AnyTeleop: A General Vision-Based Dexterous Robot Arm-Hand Teleoperation System

Yuzhe Qin\*, Wei Yang\textsuperscript{2}, Binghao Huang\textsuperscript{1}, Karl Van Wyk\textsuperscript{2}
Hao Su\textsuperscript{1}, Xiaolong Wang\textsuperscript{1}, Yu-Wei Chao\textsuperscript{2}, Dieter Fox\textsuperscript{2}
\textsuperscript{1}UC San Diego \textsuperscript{2}NVIDIA

\url{http://anyteleop.com}

Fig. 1: We present AnyTeleop, a vision-based teleoperation system for a variety of scenarios to solve a wide range of manipulation tasks. AnyTeleop can be used for various robot arms with different robot hands. It also supports teleoperation within different realities, such as IsaacGym (top row), and SAPIEN simulator (middle row), and real world (bottom rows).

Abstract—Vision-based teleoperation offers the possibility to endow robots with human-level intelligence to physically interact with the environment, while only requiring low-cost camera sensors. However, current vision-based teleoperation systems are designed and engineered towards a particular robot model and deploy environment, which scales poorly as the pool of the robot models expands and the variety of the operating environment increases. In this paper, we propose AnyTeleop, a unified and general teleoperation system to support multiple different arms, hands, realities, and camera configurations within a single system. Although being designed to provide great flexibility to the choice of simulators and real hardware, our system can still achieve great performance. For real-world experiments, AnyTeleop can outperform a previous system that was designed for a specific robot hardware with a higher success rate, using the same robot. For teleoperation in simulation, AnyTeleop leads to better imitation learning performance, compared with a previous system that is particularly designed for that simulator.

I. Introduction

A grand goal of robotics is to endow robots with human-level intelligence to physically interact with the environment. Teleoperation [39], as a direct means to acquire human demonstrations for teaching robots, has been a powerful paradigm to approach this goal [22, 11, 64, 17, 34, 6, 19, 38, 51]. Compared to gripper-based manipulators, teleoper-
ating dexterous hand-arm systems poses unprecedented challenges and often requires specialized apparatus that comes with high costs and setup efforts, such as Virtual Reality (VR) devices [4] [17] [15], wearable gloves [29] [30], handheld controllers [45] [46] [20], haptic sensors [12] [23] [50] [53], or motion capture trackers [65]. Fortunately, recent developments in vision-based teleoperation [2] [24] [16] [26] [49] [41] have provided a low-cost and more generalizable alternative for teleoperating dexterous robot systems.

Despite the progress, the current paradigm of vision-based teleoperation systems still falls short when it comes to scaling up data collection for robot teaching. First, prior systems are often designed and engineered towards a particular robot model or deployment environment. For example, some systems rely on vision-based hand tracking models trained on datasets collected in the deployed studio [24] [16], and some rely on human-robot retargeting models [62] [13] or collision avoidance models [52] trained for the particular robot at use. These systems will scale poorly as the pool of robot models expands and the variety of operating environments increases. Second, each system is created and coupled with one specific “reality”, either only in the real world or with a particular choice of simulators. For example, the HAPTIX [23] motion capture system is only developed for teleoperation in MuJoCo-based environments [56]. To facilitate large-scale data collection with simulation as well as closing sim-to-real gaps, we need teleoperation systems to operate both in virtual (with arbitrary choices of simulators) and in the real world. Finally, existing teleoperation systems are often tailored for single-operator and single-robot settings. To teach robots how to collaborate with other robot agents as well as with human agents, a teleoperation system should be designed to support multiple pilot-robot partners where the robots can physically interact with each other in a shared environment.

In this paper, we aim to set the foundation for scaling up data collection with vision-based dexterous teleoperation, by filling in the aforementioned gaps. To this end, we propose AnyTeleop, a unified and general teleoperation system (Fig. 1), which can be used for:

- **Diverse** robot arm and dexterous hand models;
- **Diverse** realities, i.e. different choices of simulators or the real world;
- Teleoperation from **diverse** geographic locations, via a browser-based web visualizer developed for remote visual feedback;
- **Diverse** camera configurations, e.g. RGB camera with or without depth, single or multiple cameras;
- **Diverse** operator-robot partnerships, e.g. two operators separately piloting two robots to collaboratively solve a manipulation task.

