XRoboToolkit: A Cross-Platform Framework for Robot Teleoperation

Zhigen Zhao[1,2], Liuchuan Yu[1,3], Ke Jing[1] and Ning Yang[1]

***Abstract*— The rapid advancement of Vision-Language-Action models has created an urgent need for large-scale, high-quality robot demonstration datasets. Although teleoperation is the predominant method for data collection, current approaches suffer from limited scalability, complex setup procedures, and suboptimal data quality. This paper presents XRoboToolkit[4][5], a cross-platform framework for extended reality based robot teleoperation built on the OpenXR standard. The system features low-latency stereoscopic visual feedback, optimization-based inverse kinematics, and support for diverse tracking modalities including head, controller, hand, and auxiliary motion trackers. XRoboToolkit's modular architecture enables seamless integration across robotic platforms and simulation environments, spanning precision manipulators, mobile robots, and dexterous hands. We demonstrate the framework's effectiveness through precision manipulation tasks and validate data quality by training VLA models that exhibit robust autonomous performance.**

I. Introduction

Recent advances in deep generative robot learning [1], especially Vision-Language-Action Models (VLAs) [2]–[5], are critically dependent on large-scale, high-quality datasets containing robot skill demonstrations. Robot teleoperation [6]–[10] is one of the primary approaches for generating human demonstrations of complex manipulation and mobility tasks, leveraging human operators' natural ability to generalize across diverse environments and tasks.

Recent robot teleoperation frameworks follow several paradigms, each with distinct trade-offs. Leader-follower approaches [11] offer low latency and intuitive operation but require custom hardware tailored to specific robot platforms, limiting scalability and accessibility. Vision-based teleoperation systems [12] provide greater flexibility and generalizability across diverse robotic hardware but often suffer from unstable tracking performance and higher latency, degrading operator performance and data quality. Virtual Reality (VR) or Extended Reality (XR) teleoperation [13]–[16] has emerged as a promising alternative, utilizing commercially available headsets to create intuitive control interfaces with stereoscopic visual feedback that generalize across multiple platforms. However, existing XR solutions remain difficult to configure and often rely on individual Unity SDKs or WebXR platforms that introduce additional latency and compatibility challenges. Another significant limitation is the lack of standardized data formats between XR devices and robot controllers, necessitating substantial integration work for new XR devices or robot platforms.

To address these limitations, we present XRoboToolkit—a comprehensive suite of cross-device software development kits and applications for real-time robot teleoperation via XR devices. The toolkit provides a generalized interface layer that resolves standardization challenges by adopting OpenXR [17] conventions on the XR side and modular, extensible Python and C++ interfaces on the robot side for seamless integration across robotic platforms. Current support includes devices such as PICO 4 Ultra and Meta Quest 3.

A major contribution is the stereoscopic visual feedback system, which integrates a low-latency communication protocol and a highly efficient video streaming pipeline, both optimized to minimize latency and reduce motion sickness.

On the robot side, the system employs a quadratic programming (QP)-based inverse kinematics solver (IK) that generates smooth, reliable robot motion, particularly near kinematic singularities, and incorporates dexterous hand tracking for retargeting human hand motions to robotic hands in fine-grained manipulation tasks. The modular architecture of XRoboToolkit enables straightforward integration with diverse robotic systems and simulation environments, tested on platforms such as UR5 and ARX R5 arms, the Galaxea R1-Lite mobile manipulator, the Shadow dexterous hands, with native support for MuJoCo [18].

Sec. II details the architecture of the XRoboToolkit. Example applications are presented in Sec. III, performance evaluation in Sec. IV, and conclusions in Sec. V.

II. Teleoperation System

A. Overview

Fig. 1 presents an overview of the XRoboToolkit architecture. The `XRoboToolkit-Unity-Client` application, deployed on XR headsets, captures pose tracking data and delivers a stereoscopic visual interface for the human operator. This pose tracking data, including head, hand, controller, full-body, and object tracking (via motion trackers), is transmitted to the robot client via `XRoboToolkit-PC-Service` in C++, the specific tracking data format is discussed in Sec. II-B. Additionally, the package `XRoboToolkit-PC-Service-Pybind` allows direct access to the XR tracking data in Python without handling the raw data structure. Stereo vision is enabled either through the onboard cameras of PICO head-

[1]ByteDance, PICO, San Jose, CA, 95110, USA. Please send correspondence to `ning.yang@bytedance.com`

[2]Georgia Institute of Technology, Institute for Robotics and Intelligent Machines (IRIM), Atlanta, GA 30332, USA `zhigen.zhao@gatech.edu`

