-level DT and simulation

Simulation and DT play a significant role in the robotics and construction industry to manage and increase the productivity of the process, especially in the design and operation phase. They utilize digital models, such as 3D CAD models, in the virtual environment to represent physical objects and imitate the operation to verify the different designs or compare the outcome. In the simulation, different types of parameters are used to define the environment and analyze the result to determine the quality of the design. For example, a mobile robot simulation can test the performance of the path planning and localization algorithm before being actually deployed to the real robot platform (Xu et al. 2019).

The DT offers opportunities to virtually mimic the conditions of the entire physical (real) environment and integrates all associated data during the process, which allows the DT system to run several simulations simultaneously. For example, a building DT can take building information modeling (BIM) and environmental data to simulate and monitor the building performance. The construction site is an unstructured and dynamically changing environment, making it difficult for robots to perform construction tasks or for operators to program the robots. Therefore, DT provides the opportunity to mimic the work environment and offers a visual programming interface to control the construction robot (Sartori and Schlette 2021). The main advantages of DT are real-time data integration and look-ahead simulation. The DT regularly collects the current information from the physical environment and tries to achieve real-time visualization and control. In addition, it also allows look-ahead simulations using the current state for initializing the simulation, which is termed as online simulation (Song and Eldin 2012).

In the robotics industry, the DT constructs a cyber-physical system (CPS) (Aheleroff et al. 2020) where information of the current and forecast future states of the robot can be displayed for decision-making and evaluation prior to task execution (Freedy et al. 2007). The work plan of the robot can also be determined in the DT and subsequently executed on the physical robot. Madni et al. (2019) defined four levels of DT (pre-digital twin, digital twin, adaptive digital twin, and intelligent digital twin) based on the level of intelligence. The pre-digital twin and the digital twin are the common DT in the four levels. The adaptive digital twin combines user interface and machine learning with regular DT, whereas the intelligent digital twin further utilizes reinforcement learning to process the state in a partially observed and uncertain environment.

An adaptive digital twin replicates the entire physical process in real time (Colledani et al. 2009), such as the manufacturing assembly line process. Real-time is defined as whether the DT is able to complete the process correctly within pre-defined timestamps, i.e., deadlines (Shin and Ramanathan 1994). The real-time system can be categorized as a hard real-time system, firm real-time system, and soft real-time system (Kopetz 2011). The hard real-time system has to accomplish each subtask before deadlines and can cause failures upon missing any deadlines. For example, a 3D printer is considered a hard real-time system since the filament must be extruded at the right time as the extruder crosses the print bed. The firm real-time system can tolerate infrequent missing of deadlines and consider those as low-quality results. The soft real-time system can accommodate missing deadlines by reducing the quality of the result, such as live broadcasting of video streams. The real-time process-level DT has to meet all deadlines to represent the physical

environment and thus is defined as a hard real-time system (Mertens et al. 2020).

One of the major aspects of the adaptive digital twin or process-level digital twin is the synchronized model (Lu and Xu 2018). The DT first constructs the virtual model based on the physical environment, then records and tracks the changes in the physical environment and reflects them in the virtual model. The virtual model can be extracted from the designed construction model such as building information model (BIM) or scanned 3D point clouds of the as-built environment (Delbrügger et al. 2017; Marshall and Redovian 2019; Lu et al. 2020a). On the other hand, a communication mechanism is required to synchronize the data between the physical environment and the virtual model (Wang et al. 2019; Aheleroff et al. 2020). The communication needs to be bi-directional so that the virtual model can reflect the changes of the physical environment, and the user can determine the next steps in the virtual model and send the command to the physical environment. This level of data communication and connectivity is one of the challenges to applying DT in the architecture, engineering, and construction domains (Al-Sehrawy and Kumar 2020).

1.2 Research objective

To enable productive human–robot collaboration in construction work, we investigate the real-time process-level DT to bridge the virtual and physical construction robots, and characterize the extent of the state synchronization between the two systems. In this research, we focus on the robot state synchronization in the DT system and assume that the human and environmental data can be collected and integrated into the virtual simulator using our prior work (Xu et al. 2019; Lundeen et al. 2017; Feng et al. 2015; Xiao et al. 2018); therefore, only the robotic arm is included in the DT system in this paper. The robot path is planned in the virtual robot environment, and the commands are sent to the physical robot for execution. Three specific research questions are evaluated: first, how precisely can the state synchronization between two robots be achieved; second, how to ensure the work plan is executed correctly on the physical robot; and third, what kind of approach can a robot follow to determine its work plan and control commands.

The proposed framework can be adapted to any robotic arm models reflecting physical robots. We implement the system in a digital fabrication laboratory with a full-scale KUKA KR120 six-degrees-of-freedom (6DOF) robotic arm and evaluate the system by comparing a series of poses of the physical robotic arm with the virtual robotic arm. We create several complex trajectories and sets of joint angles to test the proposed system. Finally, to validate the hard real-time feature of our process-level DT system, we measure the data transmission time between the virtual robot and the physical robot.

The remainder of this paper is organized as follows. First, existing digital modeling methods and DT robotic systems are identified and reviewed to define the research gap. Second, the real-time process-level DT of the robotic construction process is developed. Third, the communication system and an algorithm for checking synchronization are introduced. Finally, experiments of robot motion planning and execution are conducted and used to evaluate the synchronization of the proposed real-time process-level DT.

2 Related work

To enable the virtual simulator to mimic the physical robot and its workspace, two aspects need to be considered. First, the virtual simulator has to reconstruct the physical environment and dynamically reflect the changes, which is the digital modeling method. Second, the virtual simulator has to plan the robot's work plan and send commands to the physical robot, which is the DT for the robotic system. The virtual simulator keeps tracking the changes of the physical robot and reflects those in the virtual robot. We discuss existing digital modeling methods and DT for construction robots in the following subsections.

2.1 Digital modeling methods

Digital modeling methods, such as 3D visualization or BIM, are used in the construction industry for design, management, and operation throughout the building life cycle (Kamat and Martinez 2005; Eadie et al. 2013). These modeling methods document the project information and provide a platform for stakeholders to record changes, collaborate, and resolve conflicts (Sampaio and Berdeja 2017; Wu et al. 2017). To achieve a productive collaboration, the model must be fully synchronized with the physical environment. It is time and cost-prohibitive to update the model manually (Ochmann et al. 2016). Thus, existing research focuses on automatically generating and updating the 3D model (Hamledari et al. 2017).

Collecting the 3D point cloud is one of the methods used for generating the 3D model of the indoor environment (Xiao et al. 2018). This type of method requires a registration method for obtaining 3D points from cameras or laser scanners (Xu et al. 2019; Feng et al. 2015; Bosché et al. 2015) and then applies segmentation methods to separate objects and reconstructs the semantic model (Dimitrov and Golparvar-Fard 2015; Macher et al. 2017; Stojanovic et al. 2018). Object recognition algorithms are also applied to identify different objects in the point cloud (Wang and Cho 2015; Lin et al. 2019). Finally, algorithms such

as site-to-BIM data transfer automation or derivative-free optimization are required to automatically update the digital model based on the identified objects (Hamledari et al. 2018; Xue et al. 2019). In the DT system, geometry assurance is developed to ensure the quality of the model (Söderberg et al. 2017; Tabar et al. 2020). The data transmission in these types of methods is from the physical environment to the virtual environment. To further control the physical robot and the environment, a DT for a robot system is required to plan the task from the digital modeling methods.

2.2 Digital twin for construction and assembly robots

Digital twins have been envisioned to be the next generation of construction cyber-physical systems that can benefit the construction industry in decision-making and monitoring (Kan and Anumba 2019). A similar approach can be used to integrate a construction robot with digital modeling methods for visualization and task planning (Tandur 2015). For example, Yang et al. (2019) utilized BIM and robot path planners to find and visualize the construction process of modular construction. Shahmiri and Ficca (2016) developed a parametric model that can directly control industrial robots to assemble the structure. Bruckmann et al. (2016) used BIM as the data source to program a cable-driven parallel robot to construct masonry buildings. Similarly, Usmanov et al. (2017) used BIM to program an industrial robotic arm to lay bricks. In addition, robot programming software also provides a simulation environment for offline programming to assemble components in the manufacturing industry (RoboDK 2021; OCTOPUZ 2021).

However, these types of systems are typically not synchronized between the virtual model and physical robot and require further adaptation to address the design-build discrepancy (Lundeen et al. 2017). For example, the size of each physical construction component might not be the same due to fabrication discrepancies and loose tolerances and may need on-site improvisation to fit them in desired locations. One way to resolve the discrepancy is to use sensors to adapt to the robot control (Lundeen et al. 2019; Sharif et al. 2016), but the adapted workspace geometry needs to be updated in the virtual model. On the other hand, the robot DT system developed in this work fulfills the demand for real-time data exchange, which is widely utilized in the manufacturing industry, digital fabrication, and human–robot collaboration assembly (Zhuang et al. 2018; Bilberg and Malik 2019; Malik and Brem 2021). For example, Naboni and Kunic (2019) used DT for complex wood structure manufacturing and assembly. Furthermore, by combining with other techniques such as augmented reality (AR), the synchronization and communication mechanism of the robot DT system can be improved (Cai et al. 2020). The data transmission in these types of methods is from the virtual environment to the physical environment.

