OpenPyRo-A1: An Open Python-based Low-Cost Bimanual Robot for Embodied AI

Helong Huang[1], Christopher E. Mower[1] Guowei Huang[1], Sarthak Das[1], Magnus Dierking[1], Guangyuan Luo[1] , Kai Tan[1], Xi Chen[1], Yehai Yang[1], Yingbing Chen[1], Yiming Zeng[1], Yinchuan Li[1], Zhanpeng Zhang[1], Shuang Wu[1], Yingxue Zhang[1], Weichao Qiu[1], Tongtong Cao[1], Mian Qin [1], Sajjad Pakdamansavoji [1], Yuecheng Liu[1], Yuzheng Zhuang[1], Guangjian Tian[1], Jianye Hao[1], Jun Wang[2], Haitham Bou-Ammar[1,2], Xingyue Quan[1]

***Abstract*—Many real-world tasks, such as assembly, cooking, and object handovers, require bi-manual coordination. Learning such skills via imitation remains challenging due to dataset scarcity, mainly caused by the high cost of bi-manual robotic platforms and barriers to entry in robotics software. To address these challenges, we introduce (1) OpenPyRo-A1, a low-cost, bi-manual humanoid robot priced at approximately $14K. OpenPyRo-A1 achieves 0.2 mm repeatability and supports a 5 kg payload per arm, and (2) a Python-first distributed control framework for seamless teleoperation, data collection, and policy deployment, designed for ease of use; moreover, the code-base is installable via `pip`. We conducted imitation learning experiments in both simulation and the real world, integrating the robot with perception models, motion planning, and a large language model. The results demonstrate that OpenPyRo-A1 is a stable, user-friendly, and high-precision dual-arm platform. We expect that the OpenPyRo-A1 hardware, control system, and curated dataset of bi-manual manipulation episodes will advance affordable and scalable dual-arm robotics.**

***Index Terms*—Distributed robot systems, bimanual manipulation, low-cost dual-arm platform, teleoperation.**

I. Introduction

AI-BASED control in robotics has seen many remarkable advancements in recent years [1]–[5]. Particularly, methods for imitation learning [6], [7] have generated sophisticated policies by learning from data collected via human demonstrations. While successful, a bottleneck is the requirement of large datasets to learn effective and scalable controllers.

Unlike natural language processing and computer vision, which have access to vast data sources from the internet, robotics data is limited [8] due to the high hardware cost and the complexity of real-world interactions. To tackle these challenges, large-scale data collection initiatives have been launched [9], and scaling data collection has emerged as an increasingly prominent research direction [7].

While a growing body of data is becoming available in robotics, most of it remains focused on single-arm systems. Although single-arm robots can handle various tasks, many tasks can be performed far more efficiently with a bi-manual arm setup, e.g., in furniture assembly, which may require the robot to steadily hold a part in place with one arm while drilling with the other [10]–[12], or in cooking, where tasks such as dicing vegetables requires one arm to stabilize the ingredient while the other performs precise cutting [13].

Unfortunately, the problem of limited data resources is even more pronounced in the bi-manual setting, as there is a general lack of data collected for dual-arm systems. Even in the most prominent dataset, OpenX-Embodiment [9], only 3 out of the 22 embodiments (PR2 and xArm bi-manual) are bi-manual, accounting for just 4 out of the 60 datasets. We argue that the high cost of bimanual robot is a key reason for the scarcity of data. Although bimanual robots are used in research, their high price and increased complexity limit widespread adoption.

In response, we developed the OpenPyRo-A1, a low-cost, bi-manual humanoid robot. As illustrated in Fig. 1, our design integrates both hardware and a distributed control system, developed from scratch. OpenPyRo-A1 is designed to be affordable, easy to repair, and scalable for future upgrades. It features two 7-DoF arms with camera sensors, a 2-DoF waist, a head with two additional cameras, and custom-built grippers. According to Table I, the total hardware cost is approximately **$14K**, with a repeatability of less than 0.2 mm and a payload capacity of 5 kg per arm, offering 5× higher precision and 6.5× greater payload than systems such as ALOHA [14]. The cost OpenPyRo-A1 is approximately 1.4× lower than ALOHA, and 2× lower than non-hobbyist platforms like Reachy 1 and 5× lower than Reachy 2.

Our modular software framework follows a Python-first design. It leverages the Lightweight Communications and Marshalling (LCM) [15] library for communication to minimize maintenance and reduce complexity. While the Robot Operating System (ROS) remains the dominant robotics framework, its complexity often demands expert-level knowledge to develop and maintain the robot interface (e.g., PAMY2, Dual UR3, TidyBot++). In contrast, our system simplifies dependency management while allowing seamless interchange of sensors, teleoperation interfaces, actuators, and communication protocols without extensive reconfiguration. It bridges the gap between ROS-based and C++-centric systems through a lightweight, modular Python-first architecture. Additionally, due to architectural similarities between LCM and ROS, users can add translation nodes to reuse existing ROS drivers while benefiting from our system.

In this paper, we focus on data collection for bimanual manipulation via teleoperation. While kinesthetic teaching [16]

(Helong Huang, Christopher E. Mower, and Guowei Huang contributed equally. Haitham Bou-Ammar and Xingyue Quan also contributed equally.) (Corresponding author: Helong Huang.)

[1]All authors are with Noah's Ark Lab, Huawei (e-mail: {lastnamefirstname}@huawei.com).

[2]Jun Wang and Haitham Bou-Ammar are also with the University College London.

