TelePreview: A User-Friendly Teleoperation System with Virtual Arm Assistance for Enhanced Effectiveness

Jingxiang Guo[1*], Jiayu Luo[1*], Zhenyu Wei[1*], Yiwen Hou[1],
Zhixuan Xu[1], Xiaoyi Lin[1], Chongkai Gao[1], Lin Shao[1†]

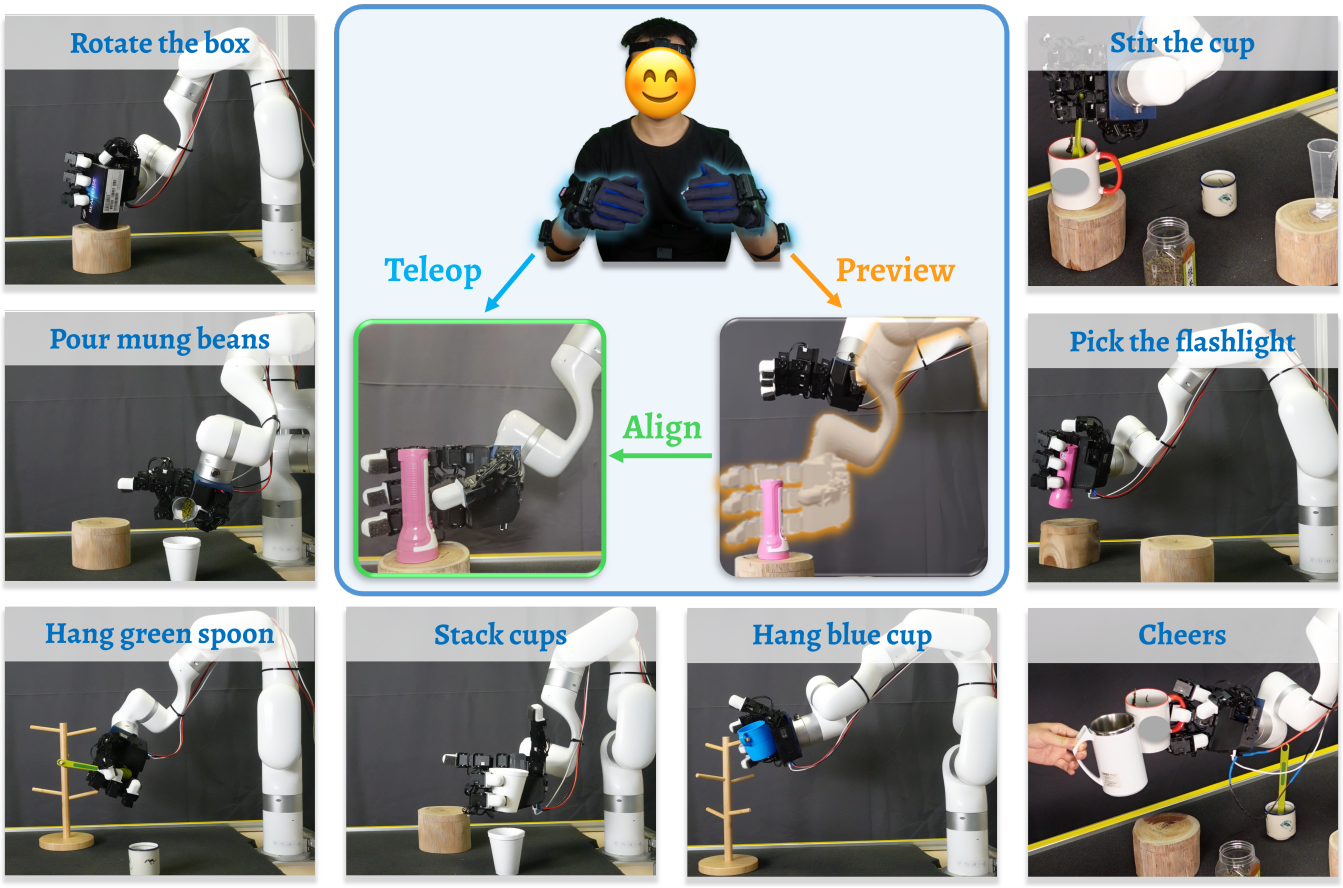


Fig. 1: **TelePreview** is a user-friendly teleoperation system enabling the real-time virtual preview before robot execution.

***Abstract*— Teleoperation provides an effective way to collect robot data, which is crucial for learning from demonstrations. In this field, teleoperation faces several key challenges: user-friendliness for new users, safety assurance, and transferability across different platforms. While collecting real robot dexterous manipulation data by teleoperation to train robots has shown impressive results on diverse tasks, due to the morphological differences between human and robot hands, it is not only hard for new users to understand the action mapping but also raises potential safety concerns during operation. To address these limitations, we introduce TelePreview. This teleoperation system offers real-time visual feedback on robot actions based on human user inputs, with a total hardware cost of less than $1,000. TelePreview allows the user to see a virtual robot that represents the outcome of the user's next movement. By enabling flexible switching between command visualization and actual execution, this system helps new users learn how to demonstrate quickly and safely. We demonstrate that it outperforms other teleoperation systems across five tasks, emphasize its ease of use, and highlight its straightforward deployment across diverse robotic platforms. We release our code and a deployment document on our website `https://nus-lins-lab.github.io/telepreview/`.**

* denotes equal contribution
† denotes the corresponding author
[1]Jingxiang Guo, Jiayu Luo, Zhenyu Wei, Yiwen Hou, Zhixuan Xu, Xiaoyi Lin, Chongkai Gao, and Lin Shao are with the Department of Computer Science, National University of Singapore. jingxiangguo@u.nus.edu, jiayu@comp.nus.edu.sg, Zhenyu_Wei@sjtu.edu.cn, yiwen328@comp.nus.edu.sg, zhixuanxu@u.nus.edu, gaochongkai@u.nus.edu, linshao@nus.edu.sg

I. Introduction

Researchers have long recognized teleoperation as an essential component for gathering on-robot data used in