3 The digital twin of the robotic construction process

The proposed real-time process-level robot DT system consists of three modules: the physical robot module, the virtual robot module, and the communication module, as shown in Fig. 1. First, the virtual robot module includes the DT for visualizing the robot and the motion planner for planning the trajectory and solving the inverse kinematics (IK) problems. We create a robotic arm model representing the physical robotic arm in the Gazebo simulation environment (Koenig and Howard 2004). Various robot motion planning methods are implemented in the DT system to control the physical robotic arm, including joint angle control and Cartesian path planning. Second, the physical robot module includes the physical robotic arm and the embedded sensors for measuring joint angles. Third, the bi-directional communication module includes two different communication protocols [message queuing telemetry transport (MQTT) (Light 2017) and automation device specification (ADS) (Beckhoff and Beckhoff 2021)] for data exchange and synchronization. The MQTT is a common industrial communication protocol that provides the generalization of the proposed DT system. The ADS is specific for data exchange in the Beckhoff associated system, which provides high performance in the system. Finally, the algorithm for checking the synchronization between the physical robotic arm and the virtual twin is also developed.

The system is developed in the robot operating system (ROS), since it is a meta-operating system that provides a message exchange mechanism between platforms across a network (Quigley et al. 2009). For instance, the motion planner in the virtual robot module plans a trajectory and then sends the control commands to the DT robot for execution and visualization. Each platform can be operated under different operating systems or programming languages. ROS and the Gazebo simulator have been utilized as modeling and operating tools for robotic buildings and environments (Linner et al. 2011) or multi-robot collaboration across different robot platforms (Vasey et al. 2020).

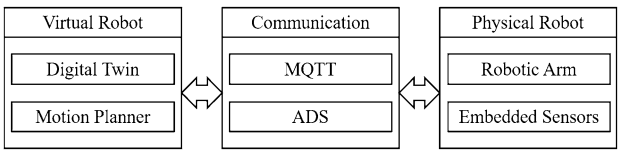

Fig. 1 The framework of the online process-level robot digital twin system

Figure 2 shows the flowchart of the data exchange between each platform in the MQTT version and the ADS version of the proposed DT. On the left side is the virtual robot module, on the right side is the physical robot, and in the middle is the communication module. The system requires at least one personal computer (PC) to run the DT system and one PC embedded on the robot to process the commands and send them to the controller to control the robot. The two PCs are connected with ethernet (blue line) for data exchange and communication using the MQTT or ADS protocol. The circle in the figure represents different ROS nodes that process and exchange data. A detailed description of each module is provided in the following subsections.

3.1 Virtual robot module

We use the Gazebo simulator and the rviz 3D visualizer to develop the DT in the virtual robot module on a Linux PC (Koenig and Howard 2004; Kam et al. 2015). The Gazebo is a real-world physics simulator that creates a world and simulates the robot, whereas the rviz is visualization software that can read and display the data from Gazebo or real-world sensors. The robotic arm model is imported to the Gazebo and rviz using the unified robot description format (URDF), as shown in Fig. 3. Two different robots are used as examples in the DT, i.e., KUKA KR5 and KUKA KR120 robotic arms. The joint angles of the robotic arms are exchanged between the two programs to ensure synchronization.

To plan the specific construction task or motion, a motion planner is required in the module. Either MATLAB or the MoveIt! motion planning framework can be used as the motion planner to achieve the task (Coleman et al. 2014). The Robotic System Toolbox in MATLAB can plan the trajectory and solve the inverse kinematics problems encountered by the robot. The built-in functions in MATLAB provide a faster programming ability to control the robot in various ways. However, it suffers from a latency issue and is not fast enough for real-time planning purposes.

On the other hand, the MoveIt! is a motion planning package for ROS, which plans the motion inside rviz and sends it to Gazebo. Figure 3a shows the interface of the MoveIt! motion planning in rviz. The goal state, velocity, and time parameters can be customized and determined by the user as input to the motion planner. The result of the motion planning will then be demonstrated in rviz and sent back to Gazebo for execution. Both MATLAB and MoveIt! can be run on the same Linux PC as the DT or run on a different PC and connected through the network.

To allow the robot to perform different construction tasks, we include two control modes in the DT: joint state control mode and Cartesian path control mode. In the joint state control mode, the user can determine the target joint angles of the robot and let the MoveIt! package plan the trajectory starting from the current robot joint states. This is an intuitive way for the user to control the robot to the desired pose. In the Cartesian path control mode, the user can specify a list of waypoints and let the robot end-effector follow the trajectory. The MoveIt! package will calculate the robot joint

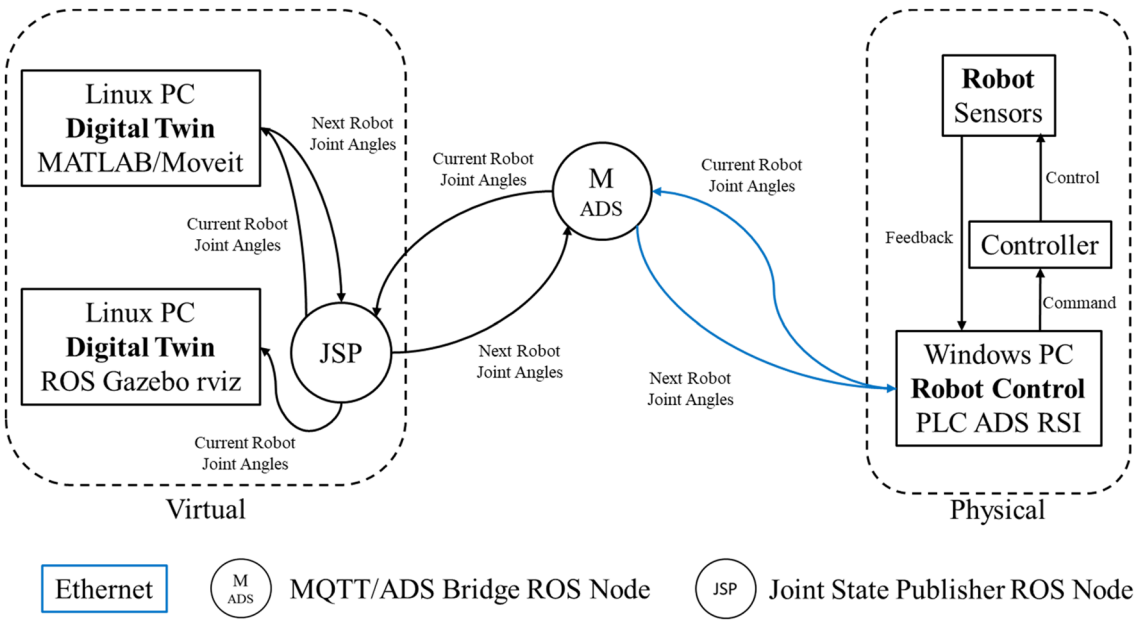

Fig. 2 The flowchart of the data exchange between each platform

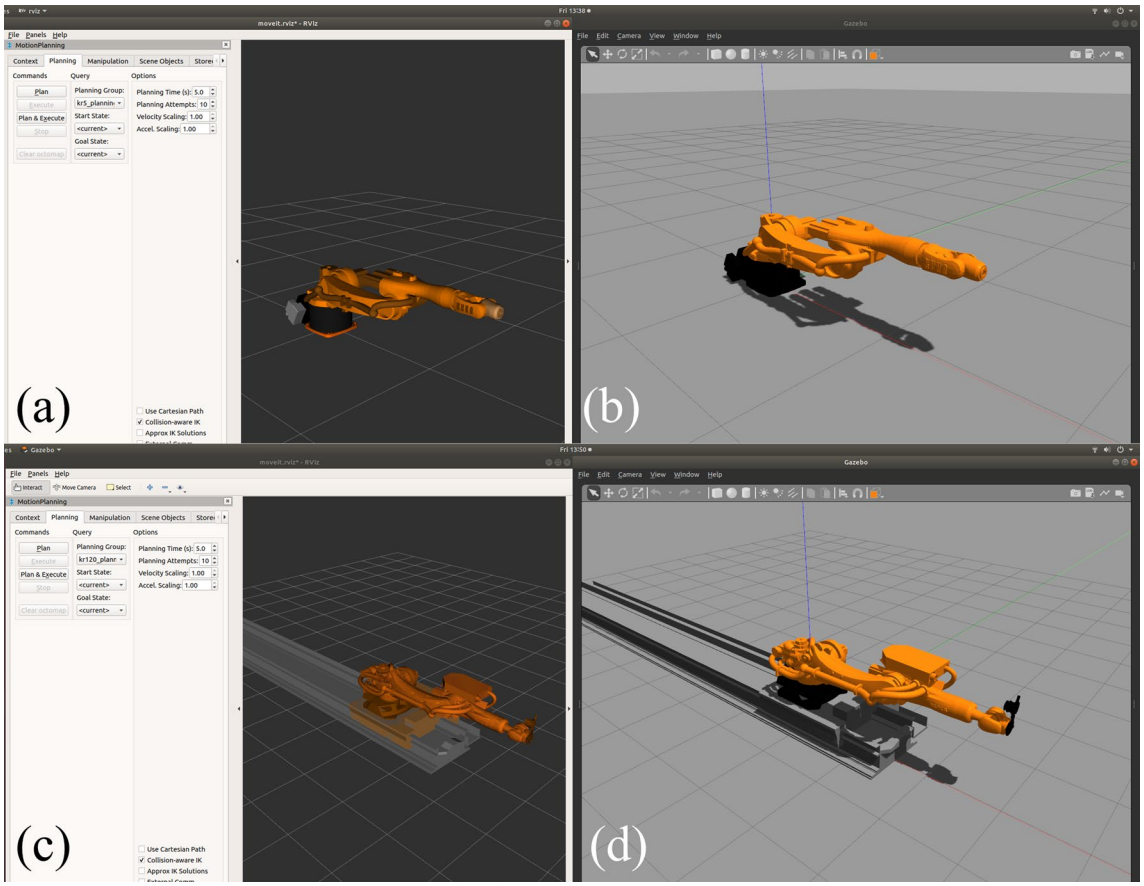

Fig. 3 The KUKA KR5 robotic arm in **a** rviz with MoveIt! package and **b** Gazebo. The KUKA KR120 robotic arm in **c** rviz with MoveIt! package and **d** Gazebo

angles using inverse kinematics and control the robot to execute the plan. For example, the user extracts the waypoints from a BIM model geometry for a 3D-printing robotic arm to determine the work plan.