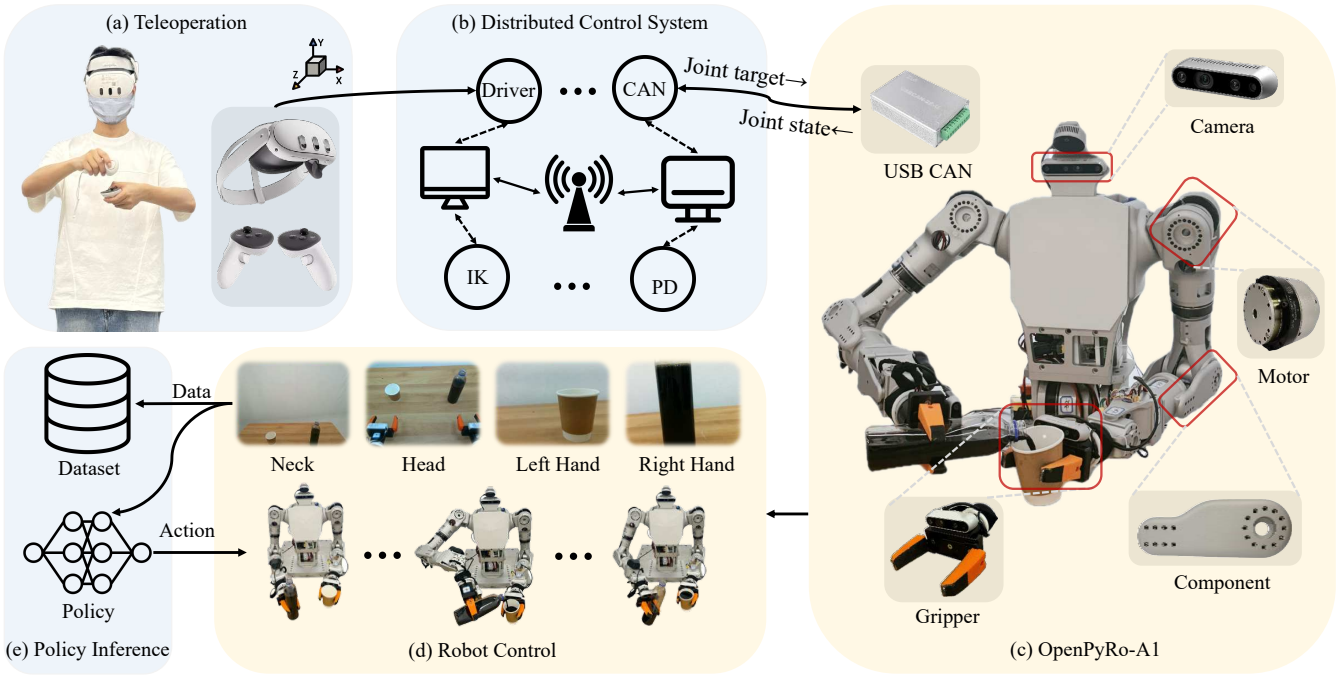


Fig. 1. **Overview of OpenPyRo-A1**. (a) Teleoperation: a VR system is used to transmit positional information to the control system, enabling real-time teleoperation of the robot. (b) Distributed Control System: the control system follows a publisher-subscriber communication model, onboard and external devices communicate via a single Ethernet connection, enabling modular control, perception, and data collection. (c) OpenPyRo-A1: A low-cost dual-arm robot with 7 degrees of freedom per arm and a 2-DOF waist capable of rotating left–right and moving up–down. The system integrates four RealSense cameras on the neck (D455), head (L515), left arm (D435i), and right arm (D435i) to provide visual feedback. (d) Robot Control: Collect data by remotely operating the robot or control it by policy reasoning. (e) Policy Inference: The trained policy is executed to perform reasoning and generate actions.

is an alternative approach, it is not explored in this work due to safety concerns and practical limitations with dual-arm systems. Teleoperation, by contrast, enables safe, immersive, and scalable data collection using a VR interface, although intuitive control through devices such as gamepads or haptic controllers [17], [18] remains challenging.

In order to maintain low cost for our system, we use a VR headset for an immersive teleoperation interface of our bi-manual system. To generate data, we designed a simulation pipeline with domain randomization and conducted sim-to-real experiments. We further evaluated both hardware and software on eight dual-arm tasks that demanded long-horizon planning and precise control. To this end, we trained policies using Action Chunking with Transformers (ACT) [14] and fine-tuned the Vision-Language-Action (VLA) model, π_0 [19], before deploying them to complete the various tasks.

To evaluate the scalability and generalization of OpenPyRo-A1, we build modular skill libraries that combines perception, motion planning, and imitation learning for tasks such as pouring, folding, and object manipulation. We integrated OpenPyRo-A1 with an Embodied AI agent that uses a vision-language model InternVL [23] for scene understanding and a large language model (DeepSeek) for instruction interpretation and skill selection, enabling grounded reasoning and execution of diverse tasks from natural language and visual inputs.

The following is a summary of our main contributions:

- We developed OpenPyRo-A1, a low-cost, dual-arm humanoid robot.
- A Python-first distributed control framework, including libraries for data collection via VR teleoperation and a simulation data collection pipeline.
- A dataset of over 560 bi-manual manipulation episodes, integrated with machine learning algorithms for skill learning from vision and low-level positional data.
- An agentic framework for high-level planning using large language and vision-language models.

TABLE I
COMPARISON OF ROBOT PLATFORMS. NOTE, WE USE ∼ TO INDICATE THE VALUE IS AN APPROXIMATE COST FOR THE ENTIRE SYSTEM AND REP. IS THE POSITION REPEATABILITY.

Platform	Arm DoF	Dual Arm	Sim Support	Rep. (mm)	Payload (kg)	Cost ($K)
PAMY2 [20]	4	✗	✗	-	-	16.3
DROID [21]	7	✗	✗	**0.1**	3	∼ 27
TidyBot++ [22]	7	✗	✗	0.15	4	∼ 38
ALOHA [14]	12	✓	✓	1	0.75	≤ 20
Dual UR3	12	✓	✗	**0.1**	3	66
Reachy 2	14	✓	✗	-	3	70
OpenPyRo-A1	**14**	✓	✓	0.2	**5**	**13.9**

II. RELATED WORK

The availability of data for single-arm manipulation tasks has led to the development of numerous successful imitation learning methods [14], [24]. However, bimanual manipulation

has not seen the same level of progress due to the lack of data and total system cost.