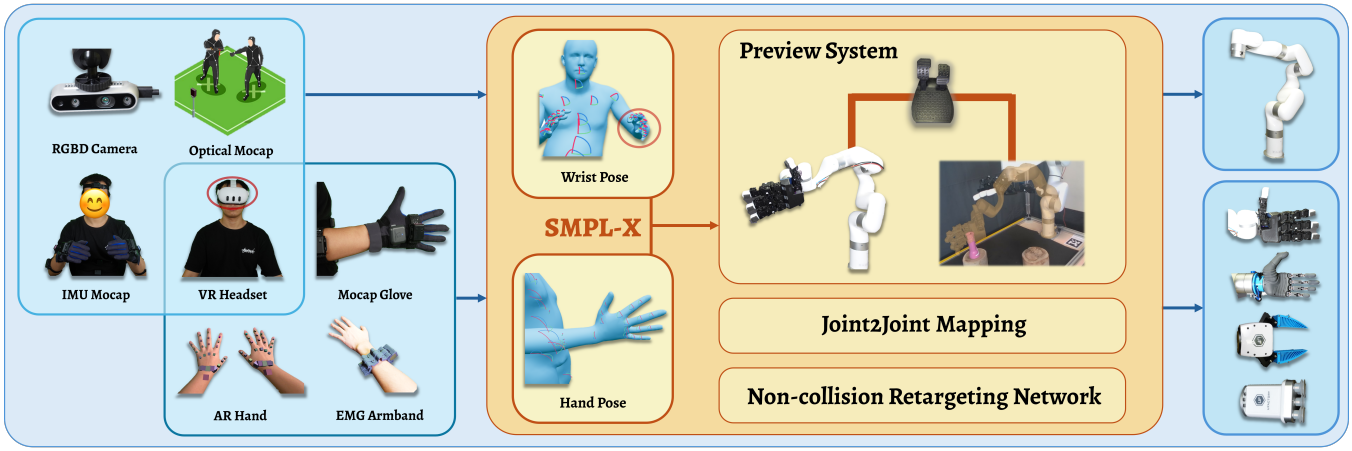


Fig. 2: **Overview of the TelePreview System Architecture:** Our system consists of three main components: (1) Various input devices, including RGB-D cameras, IMU mocap suits, VR headsets, mocap gloves, AR hand tracking, and EMG armbands for capturing human motion; (2) A central processing pipeline that uses SMPL-X as a standard to process wrist and hand pose data, feeding into our Preview System with joint-to-joint mapping and a non-collision retargeting network; (3) Support for different robot platforms as output devices for executing the mapped movements.

learning from demonstrations [1]–[12]. An intuitive and user-friendly teleoperation system is crucial for collecting high-quality, diverse, and scalable data. However, effectively controlling robots with dexterous hands remains a significant challenge. The fundamental differences between human and robotic hands make the direct mapping of human movements to robotic actions exceptionally complex. These inherent structural and functional disparities, combined with the precision required for fine-grained tasks, often result in inefficient and potentially compromised data collection processes.

Existing teleoperation systems face three significant challenges. First, new users often struggle to control the robot effectively due to the lack of intuitive feedback on how their commands translate to robot actions, especially during complex manipulation tasks. Second, without proper safeguards and real-time guidance, users may inadvertently issue unsafe commands that could damage the robot or its surroundings. Third, most current systems are tightly coupled with specific input devices or robot platforms, limiting their adaptability. These challenges significantly impact the quality and efficiency of data collection for robot learning.

To address these issues, we introduce **TelePreview**, a teleoperation system that delivers precise control while maintaining affordability through its sub-$1,000 hardware setup, as shown in Fig. 2. At its core, TelePreview provides an interface that overlays a **virtual robotic arm** onto the real-world scene, enabling users to preview motions. Precisely, a foot pedal or other external signal IO allows for mode switching between *preview* (virtual-only) and *align* (physical action alignment) modes. In *preview* mode, users can verify and refine their intended actions with a virtual robot arm that is spatially aligned with the physical robot, ensuring that the final commands are safe and accurate once applied in the real world.

We present the significant contributions of TelePreview to advance teleoperation:

- **Interactive Visual Assistance:** We develop a visualization interface that enables users to preview intended actions before execution, enhancing teleoperation precision and accessibility for both new and experienced users during complex tasks.
- **Low-Cost System:** We introduce an economical yet high-performance teleoperation solution that integrates inertial motion capture technology and mocap gloves at a total hardware cost of under $1,000, making it affordable for research.
- **Easy Adaptation to New Hardware:** We design our approach so that migrating to different input or output devices requires adjusting only a small set of physically interpretable parameters, ensuring robust adaptation across diverse hardware setups.

II. Related Work

A. Robot Teleoperation Frameworks for Manipulation

The growing interest in training robots through imitation learning has driven the need for extensive, high-quality real robot datasets. Teleoperation provides an efficient method for demonstrating and recording intricate robotic tasks [13]–[15]. Researchers have explored various teleoperation systems, including VR controllers [1]–[3], motion capture [16]–[18], wearable gloves [11], [19]–[21], exoskeletons [22], [23], each offering distinct benefits in terms of accessibility, precision, and generalizability.

Vision-based teleoperation systems [6] typically employ RGB or RGB-D cameras to detect hand poses. The hand pose retargeting module maps the human hand pose data obtained from perception algorithms into joint positions of the dexterous hand. We formulate this process as an optimization problem that minimizes the difference between the key point vectors of the human hand and the dexterous