[3]George Mason University, Computer Science, Fairfax, Virginia 22030, USA `lyu20@gmu.edu`

[4]Website: `https://xr-robotics.github.io`

[5]Github: `https://github.com/XR-Robotics`

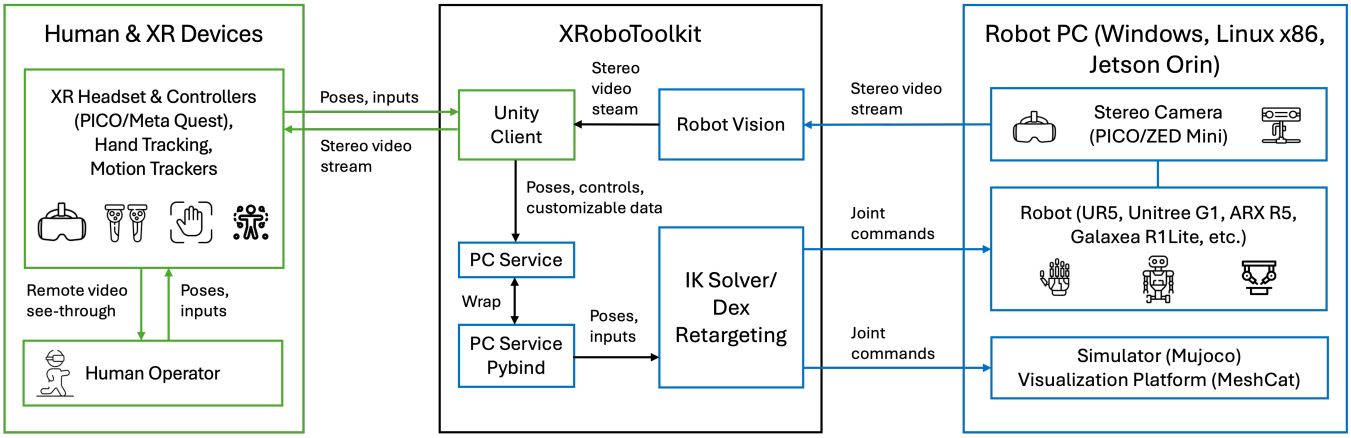

Fig. 1. Overview of XRoboToolkit, an integrative framework bridging XR and robotics. Core functionalities include real-time teleoperation and stereoscopic vision. Green blocks represent XR-side components, while blue blocks indicate components on the robot side.

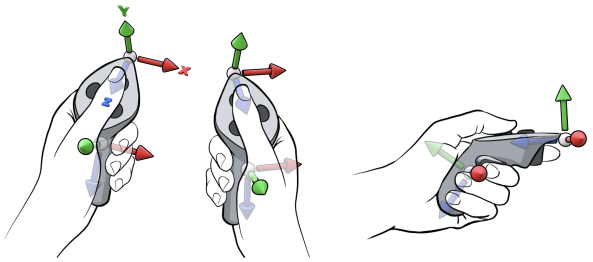

Fig. 2. OpenXR conventions for pose tracking coordinate system [17].

sets or an external ZED Mini camera with the module `XRoboToolkit-Robot-Vision`. The IK and dexterous retargeting solvers are implemented in robot teleoperation module `XRoboToolkit-Teleop-Sample-Python` to provide support for both simulated environments, such as Mujoco, and physical robot platforms, including the UR5, ARX R5, and Galaxea R1-Lite. The XRoboToolkit's modular architecture facilitates easy integration with additional simulators and robotic platforms, providing a flexible and extensible solution for stereoscopic teleoperation in virtual and physical environments.

B. Data Streaming

`XRoboToolkit-PC-Service` employs an asynchronous, callback-driven architecture for real-time data streaming from VR hardware to client applications. Communication is managed via a dedicated SDK handling connection to the streaming service and data payload reception.

XR Data Formats: Following OpenXR conventions, all positional and rotational data use a right-handed coordinate system with X-axis right, Y-axis up, and Z-axis backwards, as shown in Fig. 2(a). The origin is established at the user's head position when the application launches. The 6 degree-of-freedom (DOF) pose data are formatted as seven floating-point numbers separated by commas: 3D position vector $[x, y, z]$ followed by quaternion $[qx, qy, qz, qw]$.

All real-time tracking data are transmitted within a single JSON object at 90 Hz. This design simplifies client-side parsing and ensures consistent data structure regardless of enabled tracking features. Table I provides an overview of main tracking data fields in XRoboToolkit.