To achieve this goal, we first develop a general and high-performance motion retargeting library to translate human motion to robot motion in real time without learned models. Our collision avoidance module is also learning-free and powered by CUDA-based geometry queries. They can adapt to new robots given only the kinematic model, i.e., URDF files. Second, we develop a web-based viewer compatible with standard browsers, to achieve simulator-agnostic visualization and enable remote teleoperation across the internet. Third, we define a general software interface for visual-based teleoperation, which standardizes and decouples each module inside the teleoperation system. It enables smooth deployment on different simulators or real hardware.

While being very general to support many settings with a single system, our system can still achieve great performance in the experiments. For real-world teleoperation, AnyTeleop can outperform a previous system [52] designed for specific robot hardware with higher success rates on 8 out of 10 tasks proposed in their paper, using the same robot as [52]. For simulated environment teleportation, the smoother and collision-free demonstrations collected by AnyTeleop can bring better imitation learning results with higher success rates on 5 out of 6 tasks proposed in their paper, compared with a previous system [42] specifically designed for that simulator. Finally, we demonstrate that AnyTeleop can be extended to support collaborative manipulation, which to our best knowledge has neither been achieved in the literature of vision-based teleoperation nor on dexterous hands.

Our system is also packaged to be easily-deployable. The containerized design makes installation easy and frees users from handling software dependencies. We are committed to open-sourcing the system and benefiting the community.

**II. RELATED WORK**

**Vision-based Robot Teleoperation.** Recent years have witnessed an increasing interest in teleoperation of dexterous robot hands by human hands. It relies on accurate tracking of human hand motions and finger articulations. Compared to the costly wearable hand tracking solutions, such as gloves [29] [30] [37], marker-based motion capture systems [65] [31], inertia sensors [63] or VR headsets [4] [28] [49] [41], vision-based hand tracking is particularly favorable due to its low cost and low intrusion to the human operator. Early research in vision-based teleoperation focused on improving the performance of hand tracking [55] [58] [1] and mapping human hand pose to robot hand pose [24] [2] [5] [35]. Recent works have expanded the scope of teleoperating a single robot hand to complete arm-hand systems [16] [25] [64]. However, these systems are designed and engineered towards a particular robot model (e.g., Kuka arm with Allegro hand in [16], and PR2 arm with Shadow hand in [26]), and rely on retargeting or collision detection models trained for specific robot hardware (e.g., Allegro hand and XArm6) [16] [52], making them difficult to transfer to new arm-hand systems and new environments. In contrast, our system is highly modularized with a versatile hand-tracking solution compatible with an arbitrary number of cameras, and configurable
robot hand retargeting and motion generation modules for easy adaption to various robot arms and robot hand choices. This allows our system to achieve better performance compared to prior systems on various tasks while generalizing to a set of robot arm-hand systems and multiple environments.

**Teleoperation in Different Reality.** Manipulation with a dexterous robot hand is challenging due to its high degree of freedom. In recent years, dexterous robot teleoperation has been actively studied and shown promising progress in controlling a multi-fingered hand to perform manipulation tasks in the real world [4, 16, 36] by leveraging the morphological similarity between the dexterous hand and the human hand.

With the advancement of data-driven approaches for robot manipulation [12, 18], there is a growing need to collect human demonstrations in robotics. To enable easy and scalable data collection, teleoperation has also gained attention in simulated environments [60, 32, 56, 14, 8]. This provides a scalable solution to data collection by eliminating the need for real hardware, while maintaining access to oracle world information. For example, Mandlekar et al. [33] developed a crowd-sourcing platform to teleoperate robots via mobile devices as controllers. Tung et al. [57] further extended this framework to allow multi-arm collaborative teleoperation. The above frameworks rely on inertial sensors for control signals and thus are limited to parallel-gripper and simple tasks such as pick-and-place. Our system offers the ability to perform a wide range of dexterous tasks with robots of different morphologies by utilizing state-of-the-art techniques in perception, optimization, and control. In addition, AnyTeleop is designed to support teleoperation in both virtual and the real world with a unified framework.

