BEAVR: Bimanual, multi-Embodiment, Accessible, Virtual Reality Teleoperation System for Robots

Alejandro Posadas-Nava*
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology, Cambridge, USA
email or ORCID

Alejandro Carrasco*
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology, Cambridge, USA
acarra@mit.edu

Richard Linares
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology, Cambridge, USA
email or ORCID

***Abstract*—BEAVR is an open-source, bimanual, multi-embodiment Virtual Reality (VR) teleoperation system for robots, designed to unify real-time control, data recording, and policy learning across heterogeneous robotic platforms. BEAVR enables real-time, dexterous teleoperation using commodity VR hardware, supports modular integration with robots ranging from 7-DoF manipulators to full-body humanoids, and records synchronized multi-modal demonstrations directly in the LeRobot dataset schema. Our system features a zero-copy streaming architecture achieving ≤ 35 ms latency, an asynchronous "think–act" control loop for scalable inference, and a flexible network API optimized for real-time, multi-robot operation. We benchmark BEAVR across diverse manipulation tasks and demonstrate its compatibility with leading visuomotor policies such as ACT, DiffusionPolicy, and SmolVLA. All code is publicly available, and datasets are released on Hugging Face[1]**

***Index Terms*—Virtual Reality, Teleoperation, Policy Training, Open-source**

I. Introduction

Robots are transitioning from structured industrial lines to dynamic, human-centric settings, where *dexterous, bimanual manipulation* is a prerequisite for useful autonomy. However, current teleoperation pipelines remain fragmented, often tied to specific hardware, proprietary formats, or custom frameworks, hindering reproducibility and slowing progress in human-robot interaction (HRI) and robotics research in general.

Teleoperation exists in various forms. Classical actuation typically involves transmitting forces from grounded handles to robot grippers. Leader-follower approaches mirror movements between two identical robotic arms, offering direct motion mapping but limiting versatility due to symmetry constraints and limited ability to mimic natural, diverse actions. More recently, Virtual Reality (VR)-based teleoperation offers a promising path forward. The availability of consumer-grade devices such as the Meta Quest has inspired recent systems such as OpenTeach [1] and Bunny VisionPro [2], which enable bimanual demonstrations via immersive interfaces. Yet, these platforms are limited in key ways: they feature a steep learning curve, lack accessible system documentation, log data in non-standard formats, and offer no standardized bridge to learning-ready datasets.

We present **BEAVR**, an open-source, end-to-end VR teleoperation and policy learning system that unifies control, recording, and learning among heterogeneous robots. It enables real-time bimanual control of diverse end-effectors, such as a 7-degree-of-freedom (DoF) xArm or a full-body RX-1 humanoid, through a single VR interface, while capturing synchronized multi-modal demonstrations directly in the standardized LeRobot dataset format. The LeRobot framework [3] provides a unified schema to store robot demonstrations along with a user-friendly policy learning pipeline, supporting a robotics community focused on making AI for real-world robotics more accessible. BEAVR also supports real-time RGB and RGB-D video streaming from commodity cameras and integrates precise tracking using Meta Quest 3S VR glasses. In particular, a complete setup that combines cameras, VR headset, and robotic hardware (RX-1 with LeapHand) can be assembled for approximately \$1000[2], underscoring the system's affordability and potential for broad adoption. Beyond affordability, this work addresses a key question: *can a single VR interface scale to multiple heterogeneous robots without incurring a latency penalty*?

Contributions. Our primary contributions are: (i) a hardware-agnostic virtual-reality teleoperation interface that scales from a 7-DoF tabletop arm to full-body humanoids; (ii) a zero-copy, latency-aware streaming architecture that sustains sub-35 ms end-to-end delay over commodity networks; (iii) a dataset-native logger that exports demonstrations directly in the LeRobot schema for immediate use with modern imitation-learning frameworks; and (iv) an open benchmark suite comprising six manipulation tasks, 50 expert demonstrations, and pretrained baseline policies.

*Equal contribution

[1]Code, datasets, and VR app available at https://github.com/ARCLab-MIT/BEAVR-Bot.

[2]Indicative bill of materials (July 2025): 16×Feetech STS3215-C018 smart servos at \$34 ea [4] (total \$544), Meta-Quest-3S VR headset \$299 [5], LEAPHand parts kit \$200 [6], and ≈1.5 kg PLA/TPU filament for printed frames ≈\$40 [7]—yielding a total hardware cost of ≈\$1.08 k.

The remainder of this paper is organized as follows. Section II reviews prior teleoperation systems and datasets. Section III details the BEAVR architecture. Section IV outlines the pipeline, including coordinate transformations, inverse kinematics, and networking. Section V describes LeRobot integration and the dataset format. Section VI presents experimental results, including task success, policy learning, and system performance. Sections VII and VIII discuss findings, limitations, and future work. Section IX concludes.

II. Related Work

Teleoperation has long been a key strategy for collecting reliable demonstrations across robotic tasks. Early systems typically relied on direct actuation methods, including leader-follower setups in which two identical robotic arms were physically or virtually linked to mirror each other's motion. Other common approaches included joystick-based control or input devices such as 6-DoF controllers, which allowed users to command robot movement from a distance. These systems enabled early progress in remote manipulation, but often required hardware symmetry, offered limited dexterity, or depended on tightly coupled hardware-software configurations that were difficult to scale or adapt to new tasks.

