# Demo: Remote Human-Robot Collaboration in XR

Zhewen Yang*, Guodong Chen*, Mayank Chadha, Barath Balamurugan, Mallesham Dasari
{yang.zhewe,chen.guod,chadha.may,balamurugan.b,m.dasari}@northeastern.edu
Northeastern University
Boston, MA, USA

## CCS Concepts

• **Human-centered computing → Extended Reality**.

## Keywords

Extended Reality, Digital Twins, Low Latency Streaming.

## 1 Abstract

As extended reality (XR) technology advances, its potential for human-robot collaboration has become increasingly apparent [5]. However, enabling seamless remote interaction between humans and robots in XR environments presents several challenges in terms of network and compute resources. Remote collaboration, whether through 5G, Wi-Fi, or traditional Internet infrastructure– poses critical concerns like latency, bandwidth, and reliability. On the other hand, rendering and computational efficiency also present challenges in delivering XR experiences for human-robot interaction. XR devices, such as standalone headsets and mobile-driven solutions, are often constrained by processing power and battery life, which can limit the complexity of visualizations and interactions. Offloading rendering tasks to edge or cloud servers offers a promising solution but introduces additional latency and synchronization challenges. Furthermore, ensuring seamless hand-tracking, spatial mapping, and real-time feedback requires highly optimized algorithms and efficient data processing pipelines.

We introduce RoboTwin, a remote human-robot collaboration solution that has the potential for a range of applications like workforce training, remote assistance, and cost-effective maintenance in industrial and manufacturing sectors. RoboTwin leverages XR headsets to monitor and control the robotic arm in a highly interactive and safe environment with minimal latency. To enhance realism, the application integrates hand-tracking capabilities for intuitive interaction with virtual objects. Furthermore, recognizing the importance of real-time feedback, we incorporated an RGB camera to stream live video of the robotic arm into the VR environment, with WebRTC [1, 3] jointly supporting the streaming configuration to ensure that users can seamlessly visualize its physical movements.

**Demo:** The setup of RoboTwin consists of a 6 DoF robot and an XR headset (Quest3). For robot control, we use a Linux-based Jetson microcontroller running ROS2 [4]. It manages the data transmission through ROS topics, leveraging a lightweight publish/subscribe model for efficiency. The demo features a robot located on the Northeastern University campus in Boston, MA, being controlled remotely by a person using a Quest 3 headset at the HotMobile workshop in La Quinta, CA.

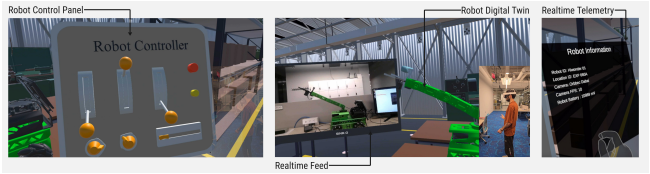


**Figure 1: Operators can use RoboTwin via an extended reality application to control the robot wirelessly, with real-time updates and low-latency camera streaming.**

When using RoboTwin, the user is virtually placed in a factory setting that represents a physical factory with the robot and the virtual robot controller, which is created with Unity. The operator operates the robot controller using hand interactions and does not require external headset controllers. The robot controller supports grab, poke, and pinch interactions to realistically represent the lever, knob, and push, it is responsible for controlling each of the joints of the robot, resetting the robot's position, and also displaying critical information like the temperatures of the robot motors. An external camera is attached that streams the live feed for visualization.

One of the most important aspects for RoboTwin of building an interconnected system is communication and scalability. The VR application needs to have a bidirectional communication channel i.e. send commands to the robot and also receive the current state of all the joints in real-time from the robot. The position of the virtual robot needs to be updated constantly concerning the physical robot. It is also crucial for the systems to handle multiple processes or devices. The system uses ROS2 for communication, which internally uses the Data Distribution Service protocol (DDS) [2], which is a real-time P2P protocol and is used in systems that need high performance and reliability. The robot opens a TCP port for communication and once the communication is established all the devices in the bus can subscribe and publish the data on ROS Topics. Also, the camera frames are transmitted using ROS Topics and designed with WebRTC for low-latency live streaming.

**Demo video link:** https://youtu.be/7PKZyjSJMXk

*Zhewen Yang and Guodong Chen contributed equally to this research.


*HOTMOBILE '25, La Quinta, CA, USA*

ACM ISBN 979-8-4007-1403-0/25/02
https://doi.org/10.1145/3708468.3715687

## References

[1] Mallesham Dasari, Edward Lu, Michael W Farb, Nuno Pereira, Ivan Liang, and Anthony Rowe. 2023. Scaling VR Video Conferencing. In *IEEE VR*. IEEE, 648–657.
[2] Inc. Object Management Group. 2010. Data Distribution Service(DDS). https://www.dds-foundation.org/what-is-dds-3/.
[3] The WebRTC project authors. 2011. WebRTC. https://webrtc.org/.
[4] Open Robotics. 2021. ROS - Robot Operating System. https://www.ros.org/.
[5] Ryo Suzuki et al. 2022. Augmented reality and robotics: A survey and taxonomy for ar-enhanced human-robot interaction and robotic interfaces. In *CHI*. 1–33.