### III. System Overview

Fig. 2 illustrates our proposed paradigms of vision-based teleoperation systems. Below we introduce the features and designs of our system which realize the paradigms.

#### A. System Features

1) **Any arm-hand.** As shown in Fig. 1, AnyTeleop is designed for arbitrary dexterous arm-hand systems that are not limited to any specific robot type.

2) **Any reality.** AnyTeleop is decoupled from specific hardware drivers or physics simulators. It can support different realities as visualized in Fig. 1.

3) **Anywhere remote teleoperation.** AnyTeleop provides a web-based visualizer to monitor the teleoperation and simulation in standard web browsers, e.g. Chrome.

4) **Any camera configuration.** AnyTeleop can consume data from both RGB and RGB-D cameras, and from either single or multiple cameras. Most importantly, it does not require extrinsic calibration as in most previous systems. This allows more flexible camera configurations and lower deployment overhead.

5) **Any number of operator-robot partnerships.** AnyTeleop supports collaborative settings where operators separately pilot two robots to collaboratively solve a manipulation task.

6) **Simple deployment.** AnyTeleop and all libraries are encapsulated as a docker image that can be downloaded and deployed on any Linux machine, which frees users from handling troublesome dependencies.

Table I compares AnyTeleop with other vision-based dexterous teleoperation systems. We compare the systems...
### Table I: Comparison of Vision-Based Teleoperation System

We compare AnyTeleop’s capabilities with related visual teleoperation system for multi-fingered dexterous robots. “No Arm” in the column of “Multiple Arms” means this system can only control hand motion but not arm-hand systems.

<table>
<thead>
<tr>
<th>Sensor Requirements</th>
<th>Robot-related Support</th>
<th>Use Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calibration Free</td>
<td>Contact Free</td>
<td>Depth Free</td>
</tr>
<tr>
<td>DexPilot [16]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Holo-Dex [4]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>DIME [5]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TeachNet [23]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Telekinesis [52]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Qin et al. [12]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MVP-Real [43]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Transteleop [26]</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Mosbach et al. [57]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>AnyTeleop</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

in three dimensions: (i) sensor requirements; (ii) robot-related support; (iii) afforded use cases. Among all these teleoperation systems, AnyTeleop is the only one which can support different robot arms and enable collaborative teleoperation. It is also one of the only two systems that can support different dexterous hands.

### B. System Design

The architecture of the teleoperation system is shown in Fig. [3]. The teleoperation server (Section IV) receives the camera stream from the driver, detects the hand pose, and then converts it to joint control commands. The client receives these commands via network communication and uses them to control a simulated or real robot. The system is designed with three key principles: modularity, communication-focused, and containerization. Modularity is achieved by implementing well-defined input-output interfaces for each sub-component, allowing for wide applicability to different robot arms, dexterous hands, cameras, and realities. Communication-focused design allows for remote and collaborative teleoperation and reduces computation requirements on the operator’s side by deploying heavy computations on a powerful server. Finally, the containerized design makes installation and deployment easier compared to other robotics systems with heavy software dependencies.

### IV. Teleoperation Server

The teleoperation server, outlined in Section III, utilizes the RGB or RGB-D data from one or multiple cameras and generates smooth and collision-free control commands for the robot arm and dexterous hand. It consists of four modules: (i) the hand pose detection module, which predicts hand wrist and finger poses from the camera stream, (ii) the detection fusion module, which integrates the results from multiple cameras, (iii) the hand pose retargeting module, which maps human hand poses to the dexterous robot hand, and (iv) the motion generation module, which produces high-frequency control signals for the robot arm. A standardized software interface is defined for all four modules to facilitate flexibility and generalizability in AnyTeleop.