For the data exchange, only the current robot joint angles and the next robot joint angles are displayed within the virtual robot module. Both Gazebo and rviz read the current robot joint angles to visualize the robot state. The MATLAB or MoveIt! package read the robot joint angles, determine the next robot joint angles, and send them back to Gazebo and rviz for execution. The joint state publisher (JSP) is the ROS node for publishing the current robot state to different ROS nodes, including the current robot joint angles from the physical robot module.

3.2 Physical robot module

In the proposed process-level DT system, the KUKA KR120 robotic arm is the physical robot, as shown in Fig. 4. The KUKA KR120 robotic arm is a 6DOF robot with an additional external degree-of-freedom for the track system. The central control and communication node consists of a software PLC (Beckhoff and Beckhoff 2021) running on an

Fig. 4 The KUKA KR120 robotic arm for the physical robot module

embedded PC that communicates over TCP/IP with the DT system. Joint angles are then communicated via a hardware bridge using the EtherCAT protocol to the robot IO, which

is read via the Kuka Robot Sensor Interface (RSI). The embedded encoders on the robotic arm are used to measure the joint angles and written to the EtherCAT bus by RSI. In the ADS communication version, the TwinCAT ADS is also running on the embedded Windows PC to publish and receive the messages.

After activating the robotic arm, the system first records the current robot joint angles as the origin of the robot for robot controlling purposes. Once the physical robot receives the next joint angles from the virtual robot, it will calculate the differences of the joint angles and then uses the recorded origin to control the robotic arm in the relative mode. The robot control command and the sensor measurement are two data exchanges inside the physical robot module, as shown on the right side of Fig. 2.

Due to the limitation of the hardware data transmission speed and the missing data issue, some jitter effects might occur on the physical robot. To resolve this issue, we use two different methods in the MQTT communication and the ADS communication. In the first method, we apply the first-order delay filter in the TwinCAT PLC to smooth the robot trajectory. If a situation where missing data might arise, the delay filter can still interpolate and smooth the robot trajectory and avoid the jitter effects or sudden movements. In the second method, we apply the TwinCAT computer numerical control (CNC) package to generate the physical robot motion. The CNC package can plan and interpolate the received waypoints to control the robot while respecting all dynamic limits and singularities of the robot. The density of the robot waypoints can be very high (up to 10×higher than in a normal robot program) while maintaining look ahead and synchronization of all robot and external axes.

3.3 Communication module

Finally, the communication module links the virtual robot module and the physical robot module. We develop two different communication protocols, i.e., MQTT communication protocol and TwinCAT ADS communication protocol, for data exchange between the ROS system in the virtual robot module and the PLC in the physical robot module. Both the MQTT communication protocol and the TwinCAT ADS communication protocol are capable of near real-time communication (4 ms update rate) and thus are suitable for smooth robotic control. First, we develop an MQTT Bridge ROS node (M) to connect the MQTT to the ROS system, as shown in the middle of Fig. 2. The MQTT Bridge node is run on the same Linux PC as the DT system to exchange the joint angles with the JSP node and connect with PLC in the physical robot module through the ethernet cable. The data exchange frequency in the MQTT Bridge ROS node is set to be 250 Hz to match the update rate of the Kuka RSI software.

The joint angles of the robotic arm and the location of the track system are the primary data streams exchanged in the MQTT bridge ROS node. Figure 5 illustrates the data structure and exchange process in the MQTT bridge ROS node. The data stream concatenates the robot joint angles from A1 to A6 and the track location E1 joint with a plus-minus sign and comma. Each joint angle is rounded to three decimal places (E1 joint is rounded to four decimal places) and pads zeros to the left. Thus, the length of the data is consistent and easily retrieved by the PLC. The data streams have to be converted between ROS topic, Python string, and MQTT string formats to process the data correctly.

After receiving the joint angles data from the virtual robot module through the ROS topic, the system first converts the data to python string for easy storage and access. Next, the data are converted to the MQTT string type and sent to the physical robot module. This process can also avoid the corrupted text issue when directly converting from the ROS topic to the MQTT string type. The data stream from the physical robot module is also processed with the same procedure and data structure and sent to the virtual robot module.

Second, we develop a TwinCAT ADS Bridge ROS node (ADS) to connect the TwinCAT ADS to the ROS system, as shown in the middle of Fig. 2. Similar to the MQTT Bridge node, the TwinCAT ADS Bridge node is also run on the same Linux PC as the DT system to exchange the joint angles with the JSP node and connect with TwinCAT ADS and PLC in the embedded PC through the ethernet cable. The data exchange frequency in the TwinCAT ADS Bridge node is set to be 1,000 Hz to ensure the transmission speed on the robotic arm. The joint angles of the robotic arm and the location of the track system are stored

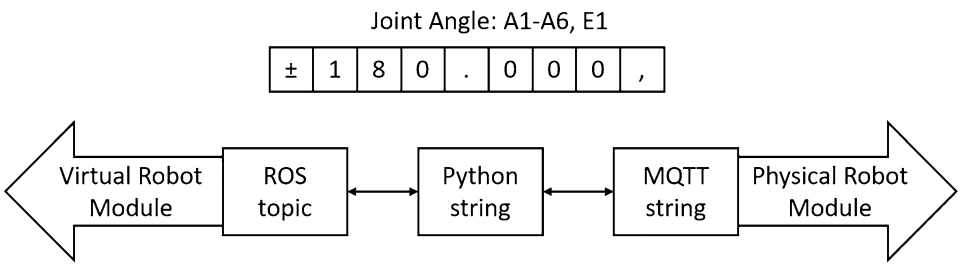

Fig. 5 The data structure and exchange in the MQTT Bridge ROS node

in an array and directly sent between the ADS and the ROS system. Both ADS and the ROS system can read and change the array data to reflect the work plan and the robot condition.

When exchanging the data between the virtual robot module and the physical robot module, the system must ensure the control commands are executed completely, and the pose of the physical and virtual robot is synchronized. We develop a robot pose checking algorithm to confirm the synchronization between the two robotic arms. Algorithm 1 shows the pseudo-code of the pose checking algorithm (PCA). The algorithm takes the current virtual robot pose θ_{virtual}, current physical robot pose θ_{physical}, and the next robot pose θ_{next} as input.

Algorithm 1: Pose Checking Algorithm

```
procedure Next Pose(θ_virtual, θ_physical, θ_next)
    diff(θ) ← |θ_virtual − θ_physical|
    if diff(θ) > threshold then
        θ_next ← θ_virtual
        Re-plan the trajectory based on θ_next
    else
        θ_next ← θ_next
    end if
    return θ_next
end procedure
```

First, the PCA calculates the difference between θ_{virtual} and θ_{physical}. If the difference exceeds the pre-defined threshold, the next joint angles θ_{next} will be assigned with the current joint angles θ_{virtual} to ensure the physical robot can reach the desired joint angles. The trajectory also needs to be re-planned to reflect the new current joint angles. On the other hand, if the difference does not exceed the threshold, the robot will simply execute the next joint angles.

4 Experiments and results

The real-time process-level robot DT system is implemented and deployed in the Digital Fabrication Laboratory at the Taubman College of Architecture and Urban Planning and the Civil Engineering Robotics Laboratory at the College of Engineering. Three KUKA KR120 robotic arms are the target physical robots, as shown in Fig. 4.

4.1 Transmission time experiment and result

To evaluate the proposed system, we conducted experiments to verify the transmission time between the ROS system and PLC, and confirm that the pose between the physical robot and its DT are synchronized during trajectory execution. In the first experiment, we set up a local network between two computers and built the MQTT communication protocol and the TwinCAT ADS communication protocol to test the ROS Bridge node. Figure 6 shows the first experimental setup and the data exchange. The robot motion is planned on the 1st computer, and the trajectory is sent to the 2nd computer through the MQTT or ADS for execution. The Cartesian path control mode is applied to plan four different motions, i.e., x-axis motion, y-axis motion, z-axis motion, and triangle motion. During the experiment, the timer is triggered when the pose from the 1st computer is sent and stopped when the corresponding pose from the 2nd computer is received to record the data transmission time.

For the transmission time between the virtual and the physical robot, we replicated the above experiment and recorded the time when the corresponding pose data from the physical robot was received. The robot motion is planned in the virtual robot module, and the pose is sent to the physical robot. The physical robot will then send the pose data

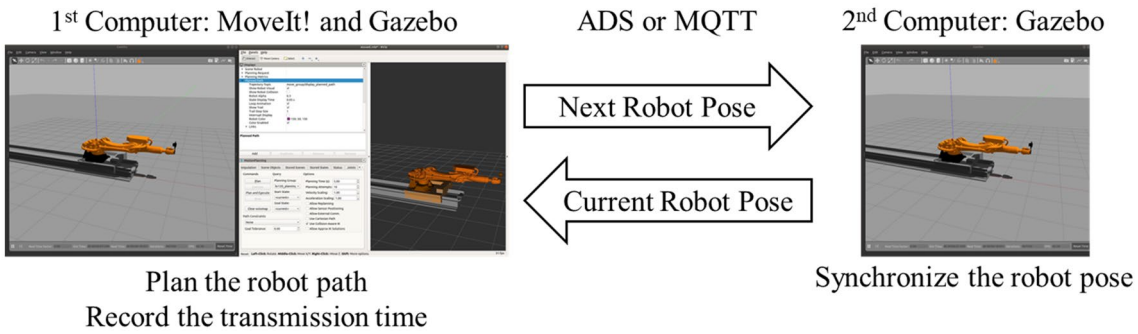

Fig. 6 Two virtual robots are used to evaluate the MQTT and ADS data transmission time

Table 1 Data transmission time between two robots using the MQTT and ADS communication

	Average time (ms)	Maximum time (ms)	Minimum time (ms)
MQTT (VtoV)	8.786	9.024	8.141
ADS (VtoV)	5.173	5.905	4.237
MQTT (VtoP)	12.547	15.771	11.754
ADS (VtoP)	9.483	12.688	9.095

VtoV virtual to virtual robot, *VtoP* virtual to physical robot

back to the virtual robot to record the transmission time. In this experiment, the pose is not executed in the physical robot to avoid the first-order delay filter effect and only focuses on the data transmission time.