To bring down the cost of robotics there have been numerous open-source efforts targeting single-arm systems [22], [25], and even full humanoids [26], [27], mainly through platforms like the RoboCup tournament [28], [29]. However there are only a few commercially available upper torso robots and even fewer open source models [30], [31]. Due to easier access and broader commercial availability, research labs use humanoid robots for static bimanual tasks.

Using humanoids for these tasks tends to be unnecessarily complex for the specific purpose of object manipulation, often requiring more room and hardware to suspend the robot. These complex robots make it difficult for smaller labs to do bimanual manipulation, forcing them to turn to other solutions.

One solution mounts two commercial single-arm robots on a shared torso [32]. For example, Bi-VLA [33] and Shake [34] use dual UR3 arms for kitchen or liquid-mixing tasks, while the APEX system [35] employs dual Franka arms for precision vector alignment. While effective, these DIY setups often void robot warranties and remain costly, e.g., two Franka Emika Panda or Universal arms.

Few open-source fixed-base upper torso platforms remain expensive. For example, the Pollen robot provides Python-based teleoperation but costs 33K USD to over 50K USD and relies on ROS, while cheaper options [30], [31] or hobbyist-built robots often lack payload or accuracy. Consequently, they do not provide a viable alternative for bimanual tasks, highlighting a clear gap for a low-cost, reliable dual-arm robot suitable for data collection and research usage.

III. LOW-COST HARDWARE SYSTEM

Our hardware balances cost, performance, and flexibility, featuring two arms, a torso, a head, and custom grippers. Combining 3D-printed components for rapid prototyping with an aluminum alloy frame, it ensures precision, modularity, strength. We follow three key design principles:

Low-cost: The robotic system is designed to be cost-effective without compromising performance. It uses 3D-printed parts for non-load-bearing components and Computer Numerical Control (CNC) machined aluminum (also 3D-printable) for structural elements, balancing durability and affordability. Off-the-shelf actuators, sensors, and electronics minimize custom hardware and simplify procurement. As shown in Table II, the system supports easy assembly with minimal tools, making it accessible to both researchers and industrial users.

Ease of repair: A modular architecture ensures individual components can be easily replaced or upgraded without extensive disassembly. Key features include easy-access panels (in the torso), standardized fasteners, and color-coded wiring for simplified troubleshooting. All electrical and mechanical connections are quick-release or plug-and-play. The gripper and end-effector mounting system use universal attachment points, allowing users to swap parts without modifications.

Scalability: The system is designed for easy upgrades as new technologies emerge. Its electronics and software support modular firmware updates, and pre-drilled expansion points allow adding sensors, cameras, or actuators without major modifications. The power and communication architecture includes extra capacity for seamless integration of new components. The base structure also supports mobile chassis, enabling extension to mobile manipulation.

TABLE II
SPECIFICATIONS AND COST OF OPENPYRO-A1.

Category	Parameter/Item	Value/Price ($)
Specifications	Max Gripper Width	9.5 cm
	Payload (One Arm)	5 kg
	Camera Number	4
	Power Voltage	48 V
	Repeatability	0.2 mm
	Weight	46 kg
	Reach	750 mm
	DoF	16
Cost	Component Print	2,912
	Four Cameras	1,711
	Accessories	118
	USB CAN	319
	Motor	8,552
	Power	343
	Total Cost	**13,955**

A. Component

To reduce cost and simplify assembly, all components were designed with simple structures (Fig. 2). The torso uses two support plates forming an internal space for electronics, with an upper-wide, lower-narrow profile to maximize arm mobility. Each 7-DoF arm features protective structures around motors from shoulder to the wrist. While compatible with various materials, components are CNC-machined from aluminum alloy to ensure strength and high payloads.

B. Motor and Gripper

To enable smooth and cost-effective control, the robot uses low-cost EYOU motors. As shown in Fig. 2, two high-torque motors at the waist handle torso loads, allowing up-down and left-right motion. From shoulder to wrist, progressively smaller motors balance torque, size, and weight for efficient and strong actuation. The custom 0.4 kg parallel gripper operates at 20 Hz, with 9.5 cm wide, 8 cm long TPU95A fingers for flexibility and durability, using the same actuators and communication protocol as the arms for seamless integration.

C. CAN System

The system employs USB-to-CAN interfaces to facilitate communication between the onboard computer and the actuators. To achieve real-time control, motors are driven using the Transmit Process Data Object (TPDO) protocol of CANopen, which allows for efficient transmission of target commands and state feedback.

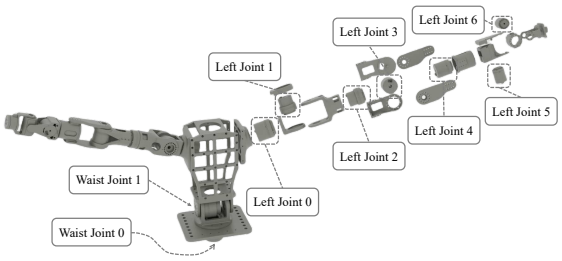


Fig. 2. Exploded view of the left arm of OpenPyRo-A1.

IV. DISTRIBUTED CONTROL SYSTEM

To complement OpenPyRo-A1, we introduce a Python-first distributed control system for accessible robot development. As shown in Fig. 1(b), it uses a publisher-subscriber architecture to simplify control, perception, and data collection. Installation requires only a single `pip install`, and onboard and external devices connect via Ethernet for rapid development.

A. Communication and Robot Interface

Efficient communication is critical for real-time robotics, and we use the LCM library for its support of custom message types, low latency, and high-frequency data transmission—ideal for distributed systems. At the core, a structured `BaseNode` class simplifies building publishers and subscribers with configurable execution rates. A high-level robot interface abstracts low-level communication, allowing users to (i) access real-time robot state, (ii) execute joint commands (streaming or trajectory-based), and (iii) perform task-space control for complex manipulation.