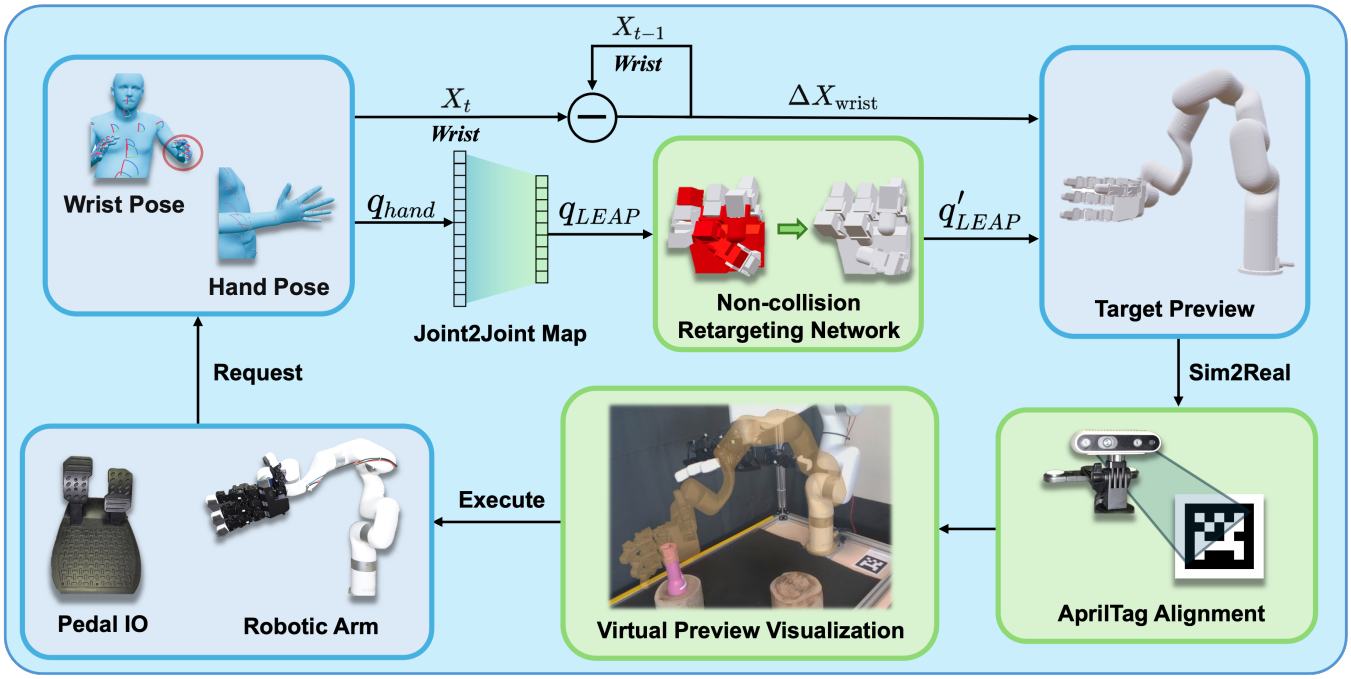


Fig. 3: **Pipeline of Our Modules:** The system tracks user wrist and hand poses, maps them to robot joint configurations through joint-to-joint mapping and non-collision retargeting, and provides visual preview before physical execution. We achieve precise alignment between virtual and physical robots through AprilTag calibration.

hand. To generate the end-effector motions, 3D positions of key points in the camera frame are calculated by using an RGB-D camera. The wrist pose is then computed by keypoint positions in the local wrist frame and global camera frame using the Perspective-n-Point (PnP) algorithm.

Although vision-based teleoperation systems offer a relatively low-cost solution [1]–[3], [6], they face significant challenges, including limited degrees of freedom (DoFs), constrained reachable workspace, and high computational demands. This limitation stems from the requirement of vision-based approaches for operators to keep their hands within the camera's field of view, thereby restricting the action space of the human hand. Although increasing the scale factor can amplify end-effector movement and expand the action space of the robot, it aggravates the difficulty for operators in performing fine-grained tasks. These approaches also struggle to handle scenarios where fingers overlap. The extra keypoint retargeting module can be computationally expensive in real-time. Although specialized hardware [4] can improve accuracy, it substantially increases system costs. Our proposed system directly uses the joint pose obtained from the motion capture suit and data gloves, preventing the deviation of vision-based systems and reducing the computational cost. Recent unified teleoperation frameworks attempt to address these limitations but often introduce new trade-offs. Some approaches [3], [6] prioritize implementation simplicity by tracking only fingertip positions rather than enabling detailed joint-level control. Furthermore, your customization options typically require time-consuming manual parameter tuning that requires significant expertise.

B. *Visual Feedback in Teleoperation Systems*

Visual feedback plays a crucial role in teleoperation systems by providing users with spatial awareness and task-relevant information. Traditional approaches primarily rely on real-time camera feeds [24], [25], which can be limited by occlusions and restricted viewing angles. Moreover, the lack of preview capabilities for intended commands leaves users (especially new users) uncertain about the robot's response, which can increase error rates and user fatigue.

To address these limitations, researchers have explored enhanced visualization techniques [26]–[30] that overlay virtual information on the workspace view. Even though these approaches improve user perception by displaying planned trajectories and environmental constraints, they primarily design and visualize AR objects for a specific task, such as a pink box with dimensions of 10 cm × 10 cm × 10 cm. Any changes to the object's attributes (e.g., dimensions, color, or type) invalidate the preconfigured AR object, so the system must rebuild it to match the new characteristics.

Instead of visualizing the manipulated objects, our system provides an AR representation of the robot that directly maps operator commands. This design ensures safety for new users who may be unfamiliar with how the robot responds to their inputs. Our configuration allows users to use their own URDF file, eliminating the need to reconstruct manipulated objects for every new task.

III. System Overview

Our system offers an integrated workflow for controlling robot movements with both precision and safety, leveraging low-cost hardware (under $1,000) and diverse input devices. This workflow comprises two complementary modules:

1) **Teleoperation Module**: Includes a teleoperation pipeline for robust motion retargeting.
2) **Preview-Based Module**: Allows operators to check intended movements in a virtual setting before execution.

As illustrated in Fig. 3, the *teleoperation module* (Section V) offers a safe and intuitive pipeline that proceeds in three stages:

1) Wrist pose estimation via IMU-based tracking (Section IV-A),
2) Hand pose estimation using a motion capture glove (Section IV-B),
3) Non-collision retargeting to ensure safe joint configurations (Section IV-C).

Together, these components collectively enable an efficient mapping from human motion to robot commands. Moreover, by adopting SMPL-X (Section IV-A.1) as a standardized representation of the human body, our pipeline can seamlessly integrate data from any input device conforming to this format.

Additionally, as illustrated in Fig. 3, the *preview-based module* (Section V) then provides a safe and intuitive pipeline that proceeds in three stages:

1) The preview robot is aligned with the physical robot using AprilTag markers [31] (Section V-B.1).
2) User commands are visualized on the aligned preview robot (Section V-B.2).
3) Once the I/O state transitions from active to inactive, the final preview pose is captured and converted into an optimal trajectory for actual execution (Section V-B.3).

This preview-then-execute scheme reduces errors and protects the robot and its surroundings by allowing users to visualize and refine their actions beforehand. Thus, it substantially improves efficiency and safety.

IV. Teleoperation Pipeline

A. Wrist Pose Estimation

Our system tracks the human operator's wrist motion to enable intuitive robot teleoperation. The input consists of raw IMU sensor data from sensors attached to the operator's arm, while the output is the 6-DoF wrist pose (position and orientation) in world coordinates. This wrist pose serves as the primary control signal for the robot's end-effector, allowing natural mapping between human arm movements and robot motion.

1) SMPL-X Standard: We adopt the SMPL-X standard [32] as our standard representation of the human body. The SMPL-X model represents a kinematic tree with standardized joint coordinate systems, where each joint's pose describes a rotation and translation relative to its parent frame. This hierarchical structure enables consistent pose representation across different motion capture devices and simplifies the integration with our TelePreview system.