#### A. Hand Pose Detection

The hand pose detection module offers a unique feature to utilize input from various camera configurations, including RGB or RGB-D cameras, and single or multiple cameras. The design principle is to leverage more information, such as depth, and additional cameras, to improve performance when available. But it can also perform the task with minimal input, i.e. a single RGB camera. The detection module has two outputs: local finger keypoint positions in the wrist frame and global 6D wrist pose in the camera frame. The finger keypoint detection only requires RGB data while the wrist pose detection can optionally use depth information to achieve better results.

**Finger Keypoint Detection.** Our finger keypoint detection utilizes MediaPipe [61], a lightweight, RGB-based hand detection tool that can operate in real-time on a CPU. The MediaPipe detector can accurately locate 3D keypoints of 21 hand-knuckle coordinates in the wrist frame and 2D keypoints on the image.

**Wrist Pose Detection from RGB-D.** We use the pixel positions of the detected keypoints to retrieve the corresponding depth values from the depth image. Then, utilizing known intrinsic camera parameters, we compute the 3D positions of the keypoints in the camera frame. The alignment of the RGB and depth images is handled by the camera driver. With the 3D keypoint positions in both the local wrist frame and global camera frame, we can estimate the wrist pose using the Perspective-n-Point (PnP) algorithm.

**Wrist Pose Detection from RGB only.** The orientation of the hand can be computed analytically from the local positions of the detected keypoints. However, determining the wrist position in the camera frame can be challenging without explicit 3D information. To enhance MediaPipe for global wrist pose estimation, we adopt the approach used in FramkMocap [48] by incorporating an additional neural
AnyTeleop is composed of four components: (i) camera driver, which captures the human hand pose in RGB or RGB-D format; (ii) teleportation server, the core component in our system, which performs hand pose detection and converts detection results to robot control commands; (iii) teleoperated robot, which is either a real robot or a simulated robot in a virtual environment; (iv) web visualizer, which enables remote visualization across the internet.

To address the second challenge, we use the SMPL-X [40] hand shape parameters predicted from the detection module, as inspired by Qin et al. [42]. During teleoperation, the true shape parameters should remain constant for a given operator, but the predicted values can contain errors during self-occlusion. We observe that larger shape parameter prediction errors often correspond to larger pose errors. To approximate the confidence score, we take the mean of the estimated shape parameters in the first N frames as a reference and compute the error between the predicted shape parameters and the reference. Implementation-wise, we require the operator to spread their fingers during the first N frames to ensure an accurate reference value of shape parameters. The fusion module then selects the relative motion captured by the camera with the highest confidence score and forwards it to the next module. In implementation, we choose N = 50.

C. Hand Pose Retargeting

The hand pose retargeting module maps the human hand pose data obtained from perception algorithms into joint positions of the teleoperated robot hand. This process is often formulated as an optimization problem [42, 16], where the difference between the keypoint vectors of the human and robot hand is minimized. The optimization can be defined as follows:

\[
\min_{q_t} \sum_{i=0}^{N} ||\alpha v_i^t - f_i(q_t)||^2 + \beta ||q_t - q_{t-1}||^2
\]

s.t. \( q_l \leq q_t \leq q_u \).

where \( q_t \) represents the joint positions of the robot hand at time step \( t \), \( v_i^t \) is the \( i \)-th keypoint vector for human hand computed from the detected finger keypoints, \( f_i(q_t) \) is the \( i \)-th forward kinematics function which takes the robot hand joint positions \( q_t \) as input and computes the \( i \)-th keypoint vector for the robot hand, \( q_l \) and \( q_u \) are the lower and upper limits of the joint position, \( \alpha \) is a scaling factor to account for hand size difference. An additional penalty term with weight \( \beta \) is included to improve temporal smoothness. When retargeting to a different morphology, such as a Dclaw in Figure 1, we need to specify the keypoint vectors mapping between the robot and human fingers manually. It is worth noting that this module only considers the robot hand.