Table 1 shows the result of the data transmission time experiment. MQTT (VtoV), ADS (VtoV), MQTT (VtoP), and ADS (VtoP) are four different settings, where VtoV represents Virtual robot to Virtual robot and VtoP represents Virtual robot to Physical robot. We executed four trajectories (x-axis, y-axis, z-axis, and triangle motion) 100 times and collected 400 data points for two VtoV settings. For two VtoP settings, we executed four trajectories four times and collected 16 data points. The average data transmission time for MQTT (VtoV) is 8.786 ms, and for ADS (VtoV) is 5.173 ms due to the transmission frequency limitation of the MQTT communication protocol (250 Hz). For MQTT (VtoP), the average data transmission time is 12.547 ms. Finally, ADS (VtoP) was found to have an average of 9.483 ms of data transmission time to exchange data between the DT and the physical robot to execute the work plan. The results of the MQTT version were higher than the ADS version because of the additional data conversion steps and the limitation of the MQTT frequency.

4.2 MATLAB joint angle control mode experiment and result

In the second experiment, the physical robot execution accuracy is evaluated using two different communication protocols. We use MATLAB to plan the robot trajectory using the joint angle control mode in the DT and send the work plan through the MQTT or TwinCAT ADS communication protocol to the physical robot. Figure 7 shows the procedure of the second process-level robot DT system experiment. In both MQTT and ADS versions, we develop the first-order delay filter on the physical robot with a 20 ms delay to resolve the jittering effect instead of using the CNC package to compare the physical robot execution trajectory under the same condition.

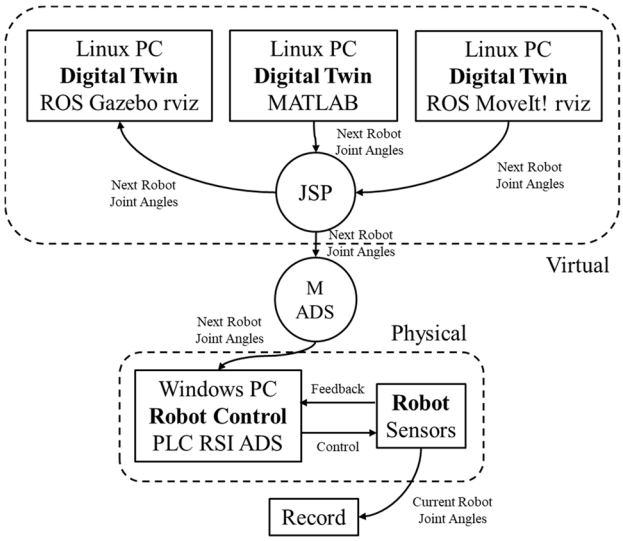

Fig. 7 The procedure of the MATLAB and MoveIt! planned robot process-level digital twin system experiment using the MQTT communication protocol and TwinCAT ADS communication protocol

Once the reaching task trajectory is planned and executed in MATLAB and Gazebo DT, the joint angles are sent to the physical robot using the MQTT or ADS communication. The robot poses and control commands are generated by the Inverse Kinematics package in MATLAB. We use the stationary robotic arm in this experiment, i.e., excluding the track system (E1) joint and the embedded encoders on the physical KUKA robotic arm to measure the joint angles of the physical robot during the execution. The pose data are sent back to the virtual robot module PC to record the executed trajectory. The differences between the planned and the executed trajectory are compared.

Figure 8 shows the results of the MQTT communicated virtual and physical robot joint angles using the MATLAB planned reaching trajectory. Each line represents the angle of each joint (A1, A2, A3, A4, A5, and A6) in radians. The trajectory from the virtual robot consists of 1500 waypoints, and the measurement from the physical robot includes 18,802 data points. The reason for the higher data points from the physical robot is that the data acquisition rate in ROS (approximately 1000 Hz) is higher than the rate in MATLAB (approximately 100 Hz). The results showed that the line of each joint angle had the same trend in the two robots, which demonstrated the consistency of the synchronization between the two robots using the MQTT communication protocol.

On the other hand, the ADS version is also evaluated using the same procedure. Figure 9 shows the results of the ADS communicated virtual and physical robot joint angles using the same MATLAB planned reaching trajectory. The trajectory from the virtual robot consists of 1500 waypoints,

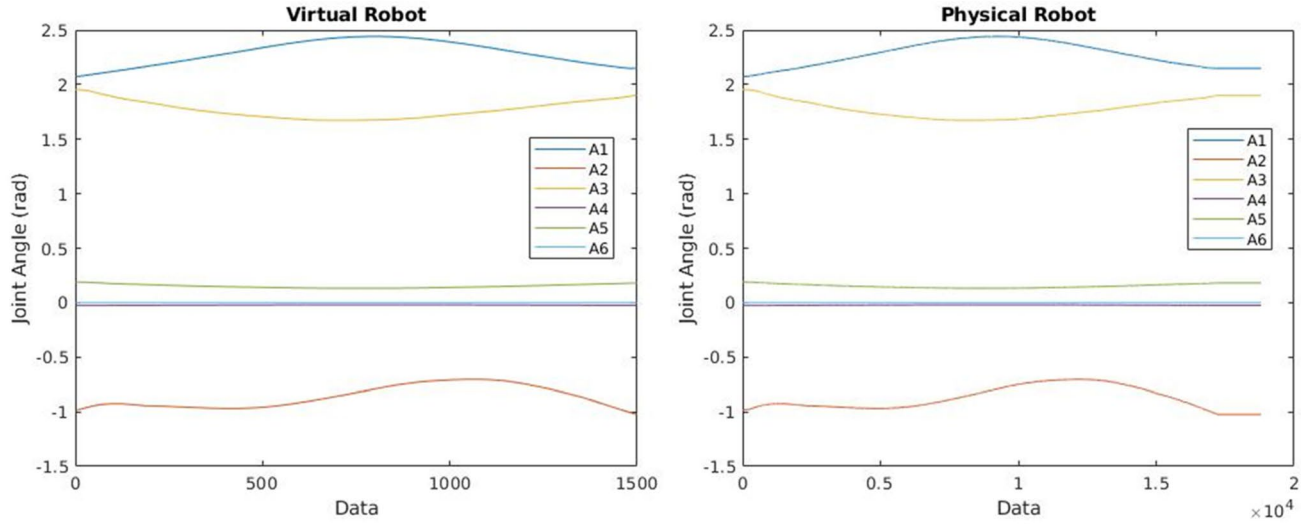

Fig. 8 The results of the MQTT communicated virtual and physical robot joint angles using the MATLAB planned reaching trajectory

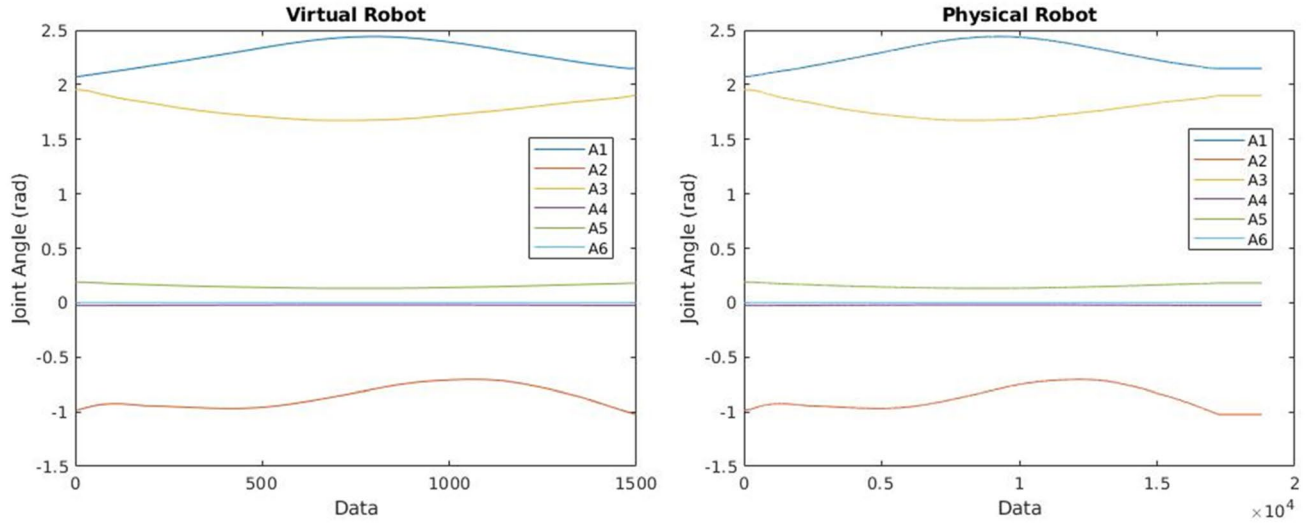

Fig. 9 The results of the ADS communicated virtual and physical robot joint angles using the MATLAB planned reaching trajectory

and the measurement from the physical robot includes 17,145 data points. The results showed that the ADS communication could also synchronize the joint angles between two robots successfully.