2) IMU-based Tracking System: IMU-based motion capture offers robust tracking regardless of visual conditions and occlusions. Following the SMPL-X convention, we define the world frame system at the midpoint between the feet. All positions and orientations in this paper are expressed in this coordinate system unless otherwise specified. After obtaining the SMPL-X model, we compute the wrist position $\mathbf{p}_w \in \mathbb{R}^3$ and orientation $\mathbf{R}_w \in SO(3)$ relative to this world frame.

Based on the wrist parameters, we map the user's wrist pose to the end-effector pose through a transformation function $\mathcal{T}$. Specifically, we use the relative wrist position to control the end-effector position and the absolute wrist orientation to control the end-effector orientation:

$$\mathbf{p}_e(t) = \mathbf{p}_e(0) + (\mathbf{p}_w(t) - \mathbf{p}_w(0)), \quad \mathbf{R}_e(t) = \mathbf{R}_w(t), \quad (1)$$

where $\mathbf{p}_e(t)$ and $\mathbf{R}_e(t)$ represent the end-effector position and orientation at time t, and $\mathbf{p}_w(t)$ and $\mathbf{R}_w(t)$ represent the user's wrist position and orientation at time t.

B. Hand Pose Estimation

Mocap gloves [33] offer direct joint angle measurements $\{q_i\}_{i=1}^{27}$ using flex sensors. Our joint-to-joint mapping approach provides a generalizable solution for retargeting high-dimensional human hand motion data to lower-dimensional robotic systems. This approach is particularly effective when the input space (e.g., the human hand with 27 DoF) has higher dimensionality than the target space (e.g., the robotic hand with 16 DoF), as it allows selective mapping of the most intuitive and task-relevant degrees of freedom. Through manual selection of corresponding joints based on human intuition and task requirements, followed by range alignment between input and output spaces, we maintain natural and predictable motion correspondence.

We propose a direct joint-to-joint mapping function $\mathcal{M} : \mathbb{R}^{27} \to \mathbb{R}^{16}$ that transforms mocap glove readings to LEAP hand [34] joint configurations. Let $q_g \in \mathbb{R}^{27}$ denote the glove joint values and $q_r \in \mathbb{R}^{16}$ denote the robot joint values. For each robot joint i, we define a mapping:

$$q_r^i = f_i(q_g^{k_i}), \quad i \in 1, \ldots, 16, \quad (2)$$

where k_i is the corresponding glove joint index for robot joint i which is manually picked, and f_i is a linear transformation:

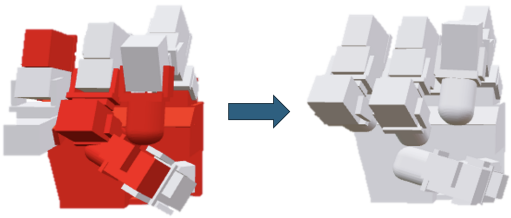
$$f_i(x) = s_i(x - b_i)r_i. \quad (3)$$

Here, $s_i \in \mathbb{R}^+$ is a scaling factor that normalizes the joint ranges, $b_i \in \mathbb{R}$ is a bias term that aligns the neutral positions, and $r_i \in$ {-1,1} is a direction indicator that ensures consistent joint rotations. This mapping guarantees:

$$\max q_g^{k^i} \Leftrightarrow \max q_r^i, \quad \min q_g^{k^i} \Leftrightarrow \min q_r^i, \quad (4)$$

Where q_g and q_r represent the joint angles of the glove and the Leap Hand, respectively, and max and min denote their upper and lower joint limits.

(a) Current Retargeting Method [6] Causes Collision.



(b) Our Self-collision Avoidance Retargeting.

Fig. 4: **Comparison of Hand Configuration Retargeting Methods:** (a) shows the direct mapping between human and robot hands leading to self-collision; (b) demonstrates our collision-aware retargeting approach that maintains safe configurations.

C. Non-Collision Retargeting Network

Direct mapping from human hand postures to robotic hands using existing retargeting approaches like [6] can result in self-collisions, as demonstrated in Fig. 4a. While traditional retargeting methods can achieve real-time performance for simpler end-effectors, they rely on optimization-based approaches that become computationally intensive for high-DoF configurations.

To address these challenges, we develop a learning-based framework with two key components:

1) A Self-Collision Prediction Network (CPN) that takes robot joint configurations $q \in \mathbb{R}^n$ as input and outputs a binary collision probability vector $p \in \mathbb{R}^m$ indicating collision likelihood for each link.
2) A Configuration Correction Network (CCN) that maps collision-prone joint configurations to their nearest collision-free counterparts. Specifically, it transforms an invalid robot configuration $q_{invalid} \in \mathbb{R}^n$ to a valid configuration $q_{valid} \in \mathbb{R}^n$ that minimizes both collision risk and deviation from the original pose.

As demonstrated in Fig. 4b, our method successfully transforms problematic configurations into stable, safe positions while maintaining efficient real-time performance (about 60Hz).

V. Preview Pipeline

A. Why Preview System Assistance?

Traditional teleoperation approaches face significant challenges in data quality when collecting demonstrations for imitation learning. During complex manipulation tasks, users frequently make exploratory movements that contaminate the demonstration data with sub-optimal trajectories. These repeated probing actions to locate the socket edge and refine the approach angle before successful insertion introduced significant noise into the collected datasets. Such noise is particularly problematic for training imitation learning models, which benefit most from clean, purposeful demonstrations rather than trajectories cluttered with exploratory movements.

To address these limitations and provide a more efficient workflow, we propose incorporating a dedicated *preview* feature that allows users to refine their actions before physical execution. By separating exploration from the final command, our approach reduces sub-optimal motions that would otherwise degrade data quality.

B. Technical Implementation of Preview System

We design our preview feature to integrate seamlessly with the teleoperation workflow. It requires three critical elements: (1) precise spatial calibration to maintain alignment between the virtual preview and the physical robot, (2) realistic rendering to blend the preview with camera feeds, and (3) state management to handle transitions between preview and execution phases. The following subsections detail each of these technical components, highlighting how they collectively enable users to view and adjust planned motions prior to committing them to the physical robot.

1) Spatial Alignment Using AprilTag: To maintain a precise spatial alignment between the preview and the physical robot in the camera viewpoints, we implement a robust calibration system using AprilTags [31]. Our approach involves two sequential steps: (1) a hand-eye calibration that establishes the transformation between the robot's base frame and the fixed camera frame, and (2) AprilTag pose detection that enables real-time tracking of the robot's position and orientation relative to each camera view. This chain of transformations ensures consistent preview visualization across different viewpoints.