B. Kinematics and Control

To map task-space goals to joint commands, we implement forward kinematics using Kinpy [36], extended with CasADi and Spatial-CasADi for array-based computation and spatial transformations. Inverse kinematics is formulated as a constrained *nonlinear optimization* problem:

$$\min_{\mathbf{q}} \; w_p \|\mathbf{p}_g - \mathbf{p}(\mathbf{q})\|^2 + w_r d_R^2(\mathbf{r}_g, \mathbf{r}(\mathbf{q})) + w_v \left\| \frac{\mathbf{q} - \mathbf{q}_t}{\Delta t} \right\|^2 \quad (1)$$

Here, $\mathbf{p}_g$ and $\mathbf{r}_g$ are the desired end-effector position and orientation, $\mathbf{p}(\mathbf{q})$ and $\mathbf{r}(\mathbf{q})$ are obtained via forward kinematics, and d_R measures rotational error. The weights w_p, w_r, and w_v balance position, orientation and smoothness. Joint and velocity limits are implicitly enforced to ensure safety.

Motor control is performed through a custom CAN bus interface, built on Python's `ctypes` library. A Proportional–Derivative (PD) controller regulates joint positions, enforces safety limits, and ensures smooth motion, publishing joint targets and states at 150 Hz. The system supports both position and velocity control, and we plan to upgrade the CAN system to CAN-FD, which allows communication frequencies of 1–5 MHz, enabling torque control and other force-based methods in the future.

C. Teleoperation and Data Collection

We provide immersive and real-time teleoperation with Meta Quest 3 headset, Additionally, our system supports alternative control methods, including PlayStation 4 controllers and keyboard-based operation. The framework streamlines data collection with LCM-based logging, enabling recording of all network messages. A conversion script allows exporting data into HDF5 or LeRobot format for further analysis.

D. Safety

Safety is a core concern in our system, addressed through multiple safeguards. A dedicated safety node enforces both joint-space and task-space limits. Joint positions $\mathbf{q}$ are constrained within predefined minimum and maximum angles $q_i^{\min}$ and $q_i^{\max}$ (set in a configuration file), and commanded positions are clipped to these bounds to ensure safe operation. Task-space safety constrains the end-effector position $\mathbf{x}$ within predefined workspace boundaries. If a commanded position exceeds these bounds, it is adjusted and the corresponding joint positions are recomputed via inverse kinematics. Moreover, an emergency stop (E-stop) button was also integrated to immediately halt all motion in critical situations. These mechanisms ensure safe operation of the robot under both normal and extreme conditions.

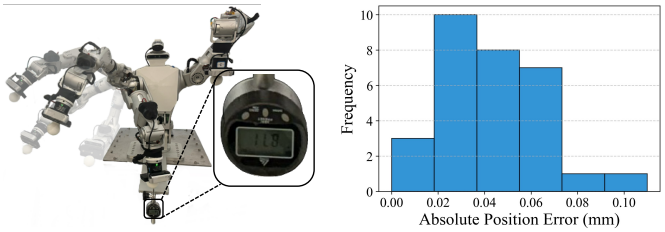
V. EXPERIMENT

To evaluate the performance and utility of OpenPyRo-A1, we conducted a series of experiments designed to answer the following key questions:

- **Q1:** Is OpenPyRo-A1 capable of stable and repeatable motion, providing a reliable physical platform for manipulation tasks?
- **Q2:** Can the system provide low-latency, accurately tracked, and safety-constrained teleoperation suitable for high-quality demonstration collection?
- **Q3:** Can the robot learn and autonomously execute precise, dual-arm tasks with high success rates?
- **Q4:** Does OpenPyRo-A1 support the integration of perception model, motion planning, and imitation learning components for building Embodied AI systems?

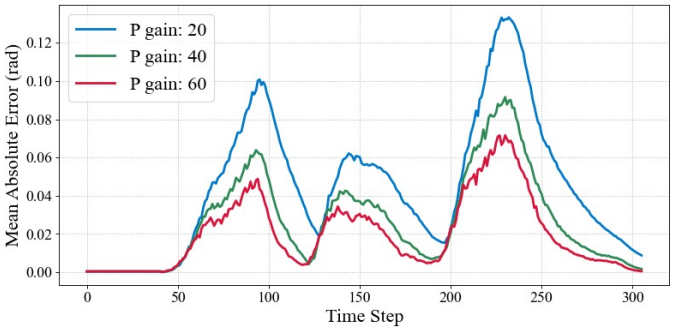
A. Positional Repeatability

To evaluate the repeatability of OpenPyRo-A1, we designed a measurement setup using a dial indicator (accuracy 0.01 mm) fixed in front of the robot, as shown in Fig. 3 (a). Since the gripper is made of elastic material, a hard ball was added to the end-effector to reduce deformation-induced error during contact. The end-effector was commanded to push against the micrometer, and the initial reading was recorded as 8.71 mm. We repeated the right arm motion 30 times under commands and recorded the micrometer reading after each trial. As shown in Fig. 3 (b), The position error is calculated as the deviation from the initial value. The results indicate a repeatability error of less than 0.2 mm. As summarized in Table I, OpenPyRo-A1 achieves higher positional repeatability than platforms such as ALOHA (1 mm), while maintaining significantly lower cost, demonstrating precise and consistent motion.