D. Motion Generation

Given the detected wrist and hand pose, our goal is to generate smooth and collision-free motion of robot arm to reach the target Cartesian end-effector pose. Real-time motion generation methods are required to have a smooth teleoperation experience. In the prior work of [16], the robot motion is driven by Riemannian Motion Policies...
Fig. 4: **Real Robot Teleoperation Tasks.** We replicate the ten manipulation tasks proposed in Sivakumar *et al.* \[52\] using same or similar objects. Top row, left to right: Pickup Box Object, Pickup Fabric Toy, Box Rotation, Scissor Pickup, Cup Stack. Bottom row, left to right: Two Cup Stacking, Pouring Cubes onto Plate, Cup Into Plate, Open Drawer and Open Drawer and Pickup Cup.

(RMPs) \[47, 7\] that can calculate acceleration fields in real-time. However, accelerations towards a particular end-effector pose do not guarantee natural trajectories. In this work, we adopt CuRobo \[54\], a highly parallelized collision-free robot motion generation library accelerated by GPUs, to generate natural and reactive robot motion in real-time. In AnyTeleop, the motion generation module receives the Cartesian pose of the end-effector at a low frequency (25 Hz) from the hand detection and retargeting modules, and generates collision-free joint-space trajectories within joint limits at a higher frequency (120 Hz). The generated trajectories are ready for safe execution by impedance controllers on either a simulated or real robot.

<table>
<thead>
<tr>
<th>HardWare</th>
<th>GPU</th>
<th>Desktop</th>
<th>Laptop</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>RTX 3090</td>
<td>i9-10980XE</td>
<td>RTX 2070</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Profiling</th>
<th>Modules</th>
<th>Time (ms)</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hand Pose (RGB)</td>
<td>26 ± 5</td>
<td>34 ± 5</td>
<td></td>
</tr>
<tr>
<td>Hand Pose (RGB-D)</td>
<td>27 ± 5</td>
<td>35 ± 5</td>
<td></td>
</tr>
<tr>
<td>Fusion</td>
<td>1 ± 0</td>
<td>1 ± 0</td>
<td></td>
</tr>
<tr>
<td>Retargeting</td>
<td>9 ± 7</td>
<td>10 ± 9</td>
<td></td>
</tr>
<tr>
<td>Motion</td>
<td>8 ± 3</td>
<td>11 ± 5</td>
<td></td>
</tr>
</tbody>
</table>

TABLE II: **Profiling Results.** We profile different modules inside teleoperation server on both desktop and laptop. The time is measured when all teleoperation modules are run on the same computer simultaneously.

VI. SYSTEM EVALUATION

A. Profiling Analysis

We perform profiling on modules mentioned in Section IV on a desktop and a laptop. As shown in Table II the most time-consuming module is hand pose detection, which runs on a GPU for real-time inference. The designed maximum frequency for hand pose detection is 25Hz, so both the desktop and laptop can meet the requirement. Both the retargeting module and the fusion module run at the same frequency as the hand detection module due to the publisher and subscriber logic. For best performance, the motion generation module should run at 120Hz but can still work with a lower frequency. Notably, we found it difficult to achieve this throughput when running all these modules on the same computer. Luckily, with our communication-oriented design, we can run the control modules on a separate machine to achieve the best performance.

B. Real Robot Teleoperation

In this section, we will test our AnyTeleop system across a wide range of real-world tasks that covers diverse ob-
TABLE IV: Real Robot Teleoperation Results. We replicate the experiment settings and tasks in [52] and compare with [52]. For the baseline method, we use the success rate reported in their paper [52].