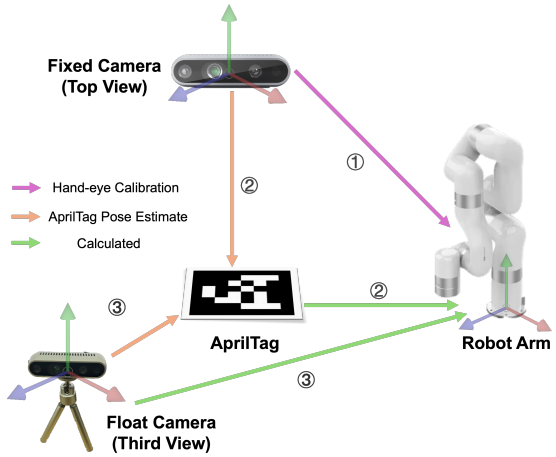
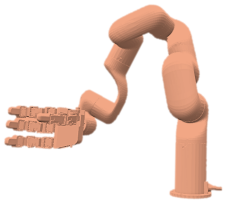


Fig. 5: **Our Transformation Relationship:** The number in the circle denotes the order of transformation acquisition.

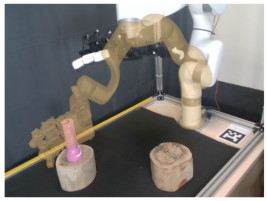
As shown in Fig. 5, our calibration procedure involves multiple sequential transformations. We begin with hand-eye calibration of the fixed top-view camera (1), followed

by detecting the AprilTag's pose from this camera's perspective (2). Using these relationships, we can compute the transformation between the AprilTag and the robot base frame (2). For the additional floating third-view camera, we first detect the AprilTag pose from its viewpoint (3), which then allows us to establish its position relative to the robot base frame through the standard AprilTag reference (3). This chain of transformations ensures consistent preview visualization regardless of camera movement or viewpoint changes, making our system robust for dynamic viewing scenarios.

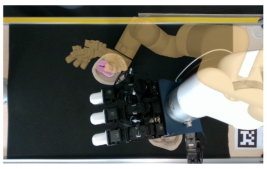
2) Visual Preview Visualization: Building upon this spatial alignment system, we implement the visual component of our preview system. As shown in Fig. 6a, we use the pyrender [35] library to render the virtual robot through the previous float camera. The rendered preview image is then composited with the actual camera feed using alpha blending, creating a seamless overlay where the virtual robot appears naturally integrated with the physical environment. This visualization approach requires careful calibration of the virtual camera parameters to match the physical cameras, ensuring that the preview accurately reflects the robot's intended configuration in the workspace. The resulting overlay provides users with an intuitive preview of planned motions. Our preview system also supports multiple viewpoints to enhance spatial awareness during complex manipulation tasks. As shown in Fig. 6b and Fig. 6c, we deploy cameras to capture both third-person and top-down views of the workspace, allowing users to verify planned movements from complementary perspectives.



(a) Preview Rendering



(b) Third-person View



(c) Top-down View

Fig. 6: Rendering and Multi-view Visualization System.

3) IO Control for Preview Visualization: Our system uses a foot pedal as an intuitive control interface for state transitions. As shown in Fig. 7, when the IO is activated, the physical robot stops while the preview appears and responds to user commands for motion preview (Preview Mode). Upon IO deactivation, the preview disappears, and the system extracts its final configuration as the target pose to align, bypassing any exploratory movements made during the preview (Align Mode). This target is then processed by the mplib motion planning library [37] to generate an optimal trajectory for execution. After reaching the target pose, the robot automatically resumes real-time teleoperation until the next IO is activated. This workflow ensures users execute only refined, intentional movements, eliminating exploratory motions from demonstration data.

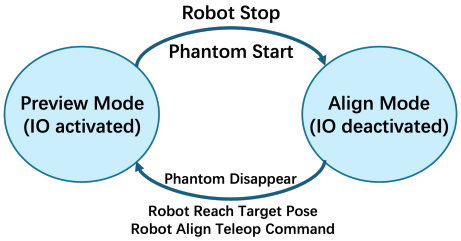


Fig. 7: State Transition of Preview Control

VI. Experiments

Our experiments aim to address the following questions:
Q1: How effective is TelePreview vs. Baselines?
Q2: How beneficial is TelePreview for new users?
Q3: How adaptable is TelePreview for new hardware?

A. Experiment Setup

TelePreview integrates several key components. We capture the user's movements via an inertial motion capture suit [38] and a pair of strain-gauge-based mocap gloves [33], providing high-fidelity joint angle data for both body posture and delicate hand movements. To visualize robot action before execution, we employ a RealSense RGB-D camera and render the preview either on a standard 2D display or within a VR headset (e.g., Meta Quest 3 [39]). For the physical hardware, we use a UFactory xArm-6 robotic arm [40] outfitted with a Leap hand [34], utilizing AprilTag [31] to align the initial poses of the preview and the real robot.

Following the methodology outlined in [6], [36], users attempted each task 10 times. We reference the baseline success rates from their papers.

B. Task Descriptions

We test TelePreview on five real-world tasks, each highlighting different manipulation challenges:

- **Pick & Place:** Grasp an object and place it at precise locations.
- **Pour:** Tilt and rotate a cup to pour the beans into another container.
- **Hang:** Hang a spoon on a peg, requiring both fine positioning and subtle wrist rotations.
- **Box Rotation:** Rotate a box to change its orientation.
- **Cup Stacking:** Stack a cup onto another cup.

C. Evaluation Metrics

We measure two key metrics for each task:

- **Success Rate:** We define the success rate as the proportion of times the operator completes the task within

Task	TelePreview	Open Teach [3]	AnyTeleop [6]	Telekinesis [36]
Pick & Place	**1.0**	0.8	**1.0**	0.9
Hang	**0.9**	-	-	-
Pour	**1.0**	0.8	0.7	0.7
Box Rotation	**1.0**	-	0.6	0.6
Cup Stacking	**1.0**	-	0.7	0.3

TABLE I: **Real Robot Teleoperation Results.** We use the success rates the baseline methods report in their papers, marking tasks they did not attempt or do not support with "-". Success rates for Open Teach [3] reflect expert performance.

Task	Success Rate			Average Success Execution Time (s)		
	w/o preview	w/ preview	Difference ↑	w/o preview	w/ preview	Difference ↓
Pick & Place	0.6	1.0	**+0.4**	23.56	13.55	**-10.01**
Hang	0.6	1.0	**+0.4**	29.30	30.83	+1.53
Pour	0.9	0.8	-0.1	43.20	36.13	**-7.07**
Box Rotation	0.6	0.8	**+0.2**	30.47	19.12	**-11.35**
Cup Stacking	0.5	1.0	**+0.5**	31.54	18.91	**-12.63**

TABLE II: Effect of Preview Assistance on New User Performance.