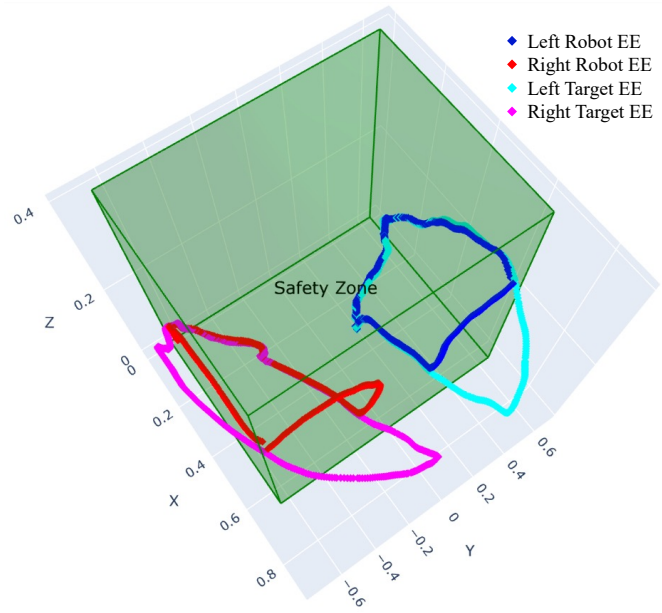


(a) Touch Micrometer with End-effector Ball (b) Histogram of Repeatability Error

Fig. 3. **Repeatability Evaluation.** (a) Experimental setup with a fixed micrometer and a hard ball attached to the end-effector. (b) Histogram of repeatability error over 30 repeated trials.



(a) Mean Absolute Error of Joint Tracking



(b) Executed vs Target End-Effector Positions with Safety Area

Fig. 4. **Teleoperation Control and Safety Evaluation.** (a) MAE of joint tracking under different proportional gains. (b) The system prevents the robot from leaving the defined boundary.

B. Teleoperation Control and Safety Evaluation

Data-driven learning relies on high-quality demonstrations, which require real-time communication, stable operation, and safety. The following experiments evaluate these aspects, including communication latency, controller tracking performance, and end-effector safety constraints.

1) Camera and Motor Communication Latency: We evaluated latency for four RGB-D cameras, which carry the largest data volume in teleoperation system. Two computers, synchronized via Precision Time Protocol (PTP) and connected through Ethernet, performed the evaluation: one published timestamped camera data while the other subscribed and measured latency. Each camera collected 1,000 samples, with an average latency below 5 ms. Motor command latency was also measured under the same setup, showing less than 1 ms for communication and an average 5.23 ms from command issuance to motor execution. In comparison, the commercial system Reachy 2 reports a hand-to-gripper latency of 74 ms, while our system achieves below 10 ms, confirming our system provides low-latency communication.

2) Controller Tracking: We design a dual-arm task where both arms simultaneously place fruits on a plate. We evaluate PD controller tracking by varying the proportional gain (20, 40, 60) during teleoperation, measuring the mean absolute error (MAE) between commanded and actual joint positions across all joints. As shown in Fig. 4(a), higher P gains reduce MAE, confirming accurate, responsive teleoperation. Our system further allows adjustment of tracking behavior to meet the precision needs of different tasks, supporting flexible future upgrades.

3) End-effector Safety: The teleoperation system defines end-effector safety regions through configuration files. The region was defined as $x \in [0, 0.6], y \in [-0.6, 0.6], z \in [0, 0.4]$ in the experiment. Target and actual end-effector poses were recorded during teleoperation, with the green area indicating the safety zone. As shown in Fig. 4(b), when the target exceeded the region, the system constrained the robot's motion to ensure safe operation. Unlike platforms such as ALOHA [14] and TidyBot++ [22], which lack explicit teleoperation safety constraints. Future extensions may include more complex constraints, such as self-collision avoidance or dynamic boundaries via configuration updates.

C. Simulation and Real-World Imitation Learning

To systematically evaluate our robot's manipulation capabilities, we adopt a two-stage validation strategy. First, we design a simulation data collection pipeline and use the collected data for imitation learning to verify the robot's design, such as its kinematic feasibility. After successful imitation learning in the simulation, we then move to real-world imitation learning experiments involving precise, dual-arm coordination and long-horizon manipulation.

1) Simulation: We use ManiSkill3 [37] for simulated experiments on three tasks: placing a bowl on a rack, inserting a fork on a rack vertically, and loading a plate. To improve robustness, initial object poses are randomized by ± 5 cm translation and $\pm 10^{\circ}$ rotation. To improve simulation to real (sim2real) transfer, we apply domain randomization on lighting, tabletop textures and object colors (Fig. 5 (a)). Using `MPlib` for motion planning, we collect 1,000 demonstrations per task to train a VLA model (π_0 [19]) and perform 25 sim and 10 real-world inference trials per task following ALOHA [14].

As shown in Table III, without domain randomization, simulation achieves high success: 98% for the bowl task and 85% for the plate task, the lower success is due to the difficulty of grasping thin objects. This demonstrates the feasibility of the hardware design and confirms the platform's suitability.

TABLE III
SUCCESS RATE (%) FOR IMITATION LEARNING (SIM & SIM2REAL).

Condition	Bowl	Fork	Plate
Sim w/o domain rand	98	90	85
Sim w/ domain rand	64	72	64
Real (Sim-to-Real)	50	60	40

As shown in rows two and three of Table III, the tasks were successfully completed on the real robot via sim2real transfer, reaching a maximum success rate of 60%, although simulation performance decreased after applying domain randomization. This highlights the ability to train deployable policies without real-world data. Failure cases were caused by physical discrepancies, such as grippers failing under low friction or forks bouncing in reality. Future work will focus on refining simulation parameters and enhancing the data collection pipeline for more large-scale real-robot experiments.

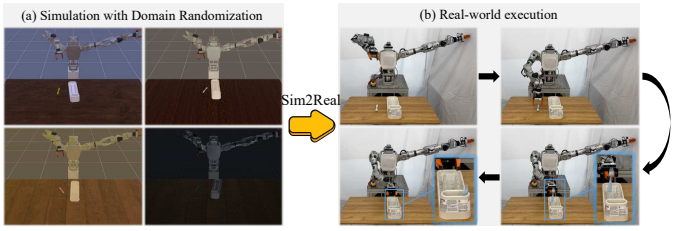


Fig. 5. **Sim-to-Real Transfer.** (a) Domain randomization including variations in lighting, tabletop textures and object appearance. (b) Snapshot of transferring from simulation to real-world execution.