TABLE I
XR TRACKING DATA FORMATS

Type	Field	Description
Head	`pose`	Headset pose
	`status`	Tracking confidence (0: unreliable, 1: reliable)
	`handMode`	Input mode (0: None, 1: controller, 2: hand)
Controller	`pose`	Controller pose
	`axisX, axisY`	Joystick position
	`axisClick`	Joystick press state
	`grip`	Grip input
	`trigger`	Trigger input
	`primaryButton`	X Button (L), A Button (R)
	`secondaryButton`	Y Button (L), B Button (R)
	`menuButton`	Menu (L), Screenshot/Record (R)
Hand	`isActive`	Tracking status (0: not active, 1: active)
	`scale`	Hand scale factor
	`HandJointLocations`	Array of 26 hand joint data entries
Whole-Body	`joints`	Array of 24 body joint data entries
Motion Tracker	`p`	Pose
	`va`	Velocity & angular velocity
	`wva`	Acceleration & angular acceleration
	`sn`	Unique serial number of tracker

Head Tracking: Head tracking data contains headset pose, status integer indicating tracking confidence, and hand mode integer specifying active input mode.

Controller Tracking: Controller tracking captures both left and right controllers' poses with button and joystick states. Joystick axes `axisX` and `axisY` provide floating-point values from -1 to 1. `grip` and `trigger` inputs are analog controls with values between 0 and 1 indicating pressure intensity. Remaining buttons provide binary state

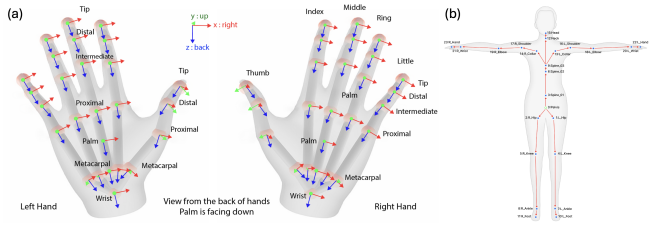

Fig. 3. Conventions for (a) hand joints [17] and (b) body joints.

information.

Hand Gesture Tracking: Each hand gesture is represented through 26 joint poses: 4 joints on the thumb, 5 joints on each remaining finger, plus palm and wrist joints, as illustrated in Fig. 3(a). Hand tracking data includes tracking quality, scale factor, and an array of 26 joint data entries per hand. Each entry contains 6-DOF pose with additional metadata including tracking status and joint radius. While transmitted with the JSON object at 90 Hz, hand tracking data updates at 60 Hz due to camera limitations.

Whole-Body Motion Capture: Whole-body joint tracking consists of 24 joint data entries corresponding to major human model joints, as shown in Fig. 3(b). Each entry contains joint pose, velocity, and acceleration. The 24-joint model follows PICO's standard, as OpenXR currently lacks a standardized whole-body model.

Motion Tracker: For PICO 4 Ultra headsets, we support object tracking mode for auxiliary motion trackers. Motion tracker data captures pose, velocity, and acceleration measurements with serial numbers for tracker identification.

C. Robot Control

The robot control module maps XR tracking state to robot commands through distinct control modes: IK for manipulator control, dexterous hand retargeting, head tracking, and mobile base control.

1) Inverse Kinematics: For manipulator control, we implement a QP-based IK solver using PlaCo [19], built on the Pinocchio rigid body dynamics library [20]. The QP problem is defined as:

$$\begin{aligned} &\min_{\dot{\mathbf{q}}} \sum_{i=1}^{N} \mathbf{w}_i \|\mathbf{J}_i(\mathbf{q})\dot{\mathbf{q}} + \mathbf{e}_i(\mathbf{q})\|^2 \\ &\text{s.t. } l \leq \mathbf{C}(\mathbf{q})\dot{\mathbf{q}} \leq u, \end{aligned} \tag{1}$$

where $\mathbf{q}$ represents manipulator configuration; each task i is defined as residual function $\mathbf{e}_i(\mathbf{q})$ with weight $\mathbf{w}_i$; $\mathbf{J}_i(\mathbf{q})$ denotes task Jacobian; and $\mathbf{C}(\mathbf{q})$ represents additional constraint matrix.

The optimization-based IK approach enables easy inclusion of constraints and regularization terms. To improve robot stability near singularities, a regularization term maximizes manipulability [21]:

$$m = \sqrt{\det(\mathbf{J}(\mathbf{q})\mathbf{J}(\mathbf{q})^\top)} \tag{2}$$

When using VR controllers, end-effector tracking activates when the user holds the `grip` button. For stable, intuitive experience, the system uses relative motion: the robot's end-effector tracks controller displacement relative to its state when `grip` was first pressed.