To further evaluate the synchronization accuracy, we calculated the average error and the maximum error of each joint angle between the two robots. We first align the virtual robot and physical robot results by interpolation to obtain the same number of data points from the two robots to calculate the mean square error. Table 2 lists the results of the average and maximum joint angle error using the MATLAB planned trajectory in the MQTT and ADS communication. In the MQTT communication, the average errors of each joint angle are less than 0.0013 in radians, and the maximum errors of each joint angle are less than 0.0025 in radians. In the ADS communication, the average errors of each joint angle are less than 0.0012 in radians, and the maximum errors of each joint angle are less than 0.003 in radians. Due to the robot system specification and decimal conversion in the physical robot, the errors under 0.001 are neglectable. These results indicated that the synchronization of the virtual and the physical robot demonstrated high accuracy using both communication

Table 2 The average and the maximum joint angle errors (rad) between the virtual and the physical robot using the MATLAB planned trajectory in the MQTT and ADS communication

Joint (rad)	MQTT		ADS	
	Average error	Maximum error	Average error	Maximum error
A1	0.00024	0.00056	0.00034	0.00136
A2	0.00040	0.00076	0.00032	0.00073
A3	0.00077	0.00149	0.00077	0.00148
A4	0.00127	0.00241	0.00120	0.00293
A5	0.00030	0.00068	0.00037	0.00068
A6	3.535e-05	7.623e-05	4.345e-05	0.00017

methods. The proposed pose checking algorithm (PCA) also helps minimize the discrepancy between two robots during the data transmission.

4.3 MoveIt! joint control mode experiment and result

In the third experiment, we used the MoveIt! package to plan the robot trajectory and compare the accuracy of the trajectory execution between the two communication methods. Figure 7 shows the procedure of the MoveIt! planned process-level robot DT system experiment using the MQTT or ADS communication method. The joint angle control mode is evaluated in this experiment. Ten different goal joint angles are randomly generated, and the trajectories are planned using the MoveIt! package. This information is then executed in the Gazebo DT and sent to the physical robot. Finally, the joint angles of the physical robotic arm are measured and recorded to compare with the virtual robotic arm.

Figures 10 and 11 show the results of the MQTT and ADS communicated virtual and physical robot joint angles using the MoveIt! joint angle control mode planned trajectory. The trajectory from the virtual robot consists of 10

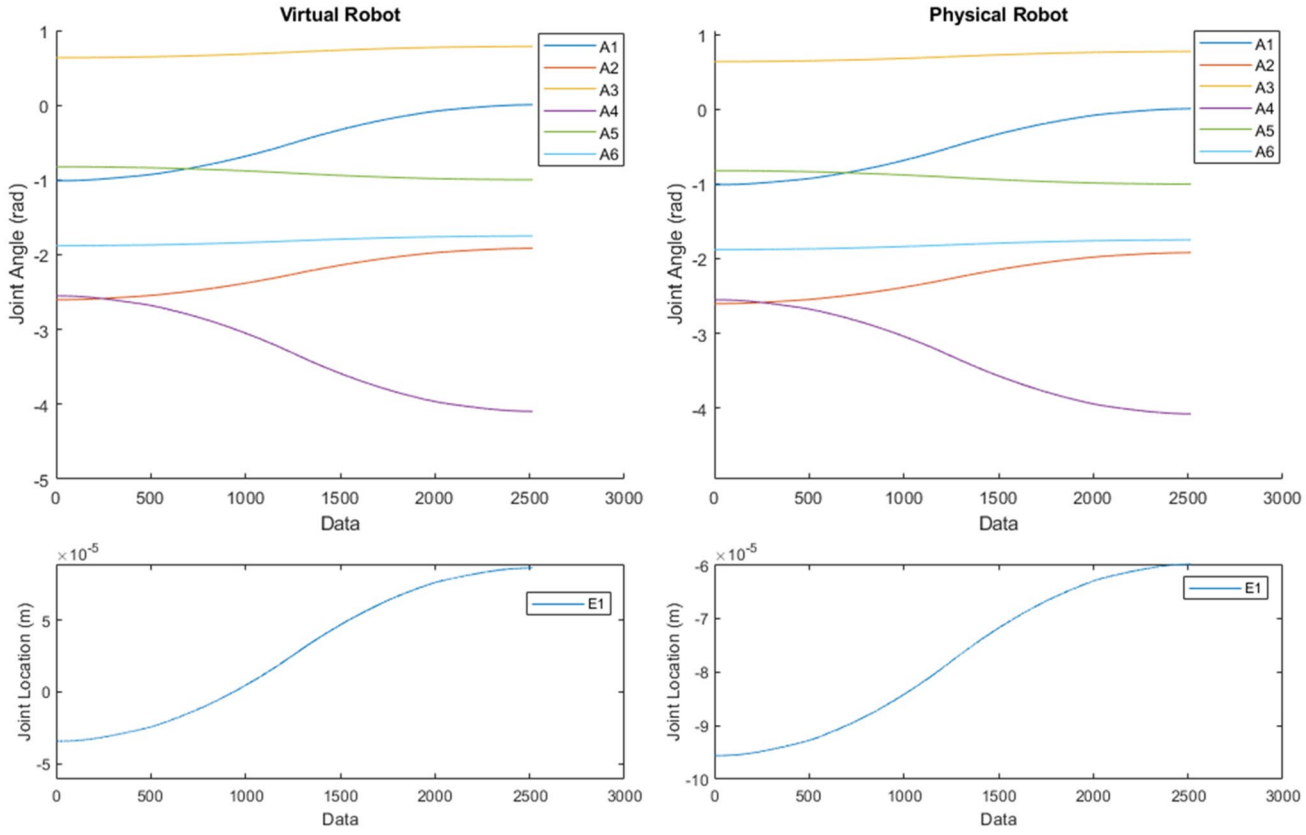

Fig. 10 The results of the MQTT communicated virtual and physical robot joint angles using the MoveIt! joint angle control mode planned trajectory

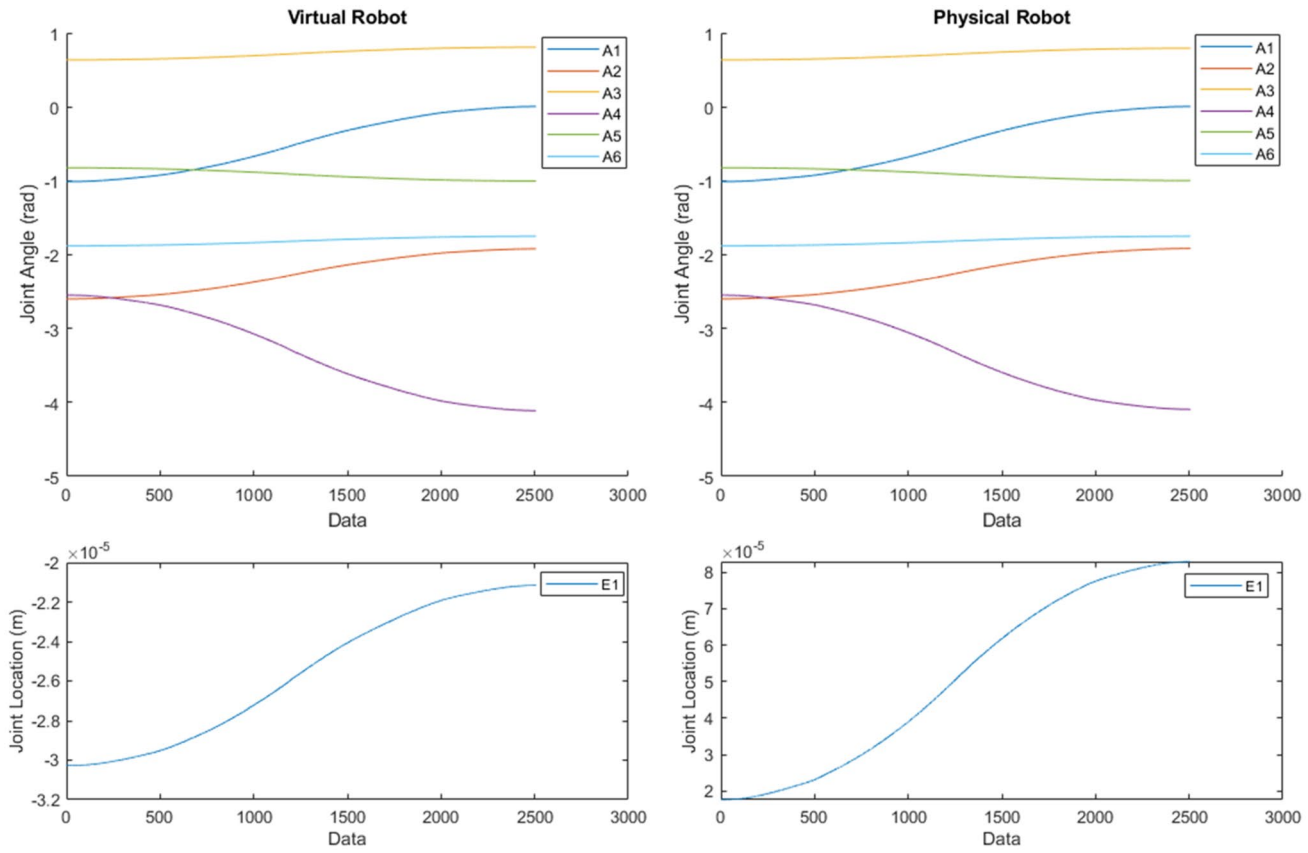

Fig. 11 The results of the ADS communicated virtual and physical robot joint angles using the MoveIt! joint angle control mode planned trajectory

Table 3 The average and maximum joint errors (rad and m) between the virtual and the physical robot using the MoveIt! joint angle control mode in the MQTT and ADS communication

Joint (rad)	MQTT		ADS	
	Average error	Maximum error	Average error	Maximum error
A1	0.00239	0.00549	0.00272	0.00711
A2	9.117e-05	0.00024	0.00061	0.00099
A3	0.00072	0.00193	0.00088	0.00269
A4	0.00434	0.01143	0.00601	0.01600
A5	0.09804	0.19458	0.09887	0.19594
A6	0.00162	0.00427	0.00190	0.00565
E1 (m)	0.00410	0.00108	0.00048	0.00144

random goal poses, which include 2507 waypoints, and the measurement from the physical robot includes 2513 data points. The trajectory patterns of the virtual and the physical robot are similar and only have minor errors. Table 3 shows the results of the average and maximum joint angle error using the MoveIt! joint angle control mode in the MQTT and ADS communication. In the MQTT communication, the average errors of each joint angle are less than 0.099 in radians and less than 0.0042 in m for the E1 joint. The maximum errors of each joint angle are less than 0.195 in radians and less than 0.0011 in m for the E1 joint. The A5 joint has the highest error in the MQTT experiment.