Recent systems have explored VR and AR interfaces for more intuitive, human-centered teleoperation. These approaches use hand tracking [1], [2], [8], glove-based input [9], [10], or controller-based interaction [11] to control robot manipulators in real time. Some allow direct mapping of hand or arm motion to robot joints, while others introduce intermediate steps, such as moving a virtual end-effector in AR before executing commands on the physical robot [12]. The systems most similar to ours rely on VR hand tracking, sometimes augmented with haptic gloves or armbands. While effective for bimanual and finger-level manipulation, these platforms tend to be tied to specific hardware, are often not fully open source, and either lack affordability or store data in proprietary formats. Some systems address one or two of these limitations, but very few meet all of them together. Without openness, low cost, hardware flexibility, and standardized data output, it is difficult to achieve reusability, extensibility, or broad adoption.

Open-source projects like LEAP Hand [6] and RX-1 [13] showcase creative, low-cost integration, yet they still lack a standardized software ecosystem for broad reuse.

III. System Overview

BEAVR is an open-source teleoperation system that translates VR-based human hand movements into real-time robot control, enabling dexterous, multi-embodiment manipulation. The architecture is organized into three main modules: teleoperation, data collection, and model training.

The **teleoperation module** uses a modular component-based design. It consists of three core components that run as separate processes and communicate via lightweight ZMQ channels:

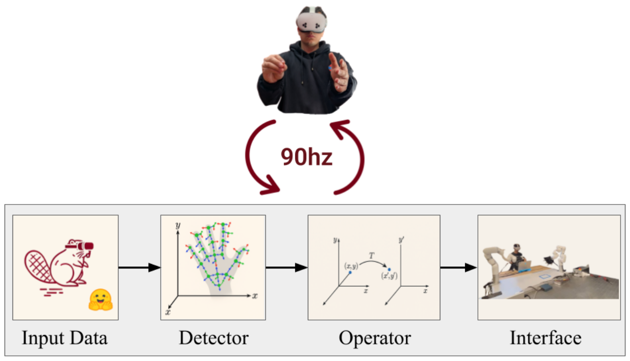

Fig. 1. Data flow visualization through the system components of the teleoperation stack. Hand keypoint data flows from the input device (a VR headset) to the detector. The operator takes the raw keypoint data, creates a coordinate frame, computes coordinate transformations, and sends a transformation matrix to the interface, which commands the robot

- **Detector:** Captures raw data from the VR headset, including hand keypoints, button presses, and session commands.
- **Operator:** Processes and transforms the input data into robot-relevant commands, such as end-effector positions or joint targets.
- **Interface:** Sends the high-level actions to the robot's control system, ensuring smooth and stable execution.

This modular design allows flexible integration of new hardware and robots, while fault isolation ensures that if one module fails, the others remain operational.

The **data collection module** interfaces seamlessly with the LeRobot framework [3]. It records synchronized streams of observations, actions, and metadata into a standardized, hardware-agnostic format, enabling immediate downstream use for benchmarking and learning.

The **model training module** leverages LeRobot's training pipeline but enhances it with asynchronous inference. A dedicated control loop streams actions at a fixed rate, while a separate inference thread computes policy outputs as compute resources become available. This decoupled scheme preserves real-time performance, even under high computational load.

Overall, BEAVR combines modularity, real-time performance, and open data standards to provide a scalable platform for research in teleoperated, dexterous robot control.

IV. BEAVR

BEAVR is a modular teleoperation pipeline designed to connect VR-based hand tracking to robot actuation across heterogeneous robotic platforms. The system is hardware-agnostic and has been deployed on a range of embodiments, including a 16-DoF dexterous robotic hand, a 7-DoF anthropomorphic arm, and an in-house assembled 6-DoF RX-1 robotic arm. For the RX-1 platform, BEAVR integrates seamlessly with ROS-based components, enabling direct interfacing with ROS control stacks and facilitating compatibility with existing robotic software ecosystems. This flexibility allows BEAVR to support diverse experimental setups, from fine-grained hand

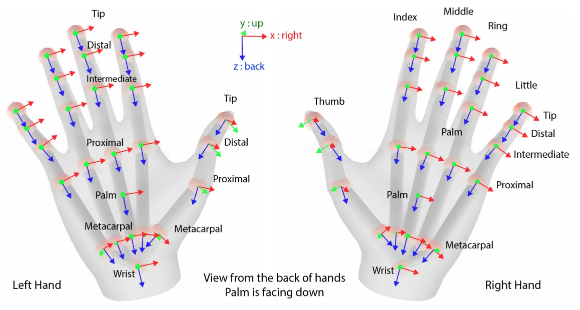

Fig. 2. Visualization of the 24 OpenXR hand keypoints captured by Meta Quest 3S at 90 Hz, providing 6-DoF wrist and finger landmarks as input for BEAVR's coordinate transformations and control solvers.

manipulation to whole-arm teleoperation, through dedicated modules for inverse kinematics (IK), temporal smoothing, and real-time control.

Hand tracking is performed using the Meta Quest 3S headset at 90 Hz, yielding 24 keypoints per hand compliant with the OpenXR standard (see Fig. 2). The resulting pose vectors serve as inputs to coordinate transformation modules that align the VR-tracked data with robot-specific frames, accommodating both dexterous hands and other end-effector types. This modular architecture enables integration with new embodiments by updating the relevant transformation matrices and solver targets without reengineering the full pipeline.