Auxiliary motion trackers can be attached to the user's body (e.g., elbow) to introduce additional pose constraints into the QP-based IK. This enables nuanced control over robot full-body posture by mapping operator positions to equivalent robot links, particularly useful for resolving null-space redundancies and achieving anthropomorphic motions.

2) Dexterous Hand Retargeting: For dexterous manipulation tasks, keypoint positions from the OpenXR hand model (Fig. 3(a)) are obtained via the XR headset's hand tracking. These keypoints are mapped to robot hand joint space via:

$$\begin{aligned} &\min_{\mathbf{q}_t} \sum_{i=1}^{N} \|\alpha \mathbf{v}_t^i - f_i(\mathbf{q}_t)\|^2 + \beta \|\mathbf{q}_t - \mathbf{q}_{t-1}\|^2, \\ &\text{s.t. } \mathbf{q}_l \leq \mathbf{q}_t \leq \mathbf{q}_u, \end{aligned} \tag{3}$$

where $\mathbf{q}_t$ is robot hand joint configuration at time t, $\mathbf{v}_t^i$ is the i-th keypoint position in the human hand model, $f_i(\mathbf{q}_t)$ computes corresponding robot hand position, α is a scaling factor for different hand sizes, and β is regularization weight for smooth motion. Implementation uses `dex_retargeting` [12].

3) Mobile Base Control: For mobile manipulators with omnidirectional platforms, the mobile base is controlled by XR controller joysticks. The left joystick's X and Y axes issue linear velocity commands in the robot's sagittal and coronal planes, while the right joystick's X-axis controls angular velocity, providing intuitive mobility interface during manipulation tasks.

D. XR Unity Application

Fig. 4 presents the application interface for `XRoboToolkit-Unity-Client`, which contains five panels: Network, Tracking, Remote Vision, Data Collection, and Log.

The **Network panel** displays essential headset status information, including serial number (SN), IP address, frame rate, connection status with the PC service, and configured service IP. The **Tracking panel** organizes controls into four functional groups. The `Source` group allows users to select pose data types for tracking (e.g., head, controller, hand). The `PICO Motion Tracker` group configures tracking modes: None, Full-Body, or Object. Note that for Meta Quest, object tracking is unavailable since the auxiliary motion tracker is not supported. The `Data & Control` group provides toggles for pose data transmission to the PC service. The `Status` group provides live tracking status. The **Remote Vision panel** handles the stereoscopic vision state. The `State` field displays current video stream status. The XR application currently supports PICO 4 Ultra and ZED Mini cameras as streaming sources, with extensibility for additional camera sources through modifications of a YAML configuration file. The **Data Collection panel** enables recording of both pose and visual data streams, with automatic local storage and timestamp-based indexing for

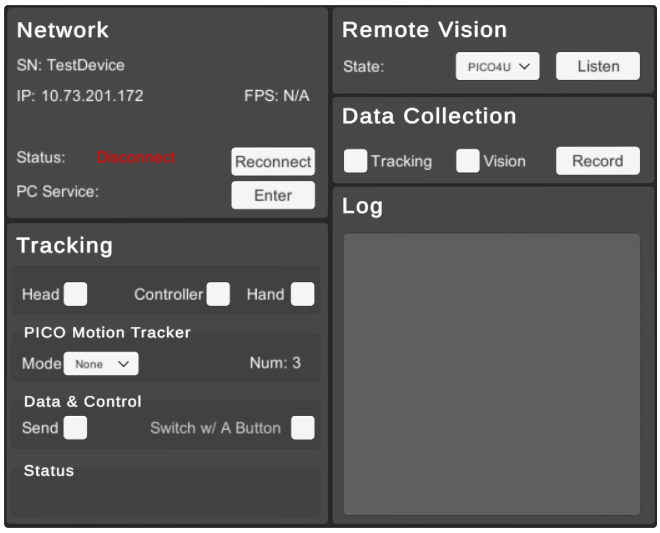

Fig. 4. Screenshot of the PICO version of the XR Unity Application.

future reference. The **Log panel** provides real-time system diagnostics and monitoring information for debugging purposes.