2) Real-World Imitation Learning: To further evaluate hardware performance, software stability, and the quality of collected data, we design eight dual-arm tasks that reflect common challenges in real-world manipulation. These tasks rely on fine motor skills, involving precise coordination and synchronization between the two arms [14]. **(1) Pick Lemon and Apple:** The left arm picks an apple, the right picks a lemon, both are placed onto a plate simultaneously. **(2) Play Music Two Tigers:** The left arm plays *do* and *re*, the right arm plays *mi* and *do*. **(3) Move Box:** Both arms move a box to the table center. **(4) Cut Corn:** The left arm holds corn steady while the right arm cuts. **(5) Pour Rice:** The left arm picks a bowl, the right arm pours rice from cup. **(6) Clean Table:** The left arm holds a dustpan while the right arm sweeps objects in. **(7) Open Drawer and Pick:** The left arm opens a drawer, the right arm picks a vegetable. **(8) Stack Cups:** Both arms pick up cups, the right stacks onto the left arm.

We collect 70 demonstrations for each task using teleoperation. Policies are trained in joint angle space using observations from the head, left-hand, and right-hand cameras. We evaluate two imitation learning models: ACT and π_0. The first four tasks are trained with ACT, while the remaining four are trained with π_0. Each trained policy is tested on 10 inference trials per task, with results shown in Table IV.

TABLE IV
SUCCESS RATE (%) FOR REAL-WORLD IMITATION LEARNING.

Model	Task	Stage	Success Rate
ACT [14]	Pick Lemon and Apple	Pick with both arms	10/10
		Place on plate	10/10
	Play Music Two Tigers	Left plays *do* and *re*	7/10
		Right plays *mi* and *do*	7/10
	Move Box	Both arms hold box	10/10
		Move to table center	10/10
	Cut Corn	Left holds corn	9/10
		Right cuts corn	8/10
π_0 [19]	Pour Rice	Left picks bowl	10/10
		Right pours rice	10/10
	Clean Table	Left holds dustpan	10/10
		Right sweeps objects	9/10
	Open Drawer and Pick	Left opens drawer	10/10
		Right picks vegetable	10/10
	Stack Cups	Both arms pick cups	9/10
		Right stacks onto left	7/10

We observe that task success is influenced by coordination complexity. *Play Music Two Tigers* and *Cut Corn* (Fig. 6(1) and (2)) require precise dual-arm coordination and fine-grained control, yet achieve success rates of 70% and 80%, respectively, demonstrating reliable performance. The remaining failures stem from limitations of the learning algorithm, which requires stronger position generalization and closed-loop control for such precision-demanding tasks. In contrast, long-horizon tasks with stable subgoals, such as *Open Drawer and Pick* (Fig. 6(3)), are completed with a 100% success rate. While single-arm platforms such as PAMY2 [20], DROID [21], and TidyBot++ [22] cannot execute these bimanual tasks. These results highlight the quality of the demonstrations and the effectiveness of OpenPyRo-A1 for real-world manipulation. Additional qualitative results are provided in Fig. 6.

D. Embodied AI

To evaluate system adaptability, we present an agentic framework for Embodied AI that integrates perception, motion planning, and imitation learning to perform tasks from natural language commands (Fig. 7). A human-recorded task description is converted to text, and a Vision-Language Model (VLM) processes an environment image to generate a scene description. Both are included in the prompt along with available skills and previously enacted actions. At each step, the LLM selects the next skill. Following the Socratic approach [23], we integrate the LLM (DeepSeek) and VLM (InternVL).

We implement three core skills: **(1) Fold Clothes:** Cloth keypoints are detected from head camera images using RAG, 3D positions are recovered from depth, and `MPlib` generates folding trajectories. **(2) Pour Water:** YOLOv10 [38] detects bottle and cup, depth is used for 3D positions and pouring trajectory planning. **(3) Select Fruits:** Two DMP-based policies guide reaching motions to fruit and basket, using pre-collected grasp configurations and gripper actions.