A. Coordinate Transformations

BEAVR computes the relative motion between the VR-tracked hand and the robot end-effector using homogeneous transformations. First, VR keypoints are translated to the wrist frame:

$$P_i' = P_i - P_0,$$

where P_0 is the wrist keypoint. An orthogonal hand basis is constructed using index, middle, and pinky landmarks, followed by Gram-Schmidt orthogonalization to ensure a stable frame.

To generalize across end-effectors, transformations are modularized:

$$H_{RT,RH} = H_{RI,RH} \cdot (H_{R,V}^{-1} H_{HT,HI} H_{R,V}),$$

where $H_{HT,HI}$ encodes the VR hand's relative motion, and $H_{RI,RH}$ the robot's initial pose. Scale-adjusted translations are applied separately depending on the embodiment (e.g., hand vs. gripper).

This modular design allows seamless integration of new robotic embodiments by updating only the transformation matrices and solver targets.

B. Inverse Kinematics Solver, Anti-Collision, and Temporal Smoothing

We implemented a multi-target inverse kinematics (IK) approach to optimize eight fingertip positions (two per finger, excluding the pinky) and compute joint configurations for dexterous manipulation. For each fingertip position p, we apply axis reflection and coordinate conversion from the Y-up VR frame to the Z-up robot frame:

$$p' = [-p_x,\ -p_z,\ p_y], \quad p'' = [p_y' \cdot s_f,\ -p_x' \cdot s_f,\ p_z' \cdot s_f],$$

where the scaling factor s_f is defined as:

$$s_f = \begin{cases} 1.8 & \text{for index, middle, and ring fingers,} \\ 1.7 & \text{for the thumb,} \end{cases}$$

following empirical tuning informed by prior estimates of robotic hand-to-human hand scaling ($\approx 1.6\times$) reported in LeapHand [6].

The IK problem is solved using a Damped Least Squares (DLS) formulation, leveraging the previous joint configuration as a seed to improve stability and convergence. A dedicated anti-collision routine ensures that all computed joint solutions respect mechanical and kinematic limits, particularly under high-density or contact-rich manipulation, by integrating real-time collision checks into the IK optimization loop.

To ensure smooth, low-latency control, we apply temporal filtering on joint angles and end-effector poses. Moving average filters are applied to keypoint trajectories, while complementary filters with quaternion SLERP [14] blending are used for pose states, mitigating jitter and producing stable control signals.

C. System Integration and Communication

Communication between the VR system, robot, and control modules is handled via ZMQ, combining publisher-subscriber channels for continuous streaming with request-reply patterns for control commands. The BEAVR system relies heavily on an efficient network module designed specifically for thread-safe messaging in real-time robotic teleoperation. **Threads** allow the central processing unit (CPU) to execute multiple tasks within the same process. When operating multiple robots, each with various interacting components a reliable communication mechanism that avoids data races, system crashes, and inconsistent behavior is crucial. To address this, we developed a dedicated network API built on top of ZMQ, providing the following key functionalities:

- **Publishers:** Components broadcasting messages to multiple subscribers.
- **Subscribers:** Components receiving specific messages from publishers.
- **Thread Management:** Ensures each thread safely manages its own ZMQ socket.
- **Reliable Delivery:** Guarantees message delivery through handshake mechanisms, addressing potential issues such as slow-joiners.

ZMQ is chosen for its ability to efficiently abstract common messaging patterns (PUB/SUB, REQ/REP, PUSH/PULL), facilitating lightweight and high-speed communication suited for real-time robotic applications. Additionally, ZMQ inherently supports thread safety by allowing individual threads to manage their own sockets. A detailed explanation of the network system may be found in Appendix A

A complementary Unity-based application supports real-time teleoperation, receiving keypoint data over TCP ports from a dedicated daemon process. This architecture ensures uninterrupted, high-frequency data transmission and seamless integration between human operators and robotic platforms.

The main algorithmic and architectural contributions of BEAVR are:

- Stable hand frame construction using multi-landmark bases with Gram-Schmidt orthogonalization.
- Explicit, modular coordinate system conversion (Y-up VR to Z-up robot) via robot-specific transformation matrices.
- Multi-target IK for dexterous robotic hands, incorporating anti-collision routines and temporal smoothing.
- Temporal filtering combining moving averages and complementary SLERP blending for smooth control.
- Flexible, modular pipeline design enabling adaptation to diverse robotic embodiments and control frameworks.

.

V. LeRobot

The integration with LeRobot provides a standardized dataset format, models, and tools for robotic hardware.

A. *LeRobot Dataset Format*

The LeRobotDataset format provides a simple yet flexible structure for robotics. It integrates with Hugging Face hub and PyTorch, allowing users to load datasets from the Hugging Face hub or a local directory. This encourages collaboration within the robotics community. Each indexed frame in a dataset includes PyTorch tensors representing observations and actions, facilitating direct usage for model training. A unique feature of the format is the ability to query temporally related frames using a delta timestamps parameter. This enables users to efficiently retrieve sequences of observations around a given frame index. Internally, the dataset is stored using widely adopted formats: observations and actions stored as Arrow/Parquet tables via the Hugging Face datasets library. Video files are compressed to MP4 files and metadata is managed using standard JSON formats. This approach ensures ease of use and extensibility for various robotics sensory inputs and state data, a practical feature for diverse robotics and learning scenarios.