E. Stereoscopic Visual Feedback

Our toolkit currently supports two stereo video sources: the PICO 4 Ultra headset and the ZED Mini camera. The PICO 4 Ultra operates as a standalone solution, whereas the ZED Mini requires connection to an external computing platform—such as a Windows or Linux PC, or an NVIDIA Orin device—to enable video streaming. When using the PICO 4 Ultra, operators benefit from a homogeneous visual experience, as both display and capture are provided through the same hardware platform.

To achieve stereoscopic vision, we implemented a custom shader that adjusts the interpupillary distance and sets the focal point at approximately 3.3 feet. This configuration provides enhanced three-dimensional depth perception, a distance considered optimal for teleoperation tasks, although it comes at the cost of sacrificing far-distance depth accuracy.

Our empirical observations indicate that the PICO 4 Ultra delivers superior visual quality compared to the ZED Mini, particularly in terms of tone reproduction, brightness, color accuracy, and dynamic range. Additionally, the PICO 4 Ultra offers a more balanced field of view (FOV), with horizontal and vertical angles of approximately 76.35° and 61.05°, respectively, compared with the 82° horizontal and 52° vertical FOV of the ZED Mini.

III. Applications and Demonstrations

Fig. 5 demonstrates the versatility of XRoboToolkit across diverse robotic platforms and simulation environments. The framework's modular architecture enables seamless integration with both hardware systems and virtual environments, supporting a wide range of teleoperation scenarios from precise manipulation to mobile robotics and dexterous hand control. Please refer to the supplementary video[6] for more information.

[6]Video link: `https://youtu.be/g6QJX2s-RCo`

A. XR Controller-Based Teleoperation

XRoboToolkit supports intuitive teleoperation using XR controllers for both dual-arm manipulation systems and mobile manipulators. The operator wears an XR headset and uses handheld controllers to directly control robot end-effectors through the inverse kinematics solver described in Sec. II-C.1. Note that the XR headset can be worn around the neck when stereoscopic visual feedback is not required. This configuration reduces head-mounted weight and fatigue while maintaining full controller tracking functionality, making it particularly suitable for tasks where operators can rely on direct visual observation of the robot workspace.

As shown in Fig. 5(a), the controller-based teleoperation has been validated on multiple platforms, including dual ARX R5 manipulators for long-horizon tasks such as bimanual carpet folding, and the Galaxea R1-Lite mobile manipulator for transportation and placement tasks. The system has also been used in the applications discussed in Sec. III-B-III-C.

B. Precision Manipulation with Active Stereo Vision

Fig. 5(b) illustrates our dual UR5 setup equipped with a 2-DOF active head and stereo vision feedback. This configuration demonstrates the framework's capability for high-precision manipulation tasks. The active head tracking system employs a 2-DOF gimbal that provides yaw and pitch rotation following the operator's head movements in real-time. The roll DOF is intentionally omitted to prevent motion sickness caused by inconsistency between visual and vestibular senses [22]. For stereoscopic visual feedback, a PICO 4 Ultra headset is mounted on the active head to serve as the stereo camera system that provides 2160×810 resolution stereo video streaming at 60 Hz.

The system is validated on high-precision insertion tasks, specifically inserting a screwdriver with a 3mm diameter into a circular hole with a 4mm diameter, requiring precise spatial perception and fine motor control with only 0.5mm tolerance on each side.

C. Motion Tracker for Redundant Manipulator Control

For redundant manipulators, auxiliary motion trackers can be integrated to provide additional control. Fig. 5(c) demonstrates the motion tracker teleoperation with a Unitree G1 upper body visualized in MeshCat. The motion trackers are attached to the operator's elbows to provide position-only tracking that serves as additional inverse kinematics constraints for 7-DOF arms. This elbow tracking enables intuitive control of redundant arms by resolving kinematic redundancies in an anthropomorphic manner, allowing operators to achieve more natural arm configurations while maintaining end-effector tracking from the controllers.

D. Dexterous Hand Control in MuJoCo

Fig. 5(d) showcases the hand pose tracking task within a MuJoCo simulation. In contrast to demonstrations that rely on XR controllers, this configuration utilizes the hand tracking mode of the headset to directly capture finger and