<table>
<thead>
<tr>
<th>Task</th>
<th>AnyTeleop</th>
<th>Telekinesis [52]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pickup Box Object</td>
<td>1.0</td>
<td>0.9</td>
</tr>
<tr>
<td>Pickup Fabric Toy</td>
<td>1.0</td>
<td>0.9</td>
</tr>
<tr>
<td>Box Rotation</td>
<td>0.6</td>
<td>0.6</td>
</tr>
<tr>
<td>Scissor Pickup</td>
<td>0.8</td>
<td>0.7</td>
</tr>
<tr>
<td>Cup Stack</td>
<td>0.9</td>
<td>0.6</td>
</tr>
<tr>
<td>Two Cup Stacking</td>
<td>0.7</td>
<td>0.3</td>
</tr>
<tr>
<td>Pouring Cubes onto Plate</td>
<td>0.7</td>
<td>0.7</td>
</tr>
<tr>
<td>Cup Into Plate</td>
<td>1.0</td>
<td>0.8</td>
</tr>
<tr>
<td>Open Drawer</td>
<td>1.0</td>
<td>0.9</td>
</tr>
<tr>
<td>Open Drawer and Pickup Object</td>
<td>0.9</td>
<td>0.6</td>
</tr>
</tbody>
</table>

The left figure visualizes the three tasks we use for both teleoperation data collection and imitation learning. The transparent object represents the goal of the task while the black arrow represents the steps of the task. The right table shows the success rate of the evaluated methods. We use ± to represent the mean and standard deviation over three random seeds. The success rate is computed from 100 trials.

VII. APPLICATIONS

A. Imitation Learning

The most important application of the proposed system is imitation learning from demonstration. We can first collect demonstrations on several dexterous manipulation tasks and then use the data to train imitation learning algorithms. In this experiment, we will show that the teleoperation data collected using our AnyTeleop can better support downstream imitation learning tasks. In the following subsection, we will first introduce the experiment setting and baseline and then discuss the experimental results.

Baseline and Comparison. To fairly compare with previous teleoperation systems, we need to align both the task setting and robot configuration precisely. It is often challenging for real-robot hardware but much easier for a simulated environment. Thus, we choose a recent vision-based teleoperation work [42] that can be used for simulated robots as our baseline. It is worth noting that we are comparing two teleoperation systems via the demonstration data collected by each system. Thus, we compared with the baseline by training the same learning algorithm on different demonstration data collected via the baseline system and our teleoperation system. We follow [42] to choose Demo Augmented Policy Gradient (DAPG) [44] as the imitation algorithm. We also compare it with a pure reinforcement learning (RL) based algorithm which does not utilize demonstrations. We provide the same dense reward for RL training as previous work [42].

Manipulation Tasks. We directly use the manipulation tasks proposed by the baseline work [42] for comparison, which include three tasks: (i) Relocate, where the robot picks an object on the table and moves it to the target position; (ii) Flip Mug, where the robot needs to rotate the mug for 90 degrees to flip it back; (iii) Open Door, where the robot needs to first rotate the lever to unlock the door, and then pull it to open the door. These tasks are visualized in the left figure of Table I. The manipulated objects in all three tasks are randomly initialized and the target position is also randomized in Relocate. Each manipulation task has two variants: the floating-hand variant and the arm-hand variant. The floating-hand is a dexterous hand without a robot arm...
that can move freely in space. The arm-hand means the hand is mounted on a robot arm with a fixed base, which is a more realistic setting.

**Demonstration Details.** For the baseline teleoperation system [42], we directly use the demonstration collected by the original authors with 50 demonstration trajectories for each task. The baseline system only utilizes a single RGB-D camera. For fairness, we also collect 50 trajectories for each task using the single camera setup. The baseline system can only handle floating hands and they propose a demonstration translation pipeline to convert the demonstration with floating hands to demonstrations with arm-hand. For our AnyTeleop, we collect demonstrations using the arm-hand setting and convert the demonstration to floating hand so that the demonstration can be used by both the floating-hand variant and arm-hand variant.