B. *Real-time Data Streaming*

The existing LeRobot framework already contains a control and recording loop designed to handle various robotic platforms. Our teleoperation stack integrates directly into this loop, enhancing it with real-time data streaming capabilities. Specifically, this integration includes two critical functionalities:

1) **Capturing Observations:** The loop captures the current state of the robot and all associated sensory inputs, including real-time images from RGB and depth cameras, and joint positions.

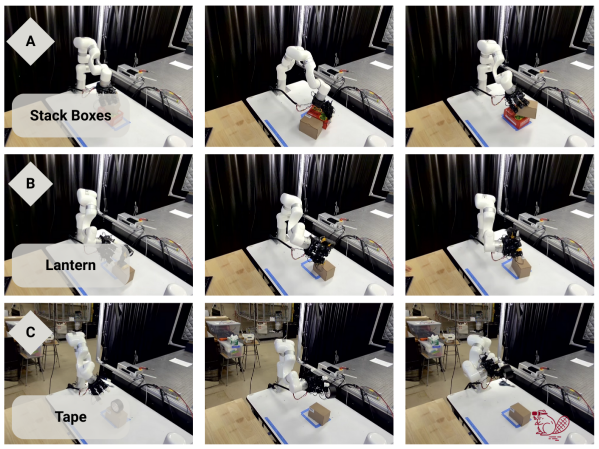

Fig. 3. Benchmark manipulation tasks in our experimental setup: (A) Stack Boxes, (B) Lantern, and (C) Tape. Each row shows a chronological progression (left → right).

2) **Sending Actions:** Actions are generated either through direct human teleoperation or policy inference, which are then published to the robotic system.

Both functions employ the network module, enabling low-latency and high-frequency data exchanges.

VI. Experimental Evaluation

Our experiments aim to answer the following questions.

1) What range of robotics tasks can BEAVR accomplish?
2) How well can the entire end-to-end policy training and evaluation pipeline perform on a given task?
3) How well does the BEAVR system scale to several robots?

We provide baseline comparisons against other VR teleoperation systems

A. *Experimental setup*

All experiments were conducted on an Alienware x16 R2 laptop equipped with an Intel Core Ultra 9 185H CPU and an NVIDIA GeForce RTX 4080 Max-Q GPU (12GB VRAM). The system runs Ubuntu 24.04.2 LTS with CUDA 12.8 and NVIDIA driver version 570.169. Figure 3 shows a sequence of images for each task, starting with the initial pose of the fixed-base 7-DoF tabletop manipulator XArm7 robot with a 16-DoF LEAP hand attachment to the end effector. All trials were conducted on a tabletop workspace. Two statically mounted RGB cameras (front-facing and overhead) captured synchronized streams at 480×640 resolution and 30 FPS. We conduct three separate experiments.

1) We perform three tasks demonstrating dexterous arm and hand use.
2) We evaluate the success rate of three visuomotor policies in a given task.
3) We analyze the performance of the entire system and its ability to scale by collecting latency, jitter, and frequency data.

TABLE I
STUDY COMPARING BEAVR EXPERT SUCCESS RATE AND MEDIAN COMPLETION TIME WITH OTHER BASELINES

Task	**Success Rate**				**Median completion time (s)**			
	Holo-Dex (n=5)	**Any-Teleop (n=5)**	**OpenTeach (n=5)**	**BEAVR (n=5)**	**Holo-Dex (n=5)**	**Any-Teleop (n=5)**	**OpenTeach (n=5)**	**BEAVR (n=5)**
Flip cube	1	1	1.0	1.0	6.58	13.71	2.85	13.32
Pour	-	-	0.8	0.8	-	-	14.83	28.92
Pick and Place	-	-	0.8	1.0	-	-	11.88	9.72

TABLE II
SUCCESS RATES ACROSS TASKS

Task	**Success rate**	**Avg. time (s)**
Tape task	6 / 10 (60%)	17.61
Lantern task	8 / 10 (80%)	21.61
Stack blocks	7 / 10 (70%)	25.89
Flip cube	10 / 10 (100%)	16.50
Pour	8 / 10 (80%)	84.67
Pick and place	10 / 10 (100%)	11.90

B. Task Evaluation

We evaluate our system on six representative manipulation tasks: manipulating adhesive tape (“Tape Task”), grasping and operating a lantern (“Lantern Task”), stacking blocks of varying sizes (“Stack Blocks”), flipping a cube (”flip cube”), pouring a cup into another cup (”pour”), and a pick and place task (”pick and place”). Each task is designed to highlight different aspects of dexterous, finger-level, and arm teleoperation. For each task, a series of 10 trials is performed using our system. We record success and failure according to task-specific completion criteria, enabling quantification of task performance. The results for each task can be seen in Table II. The completion criteria for each task are enumerated below. We also collect five additional samples of an expert completing three tasks to compare with a baseline from other teleoperation systems. The results are found in Table I.

1) **Tape task**: Insert the ring finger of the hand into the roll of tape, lift it and set it down on top of a box.
2) **Lantern task**: Grasp the lantern as if to use it for illumination and place it on top of the box.
3) **Stack blocks**: Organize two blocks into the target area by stacking them from smallest to largest (smallest on top then largest below).
4) **Flip cube**: Successfully flip a cube to another of its sides.
5) **Pour**: Grasp and tilt a paper cup as if to pour liquid into another paper cup.
6) **Pick and place**: Pick up an item and place it in a target area.