In the ADS communication, the average errors of each joint angle are less than 0.099 in radians and 0.0005 in m for the E1 joint. The maximum errors of each joint angle are less than 0.196 in radians and 0.0015 in m for the E1 joint. The A5 joint also has the highest error in the ADS experiment, and the average error of the E1 joint in ADS

is smaller than the MQTT E1 joint. The results showed that the joint angle control mode in the MoveIt! package can precisely control the robot to the goal pose with some minor errors due to the first-order delay filter and synchronize with the physical robot by using the proposed PCA algorithm to ensure the accuracy of the joint angles.

4.4 Cartesian path control mode experiment and result

In the final experiment, we used the MoveIt! package to plan the trajectory using the Cartesian path control mode to evaluate the physical robot execution accuracy. Four different sets of end-effector trajectories were prepared (*x*-axis motion, *y*-axis motion, *z*-axis motion, and triangle motion), executed in the Gazebo DT, and sent to the physical robot. Both MQTT and ADS communication methods are used for data exchange. The pose of the physical robotic arm end-effector is measured and recorded to compare with the planned trajectories and the pose of the end-effector of the virtual robotic arm. Each trajectory was repeated two times, and eight sets of data points were collected for evaluation.

Figure 12 shows the results of the MQTT and ADS communicated virtual and physical robot end-effector pose using the MoveIt! Cartesian path control mode planned trajectory. The solid blue line represents the planned trajectory in the virtual robot module, the red dashed line represents the MQTT executed trajectory, and the yellow dotted line represents the ADS executed trajectory. Each line represents the position of the robot end-effector in world coordinates (*X, Y, Z*). The results showed that both the MQTT and ADS trajectories matched the virtual robot trajectory with some minor errors due to the first-order delay filter.

In addition, to further evaluate the performance, the average and maximum errors of the robot end-effector pose are also calculated, as listed in Table 4. The average errors of the robot end-effector are 1.422 mm on the *x*-axis, 5.015 mm on the *y*-axis, 1.967 mm on the *z*-axis, and overall 6.487 mm

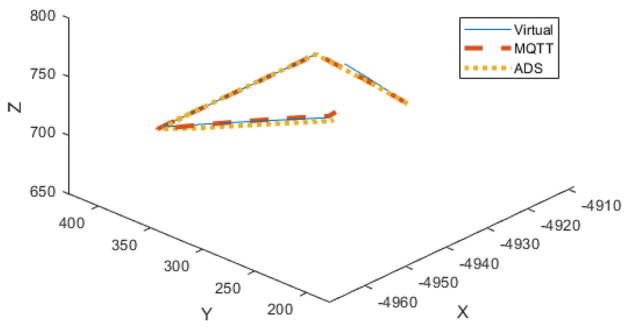

Fig. 12 The results of the MQTT and ADS communicated virtual and physical robot end-effector pose using the MoveIt! Cartesian path control mode planned trajectory

Table 4 The average and maximum end-effector pose errors (mm) between the virtual and the physical robot using the MoveIt! Cartesian path control mode in the MQTT and ADS communication

(mm)	MQTT		ADS	
	Average error	Maximum error	Average error	Maximum error
X	1.422	7.787	1.543	8.027
Y	5.015	14.345	3.667	7.695
Z	1.967	7.204	1.842	6.854
Overall	6.487	16.052	5.285	10.314

for the MQTT communication protocol, where the maximum errors are 7.787 mm on the *x*-axis, 14.345 mm on the *y*-axis, 7.204 mm on the *z*-axis, and overall 16.052 mm. For the ADS communication protocol, the average errors are 1.543 mm on the *x*-axis, 3.667 mm on the *y*-axis, 1.842 mm on the *z*-axis, and overall 5.284 mm. The maximum errors are 8.027 mm on the *x*-axis, 7.695 mm on the *y*-axis, 6.854 mm on the *z*-axis, and overall 10.314 mm.

5 Discussion

The difference in the transmission time between the MQTT communication and the ADS communication is less than 5 ms. The ADS communication can directly modify the joint angle variable in the TwinCAT system, and the MQTT communication has to convert joint angle data to several different formats and thus requires extra time to process the data. For the accuracy of the physical robot execution, both the MQTT and ADS communication achieve similar performance, and the errors of the joint angle are within 0.1 radians. With additional modifications, the CNC package can also provide the opportunity to connect to both the MQTT and ADS communication protocols and precisely control the physical robot.

The first-order delay filter causes some minor errors since the joint angle data received by TwinCAT are slightly different each time, and the smoothed trajectories are different. The errors are reduced by the proposed pose checking algorithm (PCA). Both MATLAB and the MoveIt! package achieve similar performance on planning the robot motion by joint angle control mode and Cartesian path mode and communicating with the communication module. However, the MATLAB package has limited the data communication frequency (up to 100 Hz) and requires a code generation function to improve the execution speed.

The advantages of using the proposed bi-directional communication protocol to synchronize the virtual and real robotic arms are to ensure that the collaborating human worker is informed of the work plan of the robot, to

supervise and review the robot's work plan, and to control the robot accurately. Existing DT systems focus on the analysis and optimization of industrial and construction tasks such as assembly. Welding, or asset monitoring (Mertens et al. 2020; Tabar et al. 2020; Malik and Brem 2021; Lu et al. 2020b). With the proposed DT communication method, the existing DT systems can benefit from high accurate synchronization. On the other hand, the use of the ROS framework also has the advantage of adapting to various robot systems or controlling software. With the proposed framework, the DT system is not restricted to specific hardware or software and can easily exchange data across different platforms. For example, instead of using MATLAB or the MoveIt! package, we can use modeling software such as Rhino to directly extract the trajectory from components and send it to the communication module to exchange with the physical robot module.

There are also some limitations in the proposed DT system that need to be addressed in future work. First, the pose of the physical robot is measured only by the onboard sensors. By applying additional sensors, such as cameras (Liang et al. 2018, 2019a, b), to supplement the pose estimation and to fuse that data with the onboard sensor data, the accuracy of the robot pose measurement can be improved. Second, some of the limitations of the physical robot are not reflected in the virtual robot. For example, the velocity and the acceleration limits of the robot joints are not incorporated into the path planning correctly in the first-order delay filter version, and the physical robot will stop due to the sudden high acceleration. This also constraints the proposed DT system to the position-controlled mode. To resolve this issue, the dynamic limits can be reflected in the TwinCAT CNC package to control the physical robot and further expand it to force-controlled robots.

Third, the detailed information of the surrounding environment, e.g., obstacles in the workspace, is not included in the proposed DT system. When planning the robot trajectory, those obstacles need to be considered to avoid any unintended collisions. The environmental objects can be collected and modeled into the DT system by sensing technology and model registration methods, which also encumbers the restrictions inherent in real-time modeling. For example, soft and deformable materials such as soil and fresh concrete are difficult to integrate into the DT system and manipulate using the robot.

6 Conclusions and future work

This paper presented the development of a real-time process-level robot DT system for human–robot collaboration in construction and digital fabrication. The system includes the virtual robot module, the physical robot module, and the communication module. We leveraged the ROS Gazebo and rviz components to develop the virtual robot module, i.e., the DT of the physical robot, and connect to the physical robot module through the MQTT Bridge or TwinCAT ADS Bridge in the communication module. The joint angles of the robotic arm are exchanged and synchronized between the two robots. We also utilized MATLAB or the MoveIt! package to plan and control the robotic arm in the virtual robot module, and then send the commands to the physical robot module for execution. In addition, we implemented two different control modes, i.e., joint angle control mode and Cartesian path control mode, in the MoveIt! program to control the virtual robot by joint angles or end-effector pose. Finally, we developed a pose checking algorithm (PCA) to ensure that the poses of the two robots were synchronized.

The system was implemented and deployed on a KUKA KR120 robotic arm in the Digital Fabrication Laboratory and the Civil Engineering Robotics Laboratory at the University of Michigan to evaluate the synchronization and the data transmission time. Although the system was developed for the specific KUKA robotic arm, it can be easily adapted to other robot models. We evaluated the system by comparing the data transmission time, joint angles, and end-effector pose between the virtual and physical robot using several planned trajectories and calculated the average and maximum mean square errors. The results showed that the proposed real-time process-level robot DT system can plan the robot trajectory inside the virtual environment and execute it in the physical environment with high accuracy and real-time performance. The human worker can perceive the robot work plan in advance and provide instructions to the robot in the DT system, thereby improving the safety of the human–robot collaboration. In addition, the high accuracy and real-time performance of the DT system can ensure the information displayed to human workers is accurate and thus increase the trust level between human workers and robots.