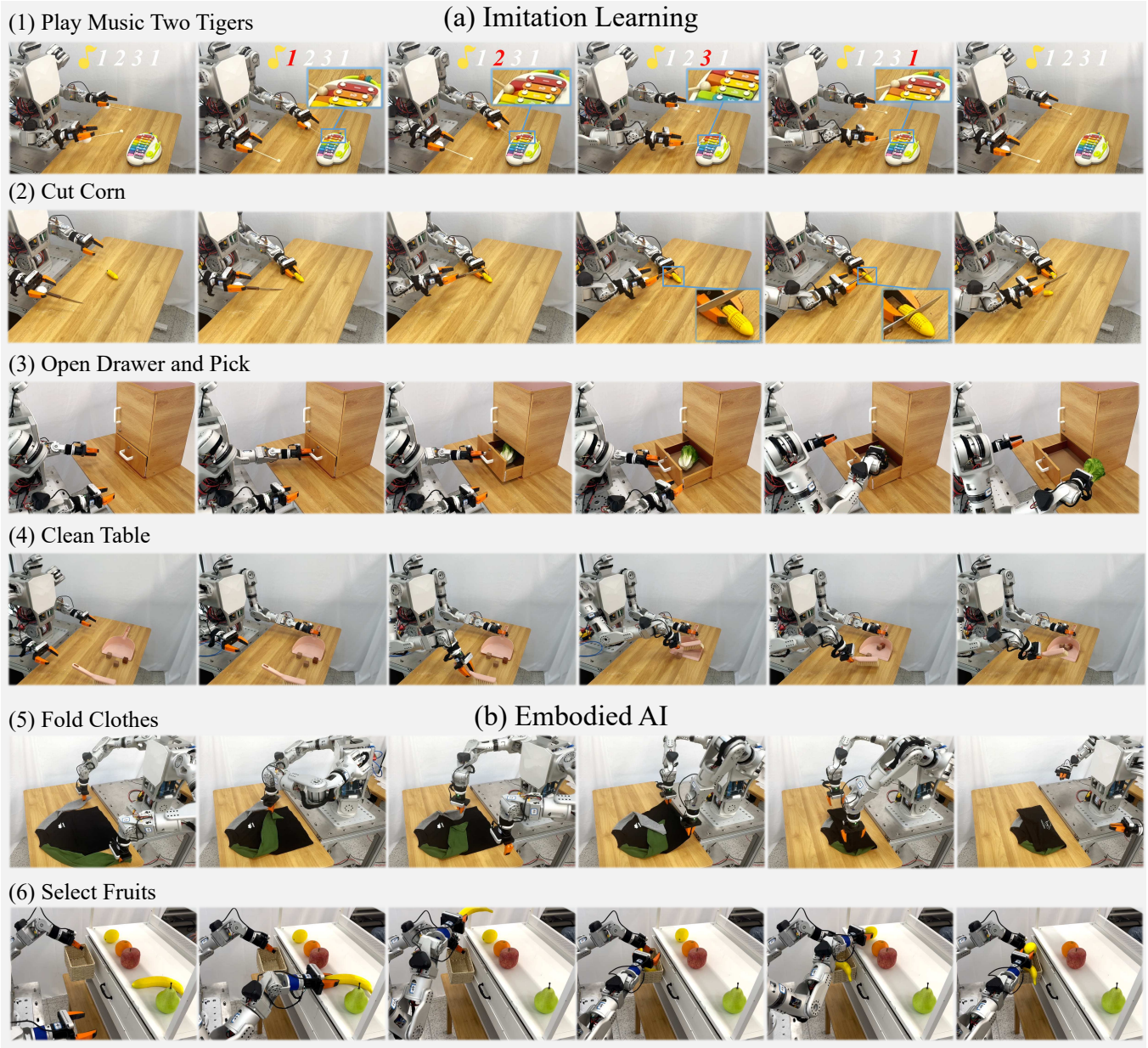


Fig. 6. **Selected snapshots from inference on OpenPyRo-A1.** The first four rows illustrate tasks performed using imitation learning, including: (1) Play Music Two Tigers, (2) Cut Corn, (3) Open Drawer and Pick, and (4) Clean Table. The last two rows show tasks from embodied AI benchmarks: (5) Fold Clothes and (6) Select Fruits. For each row, images from left to right show a sequence of frames from the video capturing the robot completing the task.

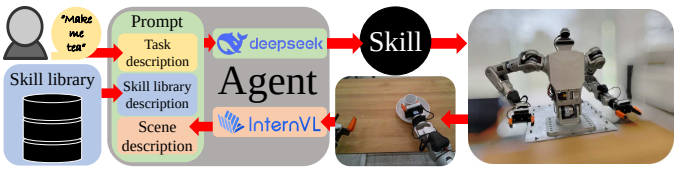


Fig. 7. Embodied AI agent framework on OpenPyRo-A1.

We evaluate the framework on tasks such as "please give me a cup of water," "fold the clothes," and "fill the basket with yellow fruit." In each case, the LLM selects relevant skills from the library, demonstrating reasoning and task-specific planning. This integration enables generalization across tasks by combining high-level decisions with low-level learned skills. Each task was repeated 10 times; *Fold Clothes*, *Pour Water*, and *Select Fruits* achieved success rates of 80%, 80%, and 70%, respectively. Most failures resulted from motion planning errors due to object placement or limited camera view. Visual results for *Fold Clothes* and *Select Fruits* are shown in the last two rows of Fig. 6.

VI. Conclusion

In this work, We introduce OpenPyRo-A1, a low-cost, modular dual-arm humanoid robot, along with a distributed

Python-based framework for control, data collection, skill learning, deployment, and Embodied AI. We validate the hardware, software, sim2real transfer, and imitation learning pipeline through experiments demonstrating precise dual-arm manipulation. Future work will enhance the software, integrate force control, and explore alternative materials to reduce cost and improve durability.

ACKNOWLEDGMENT

The authors thank Miaomiao Ouyang, Zhirui Yan, Pengxiang Xia, and Zhenyun Long for their contributions to artwork, calibration, and robot repair.

REFERENCES

[1] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, *et al.*, "Rt-1: Robotics transformer for real-world control at scale," 2022, *arXiv:2212.06817*.

[2] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," 2023, *arXiv:2307.15818*.

[3] 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," 2024, *arXiv:2406.09246*.

[4] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, *et al.*, "Octo: An open-source generalist robot policy," 2024, *arXiv:2405.12213*.

[5] H. Wu, Y. Jing, C. Cheang, G. Chen, J. Xu, X. Li, M. Liu, H. Li, and T. Kong, "Unleashing large-scale video generative pre-training for visual robot manipulation," 2023, *arXiv:2312.13139*.

[6] 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*, p. 02783649241273668, 2023.

[7] Z. Fu, T. Z. Zhao, and C. Finn, "Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation," 2024, *arXiv:2401.02117*.

[8] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, "What matters in learning from offline human demonstrations for robot manipulation," 2021, *arXiv:2108.03298*.

[9] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, *et al.*, "Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 6892–6903.

[10] Y. Cui, Z. Xu, L. Zhong, P. Xu, Y. Shen, and Q. Tang, "A task-adaptive deep reinforcement learning framework for dual-arm robot manipulation," *IEEE Transactions on Automation Science and Engineering*, 2024.

[11] Y. Liu, W. Su, Z. Li, G. Shi, X. Chu, Y. Kang, and W. Shang, "Motor-imagery-based teleoperation of a dual-arm robot performing manipulation tasks," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 11, no. 3, pp. 414–424, 2018.