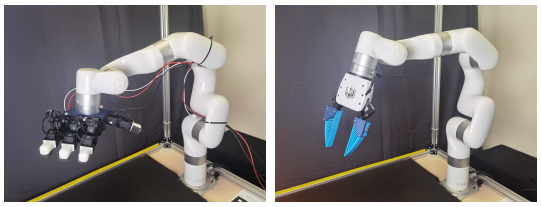
the given constraints. For example, if the user attempts a cup-hanging task X times and successfully hangs the cup on the rack Y times, the success rate is $\frac{Y}{X}$.

- **Execution Time:** The total duration from task initiation to successful completion (i.e., the length of the recorded demo episode).

D. Performance Comparison with Baselines (Q1)

To address **Q1**—evaluating how our approach compares to existing teleoperation methods—we measure the actual robot performance of TelePreview against three baselines: Telekinesis [36], AnyTeleop [6], and Open Teach [3]. For fairness and consistency, an expert user operates each method and tests it on the five tasks described in Section VI-B. The primary metrics are success rate and average execution time.

As summarized in Table I, TelePreview achieves higher success rates than all baselines. While Telekinesis, AnyTeleop, and Open Teach handle simpler, short-horizon tasks, they often struggle with complex manipulations requiring subtle wrist rotations and fine-grained positioning (e.g., the Hang task). TelePreview, by contrast, captures these nuances more effectively, reflecting stronger overall performance and versatility. These findings demonstrate TelePreview's ability to address a broad spectrum of real-world challenges, from basic pick-and-place operations to more intricate tasks involving delicate movements.



(a) LEAP Hand (b) Gripper

Fig. 8: Deployment on Different Robots.

E. Effect of Preview for Users (Q2)

To investigate **Q2**—how the preview feature benefits new (non-expert) users—we evaluate both **success rate** and **average execution time** in two conditions: (1) with the preview enabled and (2) with the preview disabled. The same five tasks from Section VI-B are attempted by new users, with each task repeated 10 times.

As shown in Table II, enabling the preview boosts user performance. Success rates increase across all tasks, with especially notable gains in challenging manipulations such as *cup stacking* (+0.5) and *pick & place* (+0.4). Moreover, average execution times drop by up to 12.63 seconds (cup stacking), demonstrating not only improved accuracy but also greater efficiency. By visualizing a virtual "preview" of the robot's intended motion and confirming the final pose prior to executing physical movements, users eliminate the need for exploratory motions in the actual workspace. This design ensures that recorded trajectories contain only intentional, task-oriented actions without extraneous trial-and-error, providing cleaner, high-quality demonstrations for imitation learning. As a result, TelePreview delivers both a more intuitive teleoperation experience and more consistent data for downstream training.

F. Deployment on Various Robots (Q3)

We deploy our TelePreview system on a Ufactory xArm [40] equipped with both a LEAP hand [34] and a parallel gripper. By updating only the kinematic parameters and transformation chain for each end-effector, TelePreview preserves its full functionality across diverse setups, as shown in Fig. 8. This successful deployment on multiple end-effector configurations illustrates the system's adaptability to new hardware. Its modular design allows users to seamlessly integrate TelePreview into different robotic platforms while retaining all key features—preview, multi-view visualization, and intuitive control interfaces.

VII. Conclusion

In this paper, we presented TelePreview, a low-cost teleoperation system featuring AR preview feedback designed for deployment across diverse robotic platforms. Our experiments demonstrate that TelePreview offers sufficient flexibility and safety to successfully perform a variety of fine-grained tasks. TelePreview provides immediate, robot-centric feedback that enables users to preview and refine their commands before physical execution.

While our results show promising advances in teleoperation interfaces, a key limitation is the visual ambiguity caused by occlusions between the preview robot and scene objects. Future work could address this by incorporating depth information from RGB-D cameras to create more accurate spatial relationships between virtual and physical elements.

References

[1] X. Cheng, J. Li, S. Yang, G. Yang, and X. Wang, "Open-television: Teleoperation with immersive active visual feedback," *arXiv preprint arXiv:2407.01512*, 2024.

[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," *arXiv preprint arXiv:2407.03162*, 2024.

[3] A. Iyer, Z. Peng, Y. Dai, I. Guzey, S. Haldar, S. Chintala, and L. Pinto, "Open teach: A versatile teleoperation system for robotic manipulation," *arXiv preprint arXiv:2403.07870*, 2024.

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

[5] A. George, A. Bartsch, and A. B. Farimani, "Openvr: Teleoperation for manipulation," *arXiv preprint arXiv:2305.09765*, 2023.

[6] 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," *arXiv preprint arXiv:2307.04577*, 2023.

[7] G. Zhu, X. Xiao, C. Li, J. Ma, G. Ponraj, A. Prituja, and H. Ren, "A bimanual robotic teleoperation architecture with anthropomorphic hybrid grippers for unstructured manipulation tasks," *Applied Sciences*, vol. 10, no. 6, p. 2086, 2020.

[8] D. Leonardis, M. Gabardi, S. Marcheschi, M. Barsotti, F. Porcini, D. Chiaradia, and A. Frisoli, "Hand teleoperation with combined kinaesthetic and tactile feedback: A full upper limb exoskeleton interface enhanced by tactile linear actuators," *Robotics*, vol. 13, no. 8, p. 119, 2024.

[9] K. Shaw, Y. Li, J. Yang, M. K. Srirama, R. Liu, H. Xiong, R. Mendonca, and D. Pathak, "Bimanual dexterity for complex tasks," *arXiv preprint arXiv:2411.13677*, 2024.

[10] Y. Park and P. Agrawal, "Using apple vision pro to train and control robots," 2024.

[11] C. Wang, H. Shi, W. Wang, R. Zhang, L. Fei-Fei, and C. K. Liu, "Dexcap: Scalable and portable mocap data collection system for dexterous manipulation," *arXiv preprint arXiv:2403.07788*, 2024.

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

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

[14] S. Lichiardopol, "A survey on teleoperation," Technische Universiteit Eindhoven, Technical Report DCT-2007.155, 2007.

[15] P. F. Hokayem and M. W. Spong, "Bilateral teleoperation: An historical survey," *Automatica*, vol. 42, no. 12, pp. 2035–2057, 2006.

[16] M. Hirschmanner, C. Tsiourti, T. Patten, and M. Vincze, "Virtual reality teleoperation of a humanoid robot using markerless human upper body pose imitation," in *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, 2019, pp. 259–265.