In future work, we plan to design an improved user interface for displaying the information of the physical robot in the DT and enabling human workers to collaborate with robots intuitively. We are also developing a robot planning mechanism that would enable the robot to first demonstrate the planned trajectory inside the DT before its execution by the physical counterpart (Wang et al. 2021). The human worker can thus anticipate the movement of the robot in advance and approve the task. On the other hand, we will integrate the proposed DT system with BIM to synchronize the digital models. The BIM model can automatically populate the DT system to create the workspace geometry. After the physical robot executes the work plan, the current workspace geometry information will be sent back to the BIM model to reflect the evolving changes.

Funding The work presented in this paper was supported financially by United States National Science Foundation Awards (#2025805 and #2128623). Any opinions, findings, and conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the United States National Science Foundation.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

Aheleroff S, Polzer J, Huang H, Zhu Z, Tomzik D, Lu Y, Lin Y, Xu X (2020) Smart manufacturing based on digital twin technologies. In: Machado C, Davim JP (eds) Industry 4.0: challenges, trends, and solutions in management and engineering. CRC Press, Boca Raton, p 77. https://doi.org/10.1201/9781351132992-3

Al-Sehrawy R, Kumar B (2020) Digital twins in architecture, engineering, construction and operations. a brief review and analysis. In: Toledo Santos E, Scheer S (eds) Proceedings of the international conference on computing in civil and building engineering (ICCCBE). Springer International Publishing, São Paulo, Brazil (Online), 2020, pp 924–939. https://doi.org/10.1007/978-3-030-51295-8_64

Beckhoff, TwinCAT ADS, Beckhoff (2021) https://github.com/Beckhoff/ADS. Accessed 6 Feb 2021

Bilberg A, Malik AA (2019) Digital twin driven human–robot collaborative assembly. CIRP Ann 68:499–502. https://doi.org/10.1016/j.cirp.2019.04.011

Bosché F, Ahmed M, Turkan Y, Haas CT, Haas R (2015) The value of integrating Scan-to-BIM and Scan-vs-BIM techniques for construction monitoring using laser scanning and BIM: the case of cylindrical MEP components. Autom Constr 49:201–213. https://doi.org/10.1016/j.autcon.2014.05.014

Bruckmann T, Mattern H, Spenglerc A, Reichert C, Malkwitz A, König M (2016) Automated construction of masonry buildings using cable-driven parallel robots. In: Proceedings of the international symposium on automation and robotics in construction (ISARC), IAARC, Auburn, AL, USA, 2016, pp 332–340. https://doi.org/10.22260/ISARC2016/0041

Cai Y, Wang Y, Burnett M (2020) Using augmented reality to build digital twin for reconfigurable additive manufacturing system. J Manuf Syst. https://doi.org/10.1016/j.jmsy.2020.04.005 (**in press**)

Coleman DT, Sucan IA, Chitta S, Correll N (2014) Reducing the barrier to entry of complex robotic software: a MoveIt! case study. J Softw Eng Robot 5:3–16. https://doi.org/10.6092/JOSER_2014_05_01_p3

Colledani M, Terkaj W, Tolio T (2009) Product-process-system information formalization. In: Tolio T (ed) Design of flexible production systems: methodologies and tools. Springer, Berlin, pp 63–86. https://doi.org/10.1007/978-3-540-85414-2_4

Delbrügger T, Lenz LT, Losch D, Roßmann J (2017) A navigation framework for digital twins of factories based on building information modeling. In: Proceedings of the IEEE international conference on emerging technologies and factory automation (ETFA), IEEE, Limassol, Cyprus, 2017, pp 1–4. https://doi.org/10.1109/ETFA.2017.8247712

Dimitrov A, Golparvar-Fard M (2015) Segmentation of building point cloud models including detailed architectural/structural features and MEP systems. Autom Constr 51:32–45. https://doi.org/10.1016/j.autcon.2014.12.015

Eadie R, Browne M, Odeyinka H, McKeown C, McNiff S (2013) BIM implementation throughout the UK construction project lifecycle: An analysis. Autom Constr 36:145–151. https://doi.org/10.1016/j.autcon.2013.09.001

Eversmann P, Gramazio F, Kohler M (2017) Robotic prefabrication of timber structures: towards automated large-scale spatial assembly. Constr Robot 1:49–60. https://doi.org/10.1007/s41693-017-0006-2

Farrell RG, Lenchner J, Kephjart JO, Webb AM, Muller MJ, Erikson TD, Melville DO, Bellamy RKE, Gruen DM, Connell JH, Soroker D, Aaron A, Trewin SM, Ashoori M, Ellis JB, Gaucher BP, Gil D (2016) Symbiotic cognitive computing. AI Mag 37:81–93. https://doi.org/10.1609/aimag.v37i3.2628

Feng C, Xiao Y, Willette A, McGee W, Kamat VR (2015) Vision guided autonomous robotic assembly and as-built scanning on unstructured construction sites. Autom Constr 59:128–138. https://doi.org/10.1016/j.autcon.2015.06.002

Freedy A, DeVisser E, Weltman G, Coeyman N (2007) Measurement of trust in human-robot collaboration. In: Proceedings of the international symposium on collaborative technologies and systems, IEEE, Orlando, FL, USA, 2007, pp 106–114. https://doi.org/10.1109/CTS.2007.4621745

Hamledari H, McCabe B, Davari S, Shahi A (2017) Automated schedule and progress updating of IFC-based 4D BIMs. J Comput Civ Eng 31:04017012. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000660

Hamledari H, Rezazadeh Azar E, McCabe B (2018) IFC-based development of as-built and as-is BIMs using construction and facility inspection data: site-to-BIM data transfer automation. J Comput Civ Eng 32:04017075. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000727

Hentout A, Aouache M, Maoudj A, Akli I (2019) Human–robot interaction in industrial collaborative robotics: a literature review of the decade 2008–2017. Adv Robot 33:764–799. https://doi.org/10.1080/01691864.2019.1636714

Kam HR, Lee S-H, Park T, Kim C-H (2015) RViz: a toolkit for real domain data visualization. Telecommun Syst 60:337–345. https://doi.org/10.1007/s11235-015-0034-5

Kamat VR, Martinez JC (2005) Large-scale dynamic terrain in three-dimensional construction process visualizations. J Comput Civ Eng 19:160–171. https://doi.org/10.1061/(ASCE)0887-3801(2005)19:2(160)

Kan C, Anumba CJ (2019) Digital twins as the next phase of cyber-physical systems in construction. In: Proceedings of the ASCE international conference on computing in civil engineering (I3CE), ASCE, Atlanta, Georgia, 2019, pp 256–264. https://doi.org/10.1061/9780784482438.033

Koenig N, Howard A (2004) Design and use paradigms for Gazebo, an open-source multi-robot simulator. In: Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS), IEEE, Sendai, Japan, 2004, pp 2149–2154. https://doi.org/10.1109/IROS.2004.1389727

Kopetz H (2011) Real-time systems: design principles for distributed embedded applications. Springer, New York

Liang C-J, Lundeen KM, McGee W, Menassa CC, Lee S, Kamat VR (2018) Stacked hourglass networks for marker-less pose estimation of articulated construction robots. In: Proceedings of the international symposium on automation and robotics in construction (ISARC), IAARC, Berlin, Germany, 2018, pp 859–865. https://doi.org/10.22260/ISARC2018/0120

Liang C-J, Lundeen KM, McGee W, Menassa CC, Lee S, Kamat VR (2019a) Fast dataset collection approach for articulated equipment pose estimation. In: Proceedings of the ASCE international conference on computing in civil engineering (I3CE),