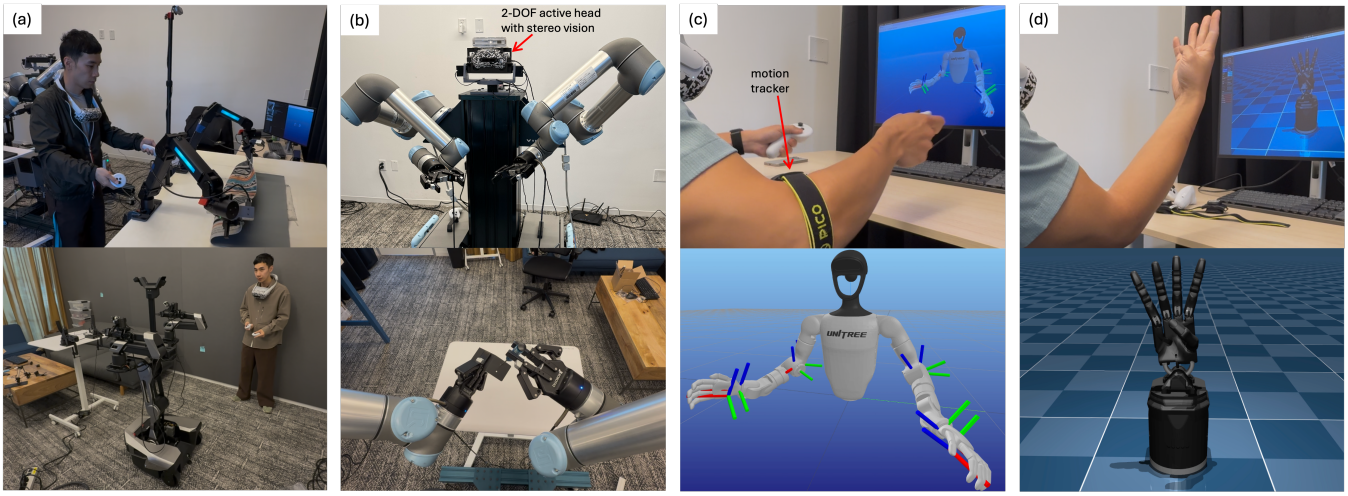

Fig. 5. Example applications of XRoboToolkit: (a) teleoperation with XR controllers for dual arm manipulation and mobile manipulators, (b) Dual UR5 manipulators with 2-DOF head tracking and stereo vision, (c) auxiliary motion trackers for robot elbow control in MeshCat visualization, and (d) dexterous hand tracking in Mujoco simulation.

hand gestures. The implementation employs the dexterous hand retargeting approach described in Sec. II-C.2, mapping the 26-joint OpenXR hand model to the Shadow Hand's kinematic structure. The hand pose tracking task demonstrates XRoboToolkit's capability to support dexterous manipulation, enabling operators to perform fine manipulation tasks through direct hand gesture control without requiring additional hardware beyond the XR headset.

IV. Experiments

A. Video Streaming Latency Comparison

A low-latency video streaming solution was developed and comparatively evaluated against Open-TeleVision [14]. To capture both virtual and real-world perspectives simultaneously, a Kandao QooCam EGO 3D Camera was used for dual-view recording. Latency measurements utilized a Precise Timing Measurement LED Panel cycling at 100 Hz. For each measurement, the illuminated LED sequence was identified, with the timestamp taken at the sequence midpoint. Latency was defined as the temporal offset between the VR display and the corresponding real-world LED panel view. Video footage was captured using OBS Studio, sampling 10 frames per condition to compute mean and standard deviation. Fig. 6 illustrates the measurement method and system setup.

Three conditions were evaluated: 1) Open-TeleVision, 2) ZED Mini to PICO 4 Ultra, and 3) PICO 4 Ultra to PICO 4 Ultra. For conditions 1 and 2, the ZED Mini connected to a Windows 11 laptop (Intel i9-13900HK, 32 GB RAM, NVIDIA RTX 4080) using ZED SDK 4.0.8. Condition 3 used no PC; video streamed directly from the PICO 4 Ultra headset. All devices are connected to the same local network. The video transmission parameters are 1280×720 resolution, 60 FPS, and 1 Mbps bitrate, matching Open-TeleVision's default configuration.

Results are summarized in Table II. XRoboToolkit (ZED Mini – PICO 4 Ultra) achieved the lowest mean latency (82.00 ms), substantially outperforming Open-TeleVision (121.50 ms) and XRoboToolkit (PICO 4 Ultra – PICO 4 Ultra) at 100.50 ms. The ZED Mini – PICO 4 Ultra configuration benefits from external laptop processing, which is more computationally powerful than the standalone PICO 4 Ultra headset, likely contributing to superior latency performance. The PICO 4 Ultra – PICO 4 Ultra configuration showed the lowest variability (STD = 3.12 ms), indicating more consistent performance, while ZED Mini – PICO 4 Ultra and Open-TeleVision setups showed higher fluctuations (6.32 ms and 6.01 ms, respectively).