**Results and Discussion.** For each method on each task, we train policies with three different random seeds. For each policy, we evaluate it on 100 trials. More details about the success metrics can be found in [42]. As shown in Table III, the imitation learning algorithm trained on demonstration collected by AnyTeleop can outperform baseline and RL on most tasks with one exception. Compared with the demonstration collected via the baseline system, our system has two benefits that contribute to better performance in imitation learning: (i) The collected trajectory is more smooth, which means that the state-action pairs are more consistent and easier to be consumed by the network. (ii) Different from the baseline, our system explicitly supports teleoperation with arm-hand system and guarantees no self-collision. On the contrary, the baseline system utilizes retargeting to generate joint trajectory for robot arm, which may lead to several self-collision for robot arm. Thus we can observe significant performance gain of our system for manipulation tasks with arm-hand. For the flip mug task, the difficulty of collecting demonstration with arm is much larger than with a floating hand, which influences the demonstration quality.

**B. Collaborative Manipulation**

Collaborative manipulation is a key technology for the development of human-robot systems [57]. Collecting demonstration data for collaborative manipulation tasks has been a challenging task since it requires multiple operators to work together seamlessly. With our modularized and extensible system design and web-based visualization, our system enables convenient data collection on collaborative tasks, even if operators are not in the same physical location. In this section, we show that our teleoperation system can be extended to a collaborative setting where multiple operators coordinate together to perform manipulation tasks. We choose human-to-robot handover as an example as shown in Fig. 5. In this setting, operator #1 control a robot hand, and operator #2 control a human hand.

**Collaborative Teleoperation System Design.** Fig. 6 illustrates the system architecture for multi-operator collaboration, which includes two components. (i) Teleoperation Units: It is composed of a computer that is connected to at least one camera and a human operator. In each teleoperation unit, the human operator will watch the real-time visualization on a web browser and move the hand accordingly to perform manipulation tasks. (ii) Central Server: it runs the physical simulation and the web visualization server. The detection results from multiple teleoperation units are sent to the server and converted into robot control commands based on the pipeline in Section IV. Meanwhile, the web visualizer server will keep synchronized with the simulated environment and maintain the visualization resources as described in Section V.
VIII. FAILURE MODES

As illustrated on the project page, we have identified two failure modes: (i) loss of tracking during fast human hand motion, which triggers a pause and re-detection process; (ii) unreliable hand pose when the hand is in self-occlusion. For mode (i), the workaround is to instruct the operator to slow down their hand motion. The issue (ii) can be solved by incorporating multiple cameras for tasks that require significant hand rotation.

IX. CONCLUSION

In this paper, we introduced AnyTeleop, a versatile teleoperation system that can be applied to diverse robots, assorted reality, and varied camera setup, and can be operated by a flexible number of users from any geographic locations. The experiments show that AnyTeleop outperforms previous systems in both simulation and real-world scenarios while offering superior generalizability and flexibility. Our commitment to an open-source approach will facilitate further research in the field of teleoperation.

REFERENCES

[21] Jonathan Kofman, Xianghai Wu, Timothy J Luu, and


[43] Ilija Radosavovic, Tete Xiao, Stephen James, Pieter Abbeel, Jitendra Malik, and Trevor Darrell. Real-world


A. Supplementary Overview

This supplementary material provides more details, results and visualizations accompanying the main paper, including:

- More details and visualization about teleoperation server, including detection and retargeting modules;
- More details about web-based teleoperation viewer;
- Additional experimental results on system evaluation.

More visualization can be found at our project page: [http://anyteleop.com](http://anyteleop.com)

B. Teleoperation Server

In this section, we will show more intermediate results from our system, including visualization of hand pose detection results and the retargeting results of various robot hands.

**Visualization of Hand Pose Detection**
We visualize the hand pose detection results in Figure 7. We showcase five typical cases, which include: (i) a hand spreading out the fingers for teleoperation initialization, (ii) fingers facing downwards in preparation for a top-down grasp, (iii) a precision grasp using the thumb and index finger, (iv) a power grasp using all five fingers, and (v) a failure case where the hand is positioned vertically relative to the camera plane.

**Visualization of Hand Pose Retargeting**
We demonstrate the results of hand pose retargeting in Figure ???. The figure displays seven gestures being performed using four different dexterous hands.