ASCE, Atlanta, GA, USA, 2019a, pp 146–152. https://doi.org/10.1061/9780784482438.019
Liang C-J, Lundeen KM, McGee W, Menassa CC, Lee S, Kamat VR (2019b) A vision-based marker-less pose estimation system for articulated construction robots. Autom Constr 104:80–94. https://doi.org/10.1016/j.autcon.2019.04.004
Liang C-J, Kamat VR, Menassa CC (2020) Teaching robots to perform quasi-repetitive construction tasks through human demonstration. Autom Constr 120:103370. https://doi.org/10.1016/j.autcon.2020.103370
Liang C-J, Wang X, Kamat VR, Menassa CC (2021) Human–robot collaboration in construction: classification and research trends. J Constr Eng Manag 147:03121006. https://doi.org/10.1061/(ASCE)CO.1943-7862.0002154
Liang C-J, Kamat VR, Menassa CC, McGee W (2022) Trajectory-based skill learning for overhead construction robots using generalized cylinders with orientation. J Comput Civ Eng 36:04021036. https://doi.org/10.1061/(ASCE)CP.1943-5487.0001004
Light RA (2017) Mosquitto: server and client implementation of the MQTT protocol. J Open Source Softw 2:265. https://doi.org/10.21105/joss.00265
Lin JJ, Lee JY, Golparvar-Fard M (2019) Exploring the potential of image-based 3D geometry and appearance reasoning for automated construction progress monitoring. In: Proceedings of the ASCE international conference on computing in civil engineering (I3CE), ASCE, Atlanta, GA, USA, 2019, pp 162–170. https://doi.org/10.1061/9780784482438.021
Linner T, Shrikathiresan A, Vetrenko M, Ellmann B, Bock T (2011) Modeling and operating robotic environments using Gazebo/ROS. In: Proceedings of the international symposium on automation and robotics in construction (ISARC), IAARC, Seoul, Korea, 2011, pp 957–962. https://doi.org/10.22260/ISARC2011/0177
Lu Y, Xu X (2018) Resource virtualization: a core technology for developing cyber-physical production systems. J Manuf Syst 47:128–140. https://doi.org/10.1016/j.jmsy.2018.05.003
Lu Q, Xie X, Parlikad AK, Schooling JM, Konstantinou E (2020a) Moving from building information models to digital twins for operation and maintenance. In: Proceedings of the Institution of Civil Engineers—smart infrastructure and construction (2020a), pp 1–11. https://doi.org/10.1680/jsmic.19.00011
Lu Q, Xie X, Parlikad AK, Schooling JM (2020b) Digital twin-enabled anomaly detection for built asset monitoring in operation and maintenance. Autom Constr 118:103277. https://doi.org/10.1016/j.autcon.2020.103277
Lundeen KM, Kamat VR, Menassa CC, McGee W (2017) Scene understanding for adaptive manipulation in robotized construction work. Autom Constr 82:16–30. https://doi.org/10.1016/j.autcon.2017.06.022
Lundeen KM, Kamat VR, Menassa CC, McGee W (2019) Autonomous motion planning and task execution in geometrically adaptive robotized construction work. Autom Constr 100:24–45. https://doi.org/10.1016/j.autcon.2018.12.020
Macher H, Landes T, Grussenmeyer P (2017) From point clouds to building information models: 3D semi-automatic reconstruction of indoors of existing buildings. Appl Sci 7:1030. https://doi.org/10.3390/app7101030
Madni AM, Madni CC, Lucero SD (2019) Leveraging digital twin technology in model-based systems engineering. Systems 7:7. https://doi.org/10.3390/systems7010007
Malik AA, Brem A (2021) Digital twins for collaborative robots: a case study in human-robot interaction. Robot Comput Integr Manuf 68:102092. https://doi.org/10.1016/j.rcim.2020.102092
Marshall MQ, Redovian C (2019) An application of a digital twin to robotic system design for an unstructured environment. In: Proceedings of the ASME international mechanical engineering congress and exposition, ASME, Salt Lake City, UT, USA, 2019, p V02BT02A010. https://doi.org/10.1115/IMECE2019-11337
Mertens J, Challenger M, Vanherpen K, Denil J (2020) Towards real-time cyber-physical systems instrumentation for creating digital twins. In: Proceedings of the spring simulation conference (SpringSim), IEEE, Fairfax, VA, USA, 2020, pp 1–12. https://doi.org/10.22360/SpringSim.2020.CPS.001
Mohammed A, Schmidt B, Wang L (2017) Active collision avoidance for human–robot collaboration driven by vision sensors. Int J Comput Integr Manuf 30:970–980. https://doi.org/10.1080/0951192X.2016.1268269
Musić S, Hirche S (2017) Control sharing in human-robot team interaction. Annu Rev Control 44:342–354. https://doi.org/10.1016/j.arcontrol.2017.09.017
Naboni R, Kunic A (2019) A computational framework for the design and robotic manufacturing of complex wood structures. In: Proceedings of the education and research in computer aided architectural design in Europe and Iberoamerican Society of Digital Graphics, joint conference, Porto, Portugal, 2019, pp 189–196. https://doi.org/10.5151/proceedings-ecaadesigradi2019_488
Nikolakis N, Maratos V, Makris S (2019) A cyber physical system (CPS) approach for safe human–robot collaboration in a shared workplace. Robot Comput Integr Manuf 56:233–243. https://doi.org/10.1016/j.rcim.2018.10.003
Ochmann S, Vock R, Wessel R, Klein R (2016) Automatic reconstruction of parametric building models from indoor point clouds. Comput Graph 54:94–103. https://doi.org/10.1016/j.cag.2015.07.008
OCTOPUZ (2021) Robot programming and simulation software: offline robotic program (OLRP). https://octopuz.com/. Accessed 7 Dec 2021
Quigley M, Gerkey B, Conley K, Faust J, Foote T, Leibs J, Berger E, Wheeler R, Ng AY (2009) ROS: an open-source robot operating system. In: Proceedings of the IEEE international conference on robotics and automation (ICRA), IEEE, Kobe, Japan, 2009, p 5. https://www.semanticscholar.org/paper/ROS%3A-an-open-source-Robot-Operating-System-Quigley/d45eaee8b2e047306329e5dbfc954e6dd318ca1e#citing-papers. Accessed 5 Jan 2021.
RoboDK (2021) Simulator for industrial robots and offline programming. https://robodk.com/index. Accessed 7 Dec 2021
Sampaio AZ, Berdeja E (2017) Collaborative BIM environment as a support to conflict analysis in building design. In: Proceedings of the experiment@international conference (Exp.at'17), IEEE, Faro, Portugal, 2017, pp 77–82. https://doi.org/10.1109/EXPAT.2017.7984348
Sartori A, Schlette C (2021) Visual programming of a human-machine interface for a multi-robot support system. In: Proceedings of the IEEE international conference on industrial cyber-physical systems (ICPS), IEEE, Victoria, Canada, 2021, pp 387–392. https://doi.org/10.1109/ICPS49255.2021.9468200
Schmidtler J, Knott V, Hölzel C, Bengler K (2015) Human centered assistance applications for the working environment of the future. Occup Ergon 12:83–95. https://doi.org/10.3233/OER-150226
Shahmiri F, Ficca J (2016) A model for real-time control of industrial robots. In: Proceedings of the international symposium on automation and robotics in construction (ISARC), IAARC, Auburn, AL, USA, 2016, pp 1065–1072. https://doi.org/10.22260/ISARC2016/0128
Sharif S, Gentry TR, Sweet LM (2016) Human–robot collaboration for creative and integrated design and fabrication processes. In: Proceedings of the international symposium on automation and robotics in construction (ISARC), IAARC, Auburn, AL, USA, 2016, pp 596–604. https://doi.org/10.22260/ISARC2016/0072
Shin KG, Ramanathan P (1994) Real-time computing: a new discipline of computer science and engineering. Proc IEEE 82:6–24. https://doi.org/10.1109/5.259423

Söderberg R, Wärmefjord K, Carlson JS, Lindkvist L (2017) Toward a digital twin for real-time geometry assurance in individualized production. CIRP Ann 66:137–140. https://doi.org/10.1016/j.cirp.2017.04.038

Song L, Eldin NN (2012) Adaptive real-time tracking and simulation of heavy construction operations for look-ahead scheduling. Autom Constr 27:32–39. https://doi.org/10.1016/j.autcon.2012.05.007

Stojanovic V, Trapp M, Richter R, Hagedorn B, Döllner J (2018) Towards the generation of digital twins for facility management based on 3D point clouds. In: Proceedings of the ARCOM 34th Annual Conference, Belfast, UK, 2018, pp 270–279. https://www.researchgate.net/publication/325737190_Towards_The_Generation_of_Digital_Twins_for_Facility_Management_Based_on_3D_Point_Clouds. Accessed 31 Mar 2021

Tabar RS, Wärmefjord K, Söderberg R, Lindkvist L (2020) Efficient spot welding sequence optimization in a geometry assurance digital twin. J Mech Des 142:1–8. https://doi.org/10.1115/1.4046436

Tandur S (2015) Towards a new BIM "dimension"—translating BIM data into actual construction using robotics. In: Proceedings of the international symposium on automation and robotics in construction (ISARC), IAARC, Oulu, Finland, 2015, pp 1–7. https://doi.org/10.22260/ISARC2015/0051

Usmanov V, Bruzl M, Svoboda P, Šulc R (2017) Modelling of industrial robotic brick system. In: Proceedings of the international symposium on automation and robotics in construction (ISARC), IAARC, Taipei, Taiwan, 2017, pp 1013–1020. https://doi.org/10.22260/ISARC2017/0140

Vasey L, Felbrich B, Prado M, Tahanzadeh B, Menges A (2020) Physically distributed multi-robot coordination and collaboration in construction. Constr Robot 4:3–18. https://doi.org/10.1007/s41693-020-00031-y

Wang C, Cho YK (2015) Smart scanning and near real-time 3D surface modeling of dynamic construction equipment from a point cloud. Autom Constr 49:239–249. https://doi.org/10.1016/j.autcon.2014.06.003

Wang L, Gao R, Váncza J, Krüger J, Wang XV, Makris S, Chryssolouris G (2019) Symbiotic human-robot collaborative assembly. CIRP Ann 68:701–726. https://doi.org/10.1016/j.cirp.2019.05.002

Wang X, Liang C-J, Menassa CC, Kamat VR (2021) Interactive and immersive process-level digital twin for collaborative human–robot construction work. J Comput Civ Eng 35:04021023. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000988

Wu T-H, Wu F, Liang C-J, Li Y-F, Tseng C-M, Kang S-C (2017) A virtual reality tool for training in global engineering collaboration. Univ Access Inf Soc 18:243–255. https://doi.org/10.1007/s10209-017-0594-0

Xiao Y, Taguchi Y, Kamat VR (2018) Coupling point cloud completion and surface connectivity relation inference for 3D modeling of indoor building environments. J Comput Civ Eng 32:04018033. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000776

Xu L, Feng C, Kamat VR, Menassa CC (2019) An occupancy grid mapping enhanced visual slam for real-time locating applications in indoor gps-denied environments. Autom Constr 104:230–245. https://doi.org/10.1016/j.autcon.2019.04.011

Xue F, Lu W, Chen K, Zetkulic A (2019) From semantic segmentation to semantic registration: derivative-free optimization–based approach for automatic generation of semantically rich as-built building information models from 3D point clouds. J Comput Civ Eng 33:04019024. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000839

Yang C-H, Wu T-H, Xiao B, Kang S-C (2019) Design of a robotic software package for modular home builder. In: Proceedings of the international symposium on automation and robotics in construction (ISARC), IAARC, Banff, AB, Canada, 2019, pp 1217–1222. https://doi.org/10.22260/ISARC2019/0163

Zhuang C, Liu J, Xiong H (2018) Digital twin-based smart production management and control framework for the complex product assembly shop-floor. Int J Adv Manuf Technol 96:1149–1163. https://doi.org/10.1007/s00170-018-1617-6