The task success rate and median completion time for the other systems were not collected by us [3], but serve as baselines for relative comparison. The ”flip cube” task for the other systems was conducted in simulation while we perform a similar task on a real robot, hence, a longer completion time. Our pour task was also modified to use a LEAP hand instead of a gripper and a cup instead of sprinkles, which also explains the increased completion time.

[3]OpenTeach numbers copied from Table IV of Iyer et al. 2024. No published numbers available for Holo-Dex and Any-Teleop for pour and pick and place tasks

TABLE III
SUCCESS RATES FOR POLICY EVALUATION ON PICKUP BOX TASK

Policy	**Success rate**	**Avg. time (s)**
ACT	10 / 10 (100%)	9.16
Diffusion	8 / 10 (80%)	23.88
SmolVLA	7 / 10 (70%)	33.84
Human operator	10 / 10 (100%)	12.08

TABLE IV
CONTROL CONFIGURATIONS USED TO EVALUATE NETWORK PERFORMANCE.

Configuration	**Robots**	**Frequency (Hz)**
1	XArm7, LEAP	30
2	XArm7, LEAP	90
3	2 XArm7, 2 LEAP	30

C. Policy Learning Evaluation

For autonomous control via policy rollouts, we collect a dataset with 50 expert demonstrations using the BEAVR system, and use this dataset to train the following visuomotor policies: Action-Chunking Transformers (ACT) [15], Action Diffusion [16], and a fine-tuned version of SmolVLA [17]. Policy performance is assessed by executing the trained policies on the task and recording successful completion rates. The completion criterion for the pickup box task is: grasp a box, lift it, and set it down in a target area. Table III shows the performance of three different policies, ACT, Diffusion, and SmolVLA trained on the same dataset.

D. Network Performance Evaluation

To evaluate how the BEAVR system scales with increasing control frequency and number of effectors, we measured latency and jitter across three control configurations. Each configuration represents a different combination of robot count and command frequency, as detailed in Table IV.

Table V summarizes the achieved control frequencies and measured jitter for each robot component across these configurations. The system maintains high timing fidelity, achieving

TABLE V
REAL-TIME PERFORMANCE METRICS FOR TELEOPERATION AT DIFFERENT COMMAND FREQUENCIES.

Configuration	Achieved Frequency (Hz)	Jitter (ms)
XArm7 (single-arm, 30 Hz)	29.93	0.90
LEAP (single-arm, 30 Hz)	29.69	0.19
XArm7 (bimanual, 30 Hz)	29.93	0.89
LEAP (bimanual, 30 Hz)	29.61	0.21
XArm7 (single-arm, 90 Hz)	99.18	0.75
LEAP (single-arm, 90 Hz)	97.22	0.13

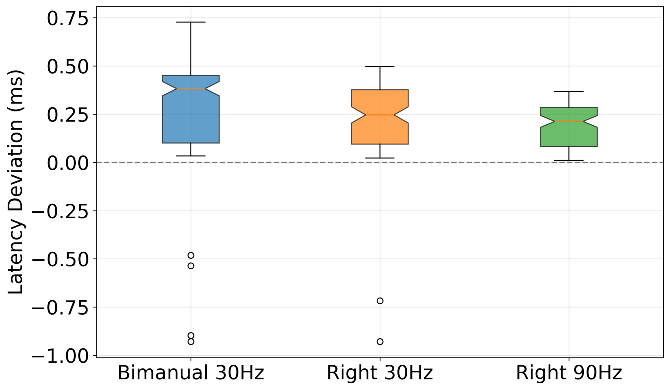

Fig. 4. Latency distribution combining XArm7 and LEAP hand control. This figure aggregates arm and hand statistics into a single distribution.

over 99% of the target frequency in all conditions with sub-millisecond jitter.

Latency and jitter were recorded over 60-second control episodes and analyzed for both arms and hands. The distributions of these metrics, aggregated across all configurations, are shown in Figures 5 and 4. Each figure combines the statistics for both the XArm7 and LEAP across configurations, providing an overall view of temporal performance stability.

We evaluate BEAVR's network performance by benchmarking it against published metrics from teleoperation systems using diverse network architectures. At a control frequency of 90 Hz, BEAVR achieves approximately 10 ms of latency and sub-millisecond jitter, outperforming typical Wi-Fi-based teleoperation systems (e.g., 57.4 ms round-trip latency and 2–4 ms jitter [18]) and VR-based interfaces such as Vicarios (40 ms one-way latency [19]).

At 30 Hz, BEAVR's latency increases to approximately 33 ms, which remains competitive, matching or surpassing other VR teleoperation platforms. While BEAVR does not yet reach the ultra-low latency of cutting-edge 5G URLLC solutions (0.8–2.0 ms latency, 1.1 ms jitter at the 99.9th percentile [20]) or wired LAN-based systems (5.7 ms round-trip latency, 1 ms jitter [18]), it demonstrates consistent, reliable timing control even with multiple effectors operating simultaneously.