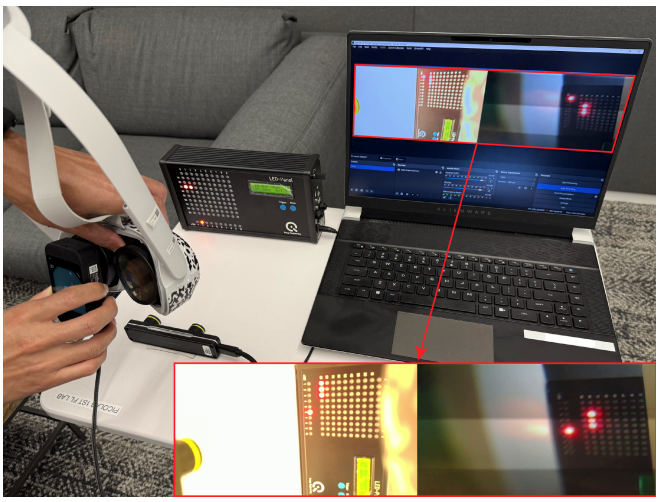

Fig. 6. Video streaming latency measurement method.

TABLE II
Video Streaming Latency Comparison

Approach (Hardware)	Open-TeleVision (ZED Mini - Quest 3)	XRoboToolkit (ZED Mini - PICO 4 Ultra)	XRoboToolkit (PICO 4 Ultra - PICO 4 Ultra)
Mean (ms)	121.50	82.00	100.50
STD (ms)	6.01	6.32	3.12

B. Data Collection for VLA Fine-tuning

To validate that the teleoperation and data collection pipeline provided by XRoboToolkit can generate high-quality

demonstration data suitable for VLA training, we collected 100 demonstrations of a bimanual carpet folding task using the ARX R5 dual-arm system equipped with RealSense D405i wrist cameras and a D435i overhead camera. The task sequence involves first folding the carpet in half along the short edge, then folding it again along the long edge, and finally pulling the carpet aside with the right arm. Each demonstration was recorded at 50 FPS, with every frame containing 14-dimensional robot joint states, 14-dimensional position control commands, and 424×240 RGB images from all three cameras. The average task completion time for each teleoperation demonstration was 20 seconds, with occasional regrasping and repositioning behaviors.

The dataset was used for Low-Rank Adaptation fine-tuning on the π_0 model [3]. Training was conducted for 80,000 steps with a batch size of 16 and an action horizon of 50 frames. The resulting policy achieved a 100% success rate during 30 minutes of continuous operation with an average task completion time of 30 seconds. Importantly, the policy demonstrated adaptive behaviors including autonomous regrasping when grippers failed to secure the carpet and intelligent repositioning when the carpet was off-center.

V. Conclusions

This paper presents XRoboToolkit, a cross-platform framework for XR-based robot teleoperation that addresses key limitations in existing systems through low-latency stereoscopic feedback, optimization-based control, and modular architecture. The framework demonstrates versatility across diverse robotic platforms and validates its effectiveness through precision manipulation tasks and VLA model training. While XRoboToolkit provides significant advances in accessibility and scalability, certain limitations remain. Current whole-body tracking relies on PICO's 24-joint model due to the absence of standardized whole-body definitions in OpenXR, potentially creating compatibility issues with other XR brands using different skeletal models. Furthermore, while whole-body tracking data is provided, it has not been validated through retargeting to humanoid robots for whole-body teleoperation. Additionally, the hand retargeting framework assumes each joint is individually controllable, and therefore cannot accurately retarget to robot hands with mechanical constraints that couple joint movements, such as the INSIPRE Hands. The framework currently supports only MuJoCo simulation, limiting its applicability across diverse simulation environments.

Future work will focus on improving hand retargeting algorithms for underactuated systems, expanding simulation support through platforms such as Roboverse [23] to enable multi-simulator compatibility, and developing humanoid teleoperation capabilities with validated whole-body motion retargeting [24]. Additionally, we will contribute to OpenXR standardization efforts to enable more consistent cross-platform compatibility.

References

[1] J. Urain, A. Mandlekar, Y. Du, M. Shafiullah, D. Xu, K. Fragkiadaki, G. Chalvatzaki, and J. Peters, "Deep generative models in robotics: A survey on learning from multimodal demonstrations," *arXiv:2408.04380*, 2024.

[2] R. Sapkota, Y. Cao, K. I. Roumeliotis, and M. Karkee, "Vision-language-action models: Concepts, progress, applications and challenges," *arXiv:2505.04769*, 2025.