[12] G. Tang, S. Liu, M. Sun, Y. Wang, W. Zhu, D. Wang, X. Li, H. Wu, S. Men, L. Zhang, *et al.*, "High-precision all-in-one dual robotic arm strategy in oral implant surgery," *BDJ open*, vol. 10, no. 1, p. 43, 2024.

[13] J. Liu, Y. Chen, Z. Dong, S. Wang, S. Calinon, M. Li, and F. Chen, "Robot cooking with stir-fry: Bimanual non-prehensile manipulation of semi-fluid objects," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5159–5166, 2022.

[14] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," 2023, *arXiv:2304.13705*.

[15] A. S. Huang, E. Olson, and D. C. Moore, "Lcm: Lightweight communications and marshalling," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 4057–4062.

[16] A. Correia and L. A. Alexandre, "A survey of demonstration learning," *Robotics and Autonomous Systems*, vol. 182, p. 104812, 2024.

[17] D. J. Brooks and H. A. Yanco, "Design of a haptic joystick for shared robot control," in *2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2012, pp. 113–114.

[18] R. Pettitt, E. Redden, and C. Carstens, *Scalability of robotic controllers: An evaluation of controller options*, 01 2011, pp. 51–113.

[19] 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," 2024, *arXiv:2410.24164*.

[20] S. Guist, J. Schneider, H. Ma, L. Chen, V. Berenz, J. Martus, H. Ott, F. Grüninger, M. Muehlebach, J. Fiene, *et al.*, "Safe & accurate at speed with tendons: A robot arm for exploring dynamic motion," 2023, *arXiv:2307.02654*.

[21] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, *et al.*, "Droid: A large-scale in-the-wild robot manipulation dataset," 2024, *arXiv:2403.12945*.

[22] J. Wu, W. Chong, R. Holmberg, A. Prasad, Y. Gao, O. Khatib, S. Song, S. Rusinkiewicz, and J. Bohg, "Tidybot++: An open-source holonomic mobile manipulator for robot learning," in *Conference on Robot Learning*, 2024.

[23] A. Zeng, M. Attarian, B. Ichter, K. Choromanski, A. Wong, S. Welker, F. Tombari, A. Purohit, M. Ryoo, V. Sindhwani, *et al.*, "Socratic models: Composing zero-shot multimodal reasoning with language," 2022, *arXiv:2204.00598*.

[24] A. Fabisch, "movement_primitives: Imitation learning of cartesian motion with movement primitives," *Journal of Open Source Software*, vol. 9, no. 97, p. 6695, 2024.

[25] D. Hanson, A. Imran, G. Morales, V. Krisciunas, A. Sagi, A. Malali, R. Mohbe, and R. Upadrashta, "Open arms: Open-source arms, hands & control," 2022, *arXiv:2205.12992*.

[26] C. W. Herron, A. J. Fuge, B. C. Beiter, Z. J. Fuge, N. J. Tremaroli, S. Welch, M. Stelmack, M. Kogelis, P. Hancock, I. F. E. Simões, C. Runyon, I. Pressgrove, and A. Leonessa, "Pandora: The open-source, structurally elastic humanoid robot," in *2024 IEEE-RAS 23rd International Conference on Humanoid Robots (Humanoids)*, 2024, pp. 24–31.

[27] P. Allgeuer, H. Farazi, M. Schreiber, and S. Behnke, "Child-sized 3d printed igus humanoid open platform," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, 2015, pp. 33–40.

[28] G. Metta, L. Natale, F. Nori, and G. Sandini, "The icub project: An open source platform for research in embodied cognition," in *Advanced Robotics and its Social Impacts*, 2011, pp. 24–26.

[29] G. Ficht, P. Allgeuer, H. Farazi, and S. N.-O. Behnke, "Grown-up 3d printed open humanoid platform for research," in *Proceedings of the 2017 IEEE-RAS 17th International conference on humanoid robotics (Humanoids), Birmingham, UK*, 2017, pp. 15–17.

[30] B. Johansson, T. A. Tjøstheim, and C. Balkenius, "Epi: An open humanoid platform for developmental robotics," *International Journal of Advanced Robotic Systems*, vol. 17, no. 2, p. 1729881420911498, 2020.

[31] M. Lapeyre, P. Rouanet, J. Grizou, S. Nguyen, F. Depraetre, A. Le Falher, and P.-Y. Oudeyer, "Poppy project: open-source fabrication of 3d printed humanoid robot for science, education and art," in *Digital Intelligence 2014*, 2014, p. 6.

[32] C. C. Johnson, A. Clawson, and M. D. Killpack, "Baloo: A large-scale hybrid soft robotic torso for whole-arm manipulation," 2025, *arXiv:2409.08420*.

[33] K. F. Gbagbe, M. A. Cabrera, A. Alabbas, O. Alyunes, A. Lykov, and D. Tsetserukou, "Bi-vla: Vision-language-action model-based system for bimanual robotic dexterous manipulations," 2024, *arXiv:2405.06039*.

[34] M. H. Khan, S. Asfaw, D. Iarchuk, M. A. Cabrera, L. Moreno, I. Tokmurziyev, and D. Tsetserukou, "Shake-vla: Vision-language-action model-based system for bimanual robotic manipulations and liquid mixing," 2025, *arXiv:2501.06919*.

[35] A. Dastider, H. Fang, and M. Lin, "Apex: Ambidextrous dual-arm robotic manipulation using collision-free generative diffusion models," 2024, *arXiv:2404.02284*.

[36] Kenta-Tanaka et al., "kinpy." [Online]. Available: https://github.com/neka-nat/kinpy

[37] S. Tao, F. Xiang, A. Shukla, Y. Qin, X. Hinrichsen, X. Yuan, C. Bao, X. Lin, Y. Liu, T.-k. Chan, *et al.*, "Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai," 2024, *arXiv:2410.00425*.

[38] A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, *et al.*, "Yolov10: Real-time end-to-end object detection," *Advances in Neural Information Processing Systems*, vol. 37, pp. 107 984–108 011, 2024.