[17] C. Stanton, A. Bogdanovych, and E. Ratanasena, "Teleoperation of a humanoid robot using full-body motion capture, example movements, and machine learning," in *Proc. Australasian Conference on Robotics and Automation*, vol. 8, 2012, p. 51.

[18] N. Miller, O. C. Jenkins, M. Kallmann, and M. J. Mataric, "Motion capture from inertial sensing for untethered humanoid teleoperation," in *4th IEEE/RAS International Conference on Humanoid Robots, 2004.*, vol. 2. IEEE, 2004, pp. 547–565.

[19] H. Liu, Z. Zhang, X. Xie, Y. Zhu, Y. Liu, Y. Wang, and S.-C. Zhu, "High-fidelity grasping in virtual reality using a glove-based system," in *2019 international conference on robotics and automation (icra)*. IEEE, 2019, pp. 5180–5186.

[20] M. Mosbach, K. Moraw, and S. Behnke, "Accelerating interactive human-like manipulation learning with gpu-based simulation and high-quality demonstrations," in *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*. IEEE, 2022, pp. 435–441.

[21] M. Schwarz, C. Lenz, A. Rochow, M. Schreiber, and S. Behnke, "Nimbro avatar: Interactive immersive telepresence with force-feedback telemanipulation," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 5312–5319.

[22] S. Yang, M. Liu, Y. Qin, R. Ding, J. Li, X. Cheng, R. Yang, S. Yi, and X. Wang, "Ace: A cross-platform visual-exoskeletons system for low-cost dexterous teleoperation," *arXiv preprint arXiv:2408.11805*, 2024.

[23] P. Wu, Y. Shentu, Z. Yi, X. Lin, and P. Abbeel, "Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators," 2023.

[24] D. A. Lawrence, "Stability and transparency in bilateral teleoperation," *IEEE transactions on robotics and automation*, vol. 9, no. 5, pp. 624–637, 1993.

[25] A. M. Ousaid, D. S. Haliyo, S. Régnier, and V. Hayward, "A stable and transparent microscale force feedback teleoperation system," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 5, pp. 2593–2603, 2015.

[26] Z. Zhao, P. Huang, Z. Lu, and Z. Liu, "Augmented reality for enhancing tele-robotic system with force feedback," *Robotics and Autonomous Systems*, vol. 96, pp. 93–101, 2017.

[27] D. Lee and Y. S. Park, "Implementation of augmented teleoperation system based on robot operating system (ros)," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 5497–5502.

[28] P. Milgram, A. Rastogi, and J. J. Grodski, "Telerobotic control using augmented reality," in *Proceedings 4th IEEE International Workshop on Robot and Human Communication*. IEEE, 1995, pp. 21–29.

[29] Immersive Web Working Group, "Webxr samples," https://immersive-web.github.io/webxr-samples/, accessed: 22 December 2024.

[30] "Steamvr," https://www.steamvr.com/en/, accessed: 22 December 2024.

[31] E. Olson, "Apriltag: A robust and flexible visual fiducial system," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 3400–3407.

[32] G. Pavlakos, V. Choutas, N. Ghorbani, T. Bolkart, A. A. Osman, D. Tzionas, and M. J. Black, "Expressive body capture: 3d hands, face, and body from a single image," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 10 975–10 985.

[33] UdexReal, "UdCap: Silk-like VR gloves for SteamVR," Kickstarter Campaign, 2024, available at: https://www.kickstarter.com/projects/udexreal/udcap-silk-like-vr-gloves-for-steamvr.

[34] K. Shaw, A. Agarwal, and D. Pathak, "Leap hand: Low-cost, efficient, and anthropomorphic hand for robot learning," *Robotics: Science and Systems (RSS)*, 2023.

[35] M. Matl, "Pyrender," https://github.com/mmatl/pyrender, 2019.

[36] A. Sivakumar, K. Shaw, and D. Pathak, "Robotic telekinesis: Learning a robotic hand imitator by watching humans on youtube," *arXiv preprint arXiv:2202.10448*, 2022.

[37] H. S. Lab, "Mplib: A motion planning library," https://github.com/haosulab/mplib, 2023, accessed: 2024-12-15.

[38] D. Technology, "Rebocap: Affordable imu-based motion capture system," http://rebocap.com/, 2024, a 15-point full-body IMU motion capture system for games, films, animations, human motion analysis and live streaming applications. Accessed: 2024-03-16.

[39] Meta, "Meta quest 3," https://www.meta.com/quest/quest-3/, 2024, accessed: 2024-03-19.

[40] UFACTORY, "xarm series - 6 dof and 7 dof collaborative robot," https://www.cn.ufactory.cc/xarm, 2024, accessed: 2024-03-19.

APPENDIX

A. Technical Implementation of Pos-to-Pos Non-Collision Module

Due to differences in action space and limitations in the precision of the retargeting algorithm, the configuration of a dexterous robotic hand often generates invalid self-collision configurations. These invalid configurations not only lack operational utility but also risk system failure or hardware damage. To address this issue, we propose a method for mapping invalid configurations to their closest valid counterparts, enabling recovery from self-collision scenarios.

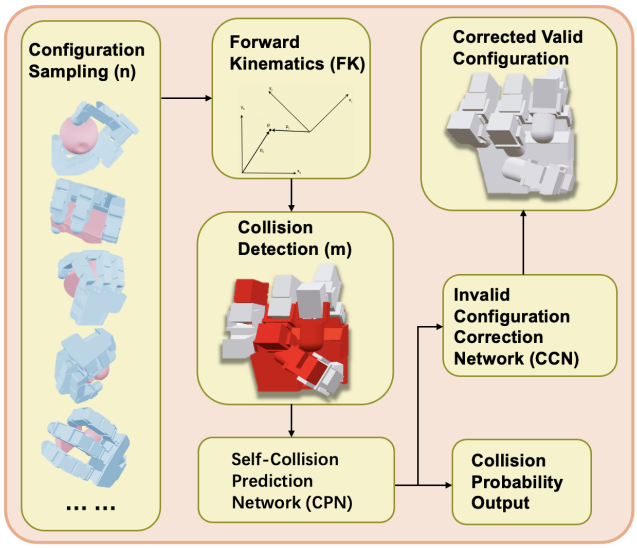


Fig. 9: Pipeline of the Non-collision Module

Fig. 9 illustrates our pos2pos implementation pipeline, which consists of several interconnected components for handling robot hand configurations.

1) Self-Collision Prediction Network (CPN): To facilitate the transformation from invalid to valid configurations, we first develop a **Self-Collision Prediction Network** (CPN). The primary objective of CPN is to predict the likelihood of self-collision for each link within a given joint configuration.