C. Web-based Teleoperation Viewer

In this section, we demonstrate how the web-based visualizer provides accessibility and multi-view support for teleoperation through its lightweight rendering and capability to run in multiple browser windows. Figure 8 shows screenshots of the web-based visualizer when it is used to visualize the five IsaacGym tasks depicted in the Figure 1 in the main paper.

**Lightweight Rendering vs High Visual Quality.** The design of our web-based viewer prioritizes accessibility and convenience, as it can be used on any device with a browser and provides minimal but sufficient rendering capabilities for teleoperation. Although the rendering quality may not be as advanced as the original simulator viewer, simulation states can be saved for offline rendering to produce high-quality visual data. For example, in visual reinforcement learning tasks using RGB images as inputs, the rendered data can be generated using a more powerful engine such as a ray tracer after teleoperation is completed.

**Multi-View Support for Teleoperation.** In teleoperation, human operators often require a clear understanding of the spatial relationships between objects and robots to make informed decisions. This information can be provided through multi-view rendering, which is a widely used technique in previous teleoperation works [26, 42, 59]. Our web-based viewer offers multi-view support to the operator by simply opening multiple browser windows. As shown in Figure 9, an example of the operator using two views to perform a manipulation task is displayed. The operator is able to open as many windows as needed to enhance their teleoperation experience.

In this section, we examine the impact of different camera configurations on the teleoperation performance of our system, AnyTeleop, which is capable of supporting diverse configurations including RGB, RGB-D, and single or multiple cameras. Even with a minimal configuration, i.e. a single RGB camera, the system can still perform effectively. Additionally, by adding more resources, such as multiple cameras, our system can achieve better performance.

We use the *Play Piano* task implemented in IsaacGym [32] as the evaluation scenario, which requires the robot hand to press piano keys in a specific order. The task is shown in the Figure 1 in the main paper and the video in the supplementary material. To quantify performance, we introduce two task metrics: (i) completion time, i.e. the elapsed time from start to finish, and (ii) the percentage of incorrect key presses, which measures the number of incorrectly pressed keys relative to the total number of keys.

A trained operator performs the task ten times for each camera configuration. As reported in Table V, with additional information, such as depth, and increasing number of cameras, the task can be completed faster and with fewer errors, which demonstrates that our system allows users to easily trade-off between efficiency and system cost based on their use case.

<table>
<thead>
<tr>
<th>Camera Configuration</th>
<th>Completion Time</th>
<th>Error Percentage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single RGB</td>
<td>109s</td>
<td>28.1%</td>
</tr>
<tr>
<td>Single RGB-D</td>
<td>87s</td>
<td>21.8%</td>
</tr>
<tr>
<td>Two RGB-D</td>
<td>74s</td>
<td>12.5%</td>
</tr>
</tbody>
</table>

**TABLE V: Comparison of Camera Configurations.** We evaluate the teleoperation performance on the Play Piano task with different camera configurations.
Fig. 7: **Hand Pose Detection Visualization.** This figure visualizes the hand detection results, with the white bounding box highlighting the predicted area and red points marking the identified finger key points. The hand skeleton is represented by the grey lines connecting the key points. Additionally, the small grey points depict the 2D projection of 3D vertices from the SMPL-X hand model. The figure showcases five diverse cases, from left to right: (i) a hand spreading out the fingers to initiate teleoperation, (ii) fingers facing downwards in preparation for a top-down grasp, (iii) a precision grasp using the thumb and index finger, (iv) a power grasp using all five fingers, and (v) a failure scenario where the hand is positioned vertically relative to the camera plane.

Fig. 8: **Web-based Viewer.** The visualization of teleoperation process can be perform in the web-based viewer, which features the five tasks from the IsaacGym tasks shown in the Figure 1 of the main paper. The viewer utilizes the *three.js* library for real-time rendering through a web browser.

Fig. 9: **Multi-view Support with More Browser Windows.** The web-based viewer offers multiple views to enhance the operator’s understanding of the 3D object relationships. Additional views can be accessed by simply opening more browser windows.