[3] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter *et al.*, "π_0: A vision-language-action flow model for general robot control," *arXiv:2410.24164*, 2024.

[4] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi *et al.*, "Openvla: An open-source vision-language-action model," *arXiv:2406.09246*, 2024.

[5] G. R. Team, S. Abeyruwan, J. Ainslie, J.-B. Alayrac, M. G. Arenas, T. Armstrong, A. Balakrishna, R. Baruch, M. Bauza, M. Blokzijl *et al.*, "Gemini robotics: Bringing ai into the physical world," *arXiv:2503.20020*, 2025.

[6] W. Si, N. Wang, and C. Yang, "A review on manipulation skill acquisition through teleoperation-based learning from demonstration," *Cogn. Comput. Syst.*, vol. 3, no. 1, pp. 1–16, 2021.

[7] K. Darvish, L. Penco, J. Ramos, R. Cisneros, J. Pratt, E. Yoshida, S. Ivaldi, and D. Pucci, "Teleoperation of humanoid robots: A survey," *IEEE Trans. Robot.*, vol. 39, no. 3, pp. 1706–1727, 2023.

[8] H. Li, Y. Cui, and D. Sadigh, "How to train your robots? the impact of demonstration modality on imitation learning," *arXiv:2503.07017*, 2025.

[9] T. Buckley and J. E. Colgate, "Dexterous manipulation with a bimanual anthropomorphic teleoperation robot," in *Proc. Workshop Toward Robot Avatars, IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2024.

[10] A. Iyer, Z. Peng, Y. Dai, I. Guzey, S. Haldar, S. Chintala, and L. Pinto, "Open teach: A versatile teleoperation system for robotic manipulation," in *Proc. Conf. Robot Learn.* PMLR, 2025, pp. 2372–2395.

[11] T. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," *Robot. Sci. Syst.*, 2023.

[12] Y. Qin, W. Yang, B. Huang, K. Van Wyk, H. Su, X. Wang, Y.-W. Chao, and D. Fox, "Anyteleop: A general vision-based dexterous robot arm-hand teleoperation system," in *Robot. Sci. Syst.*, 2023.

[13] X. Wang, L. Shen, and L.-H. Lee, "Towards massive interaction with generalist robotics: A systematic review of xr-enabled remote human-robot interaction systems," *arXiv:2403.11384*, 2024.

[14] X. Cheng, J. Li, S. Yang, G. Yang, and X. Wang, "Open-television: Teleoperation with immersive active visual feedback," in *Proc. Conf. Robot Learn.*, 2024.

[15] F. E. Jedrzej Orbik, "Oculus reader: Robotic teleoperation interface," 2021, accessed: 2025-07-08. [Online]. Available: https://github.com/rail-berkeley/oculus_reader

[16] M. Seo, S. Han, K. Sim, S. H. Bang, C. Gonzalez, L. Sentis, and Y. Zhu, "Deep imitation learning for humanoid loco-manipulation through human teleoperation," in *Proc. IEEE/RAS Int. Conf. Humanoid Robot.*, 2023, pp. 1–8.

[17] The Khronos OpenXR Working Group, "The OpenXR specification, version 1.0.34," 2024, accessed: 2025-07-09. [Online]. Available: https://registry.khronos.org/OpenXR/specs/1.0/html/xrspec.html

[18] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2012, pp. 5026–5033.

[19] Rhoban, "Placo: Rhoban planning and control," 2025, accessed: 2025-07-14. [Online]. Available: https://github.com/Rhoban/placo

[20] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *Proc. IEEE Int. Symp. Syst. Integr.*, 2019.

[21] J. Haviland and P. Corke, "A purely-reactive manipulability-maximising motion controller," *arXiv:2002.11901*, 2020.

[22] B. De Graaf, W. Bles, and J. E. Bos, "Roll motion stimuli: sensory conflict, perceptual weighting and motion sickness," *Brain Res. Bull.*, vol. 47, no. 5, pp. 489–495, 1998.

[23] H. Geng, F. Wang, S. Wei, Y. Li, B. Wang, B. An, C. T. Cheng, H. Lou, P. Li, Y.-J. Wang *et al.*, "Roboverse: Towards a unified platform, dataset and benchmark for scalable and generalizable robot learning," *arXiv:2504.18904*, 2025.

[24] Y. Ze, Z. Chen, J. P. AraÃšjo, Z.-a. Cao, X. B. Peng, J. Wu, and C. K. Liu, "Twist: Teleoperated whole-body imitation system," *arXiv:2505.02833*, 2025.