The training dataset is generated by uniformly sampling n configurations from the robot's action space. For each sampled configuration, the system employs forward kinematics (FK) to compute the robot's pose. A collision detection algorithm (e.g., geometric or physics-based) then checks the pose to derive m collision labels for the links. Each label indicates whether a link is in a collision state.

The CPN takes joint configurations as input and outputs collision probabilities for all joints. We train the network using the binary cross-entropy (BCE) loss function, defined as:

$$L_{\text{CPN}} = \frac{1}{m}\sum_{i=1}^{m} \text{BCE}(p_i, t_i), \tag{5}$$

where p_i and t_i represent the predicted and true collision probabilities, respectively.

2) Invalid Configuration Correction Network (CCN): Building on the CPN, we introduce an **Invalid Configuration Correction Network** (CCN) to map invalid configurations to valid ones. The CCN takes an invalid configuration as input and outputs a corrected configuration that minimizes collision risks while closely resembling the original input.

The CCN training process minimizes a composite loss function comprising two components:

- **Mean Squared Error (MSE) Loss**: Ensures the corrected configuration closely resembles the original configuration.

$$L_{\text{MSE}} = \frac{1}{n}\sum_{i=1}^{n} (\hat{q}_i - q_i)^2, \tag{6}$$

  where q_i and $\hat{q}_i$ denote the original and corrected joint configurations, respectively.
- **Collision Probability Loss**: Leverages the CPN to compute the mean collision probability of the corrected configuration and aims to minimize this value.

$$L_{\text{Collision}} = \frac{1}{m}\sum_{i=1}^{m} p_i(\hat{q}), \tag{7}$$

  where $p_i(\hat{q})$ represents the collision probability of joint i in the corrected configuration $\hat{q}$.

We define the total loss function as:

$$L = \alpha L_{\text{MSE}} + \beta L_{\text{Collision}}, \tag{8}$$

where α and β are hyperparameters balancing the two loss components.

3) Explanation and Optimization Strategy: The loss terms in the proposed framework serve distinct roles:

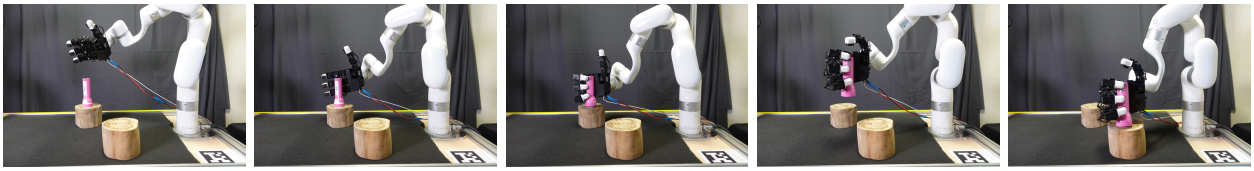
- L_{MSE} ensures the corrected configuration retains continuity with the original input.
- $L_{\text{Collision}}$ minimizes the likelihood of self-collision in the corrected configuration.
- The hyperparameters α and β significantly influence the training outcomes, and we optimize their values through grid search.

The CCN employs a fully connected multi-layer perceptron (MLP) architecture. The input is the invalid joint configuration q, and the output is the corrected configuration $\hat{q}$. We train the model using the Adam optimizer with a learning rate η and monitor convergence via the collision rate on a validation dataset.

4) Summary: By integrating the CPN and CCN, we efficiently transform invalid self-collision configurations into valid ones. This approach ensures the validity and continuity of robotic configurations, laying a robust foundation for subsequent task execution.

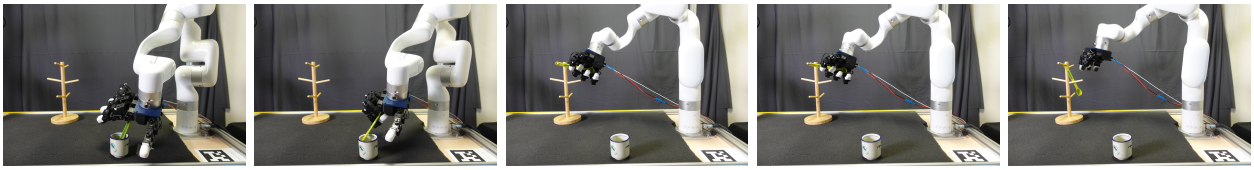
B. Visualization of our tasks

We visualize the execution process of our five manipulation tasks in Figure 10-14. They demonstrate the execution sequences of five manipulation tasks: picking and placing a cup, hanging a spoon on a peg, pouring beans between containers, rotating a box, and stacking cups.



(a) (b) (c) (d) (e)

Fig. 10: Pick&Place Visualization

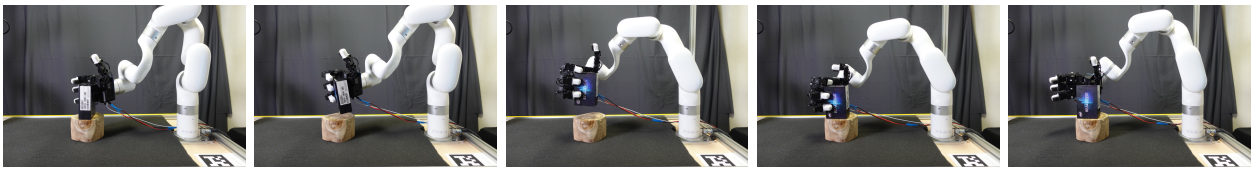


(a) (b) (c) (d) (e)

Fig. 11: Hang Visualization

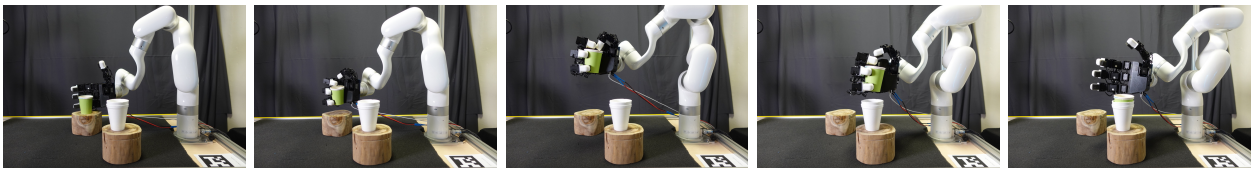
(a) (b) (c) (d) (e)

Fig. 12: Pour Visualization



(a) (b) (c) (d) (e)

Fig. 13: Box Rotation Visualization



(a) (b) (c) (d) (e)

Fig. 14: Cupstack Visualization