Across all configurations, BEAVR achieved control frequencies within 99.2%–99.4% of target, with negligible degradation when scaling to four robots or increasing command rates to 90 Hz. Jitter remained low and consistent: under 0.90 ms for xArm7 and under 0.22 ms for LEAP hand in all cases. These values held steady even when doubling the number of controlled effectors. The highest-rate configuration (Config 2: 90 Hz) exceeded expectations, with xArm7 operating at 99.18 Hz and LEAP at 97.22 Hz, more than three times the base 30 Hz rate, while maintaining sub-millisecond jitter.

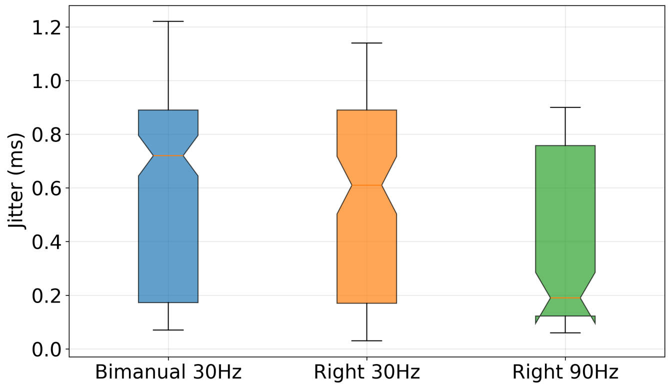

Fig. 5. Jitter distribution combining XArm7 and LEAP hand control. This figure combines both arm and hand statistics into a single distribution.

VII. DISCUSSION

By addressing the three experimental questions, several key findings about BEAVR's capabilities and performance are highlighted. First, BEAVR enables dexterous finger-level manipulation across diverse tasks, including stacking, tool use, and grasping. Although success rates varied across tasks, it is important to note that all datasets were collected in single takes. Consequently, failures arose partly from user skill and partly from the mechanical dexterity of the robots themselves, rather than from limitations of BEAVR. With improved low-level control algorithms or more dexterous hardware, the range of supported tasks is expected to expand significantly.

Second, BEAVR functions as an end-to-end framework for autonomous robotic control. Teleoperation, dataset collection, policy training, and rollout evaluation are integrated into a seamless pipeline. The policy experiments demonstrate that control policies trained within this framework (e.g., ACT, DiffusionPolicy, SmolVLA) achieve strong task success rates, underscoring BEAVR's value not merely as a teleoperation tool but also as a foundation for learning-based research.

Third, real-time performance evaluations demonstrate that BEAVR sustains reliable control across both single-arm and bimanual configurations at various command frequencies. The system consistently achieved near-target control rates (e.g., 29.93 Hz at 30 Hz, 99.18 Hz at 90 Hz) with low jitter, even under increased robot count and command frequency. Notably, higher-frequency control yielded lower jitter, which fell from 0.90 ms to 0.75 ms when we moved from 30 Hz

TABLE VI
COMPARISON OF TELEOPERATION SYSTEM PERFORMANCE METRICS

System	Control Frequency (Hz)	Latency (ms)	Jitter (ms)
BEAVR (90 Hz, **single-arm)**	97–-99	10.1 (one-way)	0.13–0.75
BEAVR (30 Hz, **single/bimanual)**	29–30	33.4–33.8 (one-way)	0.19–0.90
LAN-based teleop (Bray et al.)	30–60	5.7 (RTT)	1.0
Wi-Fi-based teleop (Bray et al.)	30–60	57.4 (RTT)	2.0–4.0
5G URLLC industrial (Reiher et al.)	100+	0.8–2.0	1.1 (99.9th %)
VR UR5 teleop (Vicarios)	30	40 (one-way)	–

to 90 Hz, indicating that the network layer and threading architecture efficiently handle real-time demands and may even improve under higher loads. Furthermore, the similarity in jitter between single-arm and bimanual setups suggests that bottlenecks are localized within local control loops rather than within the BEAVR architecture itself.

Together, these findings highlight two core strengths of the BEAVR system: its inherent flexibility in supporting heterogeneous robot platforms without sacrificing communication performance, and its robust scalability when increasing operational complexity. These properties position BEAVR as a promising open-source teleoperation platform for a broad range of research scenarios, from dexterous manipulation to multi-robot coordination, and as a valuable bridge between teleoperation and data-driven policy learning.

VIII. LIMITATIONS AND FUTURE WORK

While BEAVR demonstrates strong performance across diverse teleoperation tasks, several limitations remain that suggest future avenues for research and development.

Hardware limitations. One core challenge is the lack of affordable, high-performance robotic hands. The LEAP hand used in our experiments provides an accessible starting point, but its utility is currently constrained by factors such as its ability to execute precise, repeatable grasps like touching the thumb to any finger. Additionally, the physical bulk of the hand can obstruct VR keypoint tracking, particularly when the palm is oriented downward or during fist-like grasps.

System usability and extensibility. While BEAVR supports multiple robot embodiments, adding new hardware still requires manual configuration, calibration, and tuning. Simplifying these processes through auto-configuration tools, standardized robot templates, or GUI-based setup wizards could make the system more accessible to broader audiences.

Call for community contribution. As an open-source project, BEAVR depends on ongoing community engagement. We invite researchers and developers to contribute new robot modules, improve IK and retargeting components, simplify launch processes, and enhance documentation. Contributions of benchmark datasets, plugins (e.g., haptics or AR overlays), or alternate control strategies (e.g., shared autonomy) would expand the system's impact. A more diverse ecosystem of robots and tasks would also help establish common evaluation standards for VR-based teleoperation.

IX. CONCLUSION

BEAVR addresses long-standing challenges in robotic teleoperation of scalability, hardware flexibility, and ease of integration by providing an open-source, modular framework that unifies control, demonstration, and learning. Its plug-and-play operator interface, built on a component-based architecture, enables real-time streaming and control at up to 90 Hz across multiple heterogeneous embodiments, including RX1, xArm7, and LEAP hands. By natively integrating with Hugging Face's LeRobot ecosystem, BEAVR facilitates efficient, standardized dataset collection and seamless transitions from teleoperation to policy learning. The system supports an asynchronous "think-act" control loop that ensures uninterrupted operation, even under high computational loads.

BEAVR's architecture demonstrates excellent performance in both single-arm and bimanual configurations, maintaining low-latency communication (sub-35 ms RTT-equivalent) without requiring specialized networking infrastructure. As an open-source initiative, BEAVR aims to lower the barrier to entry for robotics research by democratizing access to high-quality teleoperation and policy learning tools.

X. ACKNOWLEDGEMENTS

This research was sponsored by the Department of the Air Force Artificial Intelligence Accelerator and was accomplished under Cooperative Agreement Number FA8750-19-2-1000.

Appendix A
Network Appendix

The BEAVR network module comprises several key components motivated by the following concepts. **Context**: A ZMQ context manages all sockets within a given process. This context coordinates the creation and management of sockets, I/O threads, and shared buffers, ensuring efficient resource handling and preventing interference across processes. **Sockets** are a way of connecting two machines so they can communicate. Each ZMQ socket is initialized through our API with appropriate configurations, such as high-water mark settings to limit buffering (a region of memory used to temporarily store data) and discard older messages if necessary. Subscribers run dedicated threads, continuously polling for new messages on their subscribed topics. The API also provides convenient wrappers to seamlessly serialize and deserialize Python objects alongside topics. A singleton publisher manager ensures that each unique host-port combination has exactly one dedicated publisher thread. These threads independently handle serialization, queuing, and message broadcasting, effectively isolating network operations from the main application logic. This design reduces the risk of race conditions.

To mitigate the slow-joiner issue inherent in PUB/SUB architectures, a handshake coordinator confirms the successful delivery of critical messages to subscribers. This guarantees subscribers are synchronized and ready before important messages are sent, ensuring reliable and orderly communication.

When teleoperation initiates, the system spawns individual processes for its primary components (e.g., detector, operator, interface). Each process independently creates its own sockets but shares a global ZMQ context. This setup enables immediate PUB/SUB communication across processes, promoting modularity and scalability within the teleoperation system.

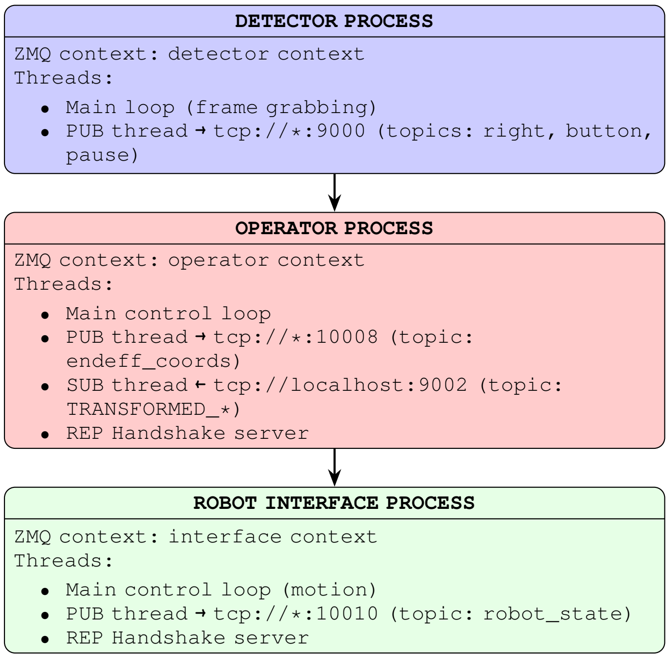
DETECTOR PROCESS
ZMQ context: detector context
Threads:
• Main loop (frame grabbing)
• PUB thread → tcp://*:9000 (topics: right, button, pause)
OPERATOR PROCESS
ZMQ context: operator context
Threads:
• Main control loop
• PUB thread → tcp://*:10008 (topic: endeff_coords)
• SUB thread ← tcp://localhost:9002 (topic: TRANSFORMED_*)
• REP Handshake server
ROBOT INTERFACE PROCESS
ZMQ context: interface context
Threads:
• Main control loop (motion)
• PUB thread → tcp://*:10010 (topic: robot_state)
• REP Handshake server

Fig. 6. Network architecture. Each component (detector, operator, interface) starts one process. Within that process, for every PUB/SUB object, we create a thread, which owns a socket. Topics support multiple logical channels (pose, buttons, commands) to flow over a single physical port

Appendix B
Dataset Access Links

We provide public access to the datasets collected in our experiments to encourage reproducibility and further research. The following Hugging Face URLs host the datasets for each task:

- **Tape Insert Dataset:** https://huggingface.co/datasets/arclabmit/lx7r_tape_insert_dataset
- **Lantern Grasp Dataset:** https://huggingface.co/datasets/arclabmit/lx7r_lantern_grasp_dataset
- **Stack Blocks Dataset:** https://huggingface.co/datasets/arclabmit/lx7r_stack_blocks_dataset
- **Flip Cube Dataset:** https://huggingface.co/datasets/arclabmit/lx7r_flip_cube_dataset
- **Pour Dataset:** https://huggingface.co/datasets/arclabmit/lx7r_pour_dataset

References

[1] A. Iyer, Z. Peng, Y. Dai, I. Guzey, S. Haldar, S. Chintala, and L. Pinto, "Open teach: A versatile teleoperation system for robotic manipulation," 2024. [Online]. Available: https://arxiv.org/abs/2403.07870

[2] R. Ding, Y. Qin, J. Zhu, C. Jia, S. Yang, R. Yang, X. Qi, and X. Wang, "Bunny-visionpro: Real-time bimanual dexterous teleoperation for imitation learning," 2024. [Online]. Available: https://arxiv.org/abs/2407.03162

[3] R. Cadene, S. Alibert, A. Soare, Q. Gallouedec, A. Zouitine, S. Palma, P. Kooijmans, M. Aractingi, M. Shukor, D. Aubakirova, M. Russi, F. Capuano, C. Pascale, J. Choghari, J. Moss, and T. Wolf, "Lerobot: State-of-the-art machine learning for real-world robotics in pytorch," https://github.com/huggingface/lerobot, 2024.

[4] "Feetech sts3215-c018 smart servo (30 kg·cm, 12 v)," https://www.robotshop.com/products/feetech-12v-30kgcm-magnetic-encoding-servo-sts3215, 2025, list price $34, accessed 15 Jul 2025.

[5] "Meta quest 3 s – product page," https://www.meta.com/quest/quest-3s/, 2025, price $299.99 USD, accessed 15 Jul 2025.

[6] K. Shaw, A. Agarwal, and D. Pathak, "Leap hand: Low-cost, efficient, and anthropomorphic hand for robot learning," 2023. [Online]. Available: https://arxiv.org/abs/2309.06440

[7] "Overture pla filament 1.75 mm, 1 kg," https://www.amazon.com/OVERTURE-Filament-Consumables-Dimensional-Accuracy/dp/B07ZJPCQBN, typical price $15–$20 per kg, accessed 15 Jul 2025.

[8] J. Guo, J. Luo, Z. Wei, Y. Hou, Z. Xu, X. Lin, C. Gao, and L. Shao, "Telepreview: A user-friendly teleoperation system with virtual arm assistance for enhanced effectiveness," 2025. [Online]. Available: https://arxiv.org/abs/2412.13548

[9] H. Zhang, S. Hu, Z. Yuan, and H. Xu, "Doglove: Dexterous manipulation with a low-cost open-source haptic force feedback glove," 2025. [Online]. Available: https://arxiv.org/abs/2502.07730

[10] B. Fang, D. Guo, F. Sun, and Y. Wu, "A robotic hand-arm teleoperation system using human arm/hand with a novel data glove," 12 2015, pp. 2483–2488.

[11] A. Imdieke and K. Desingh, "Spark-remote: A cost-effective system for remote bimanual robot teleoperation," 2025. [Online]. Available: https://arxiv.org/abs/2504.05488

[12] J. van Haastregt, M. C. Welle, Y. Zhang, and D. Kragic, “Puppeteer your robot: Augmented reality leader-follower teleoperation,” 2024. [Online]. Available: https://arxiv.org/abs/2407.11741

[13] Red Rabbit Robotics, “RX-1: Open-Source Humanoid Robot Platform,” https://www.redrabbitrobotics.ai/, 2024, accessed: 2025-07-14.

[14] K. Shoemake, “Animating rotation with quaternion curves,” *SIGGRAPH Comput. Graph.*, vol. 19, no. 3, p. 245–254, Jul. 1985. [Online]. Available: https://doi.org/10.1145/325165.325242

[15] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, “Learning fine-grained bimanual manipulation with low-cost hardware,” 2023. [Online]. Available: https://arxiv.org/abs/2304.13705

[16] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, “Diffusion policy: Visuomotor policy learning via action diffusion,” *The International Journal of Robotics Research*, 2024.

[17] M. Shukor, D. Aubakirova, F. Capuano, P. Kooijmans, S. Palma, A. Zouitine, M. Aractingi, C. Pascal, M. Russi, A. Marafioti, S. Alibert, M. Cord, T. Wolf, and R. Cadene, “Smolvla: A vision-language-action model for affordable and efficient robotics,” 2025. [Online]. Available: https://arxiv.org/abs/2506.01844

[18] N. Bray, M. Boeding, M. Hempel *et al.*, “A latency composition analysis for telerobotic performance insights across various network scenarios,” *Future Internet*, vol. 16, no. 12, p. 12, 2024. [Online]. Available: https://doi.org/10.3390/fi16120457

[19] A. Naceri and et al., “Vicarios: A virtual reality interface for teleoperation with a 7-dof robot arm,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 9357–9363.

[20] L. Reiher, B. Lampe, T. Woopen, R. van Kempen, T. Beemelmanns, and L. Eckstein, “Enabling connectivity for automated mobility: A novel mqtt-based interface evaluated in a 5g case study on edge-cloud lidar object detection,” *arXiv preprint arXiv:2209.03630*, Sep 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2209.03630