The Thesis Committee for Steve Han
certifies that this is the approved version of the following thesis:

Demonstrations for Robots in Immersive Virtual Reality

APPROVED BY

SUPERVISING COMMITTEE:

Yuke Zhu, Supervisor

Etienne Vouga

Demonstrations for Robots in Immersive Virtual Reality

by

Steve Han

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2023

Acknowledgments

Thanks to Mingyo Seo and Yuke Zhu for the project idea. Thanks to Mingyo Seo for creating the simulation, training the baselines, and adapting the whole body control code. Thanks to Kyutae Sim for helping with the data collection. Thanks to Carlos Isaac Gonzalez and Seung Hyeon Bang for help with the real robot experiments.

Demonstrations for Robots in Immersive Virtual Reality

Steve Han, MSCompSci
The University of Texas at Austin, 2023

Supervisor: Yuke Zhu

Our world is designed by humans, for humans. This makes humanoid robots the perfect general-purpose platform to automate repetitive or dangerous tasks done by people. However, due to the complexity of humanoid robots and the shortage of demonstration data, research in robot learning for humanoids is scarce. To address these challenges, I present a VR interface named DRIVR (Demonstrations for Robots in Immersive Virtual Reality) for collecting human demonstrations for humanoid robots in both simulation and reality. The demonstrations are then used to train a baseline imitation learning algorithm that uses an underlying controller to abstract away the complexity of whole-body control. I further propose that by embedding this data collection mechanism in VR video games, we can amass a large-scale dataset of high quality human demonstrations that can drive the development of future autonomous humanoids. To illustrate the feasibility of this idea, we collect a small dataset on toy tasks in simulation and a real robot using the VR interface. We then show that the trained policy can be deployed with a reasonable success rate.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Humanoid Robotics, Imitation Learning, and Virtual Reality (VR) are some of the most exciting and rapidly developing fields in technology today. It is perhaps no surprise that the marriage of these three technologies can have far-reaching impacts. Using VR, humans can control robots in an immersive simulation, or they can remotely teleoperate robots to perform dangerous tasks. The trajectories from the human demonstrations can then be used to train Imitation Learning policies, which can enable robots to perform the tasks autonomously. While deep Reinforcement Learning methods have been successful in teaching robot to learn from repeated interactions with an environment given a reward function, the sample inefficiency, need for online interactions, and reward engineering efforts make them difficult to implement in a humanoid platform. On the other hand, imitation learning skips over the requirement to create reward functions to shape a behavior and instead lets the robot learn directly from offline demonstrations.

However, due to the complexity of humanoid robots, the lack of large-scale data for training, and the difficulty of creating an intuitive interface for humans to control robots, there has been no research on teaching humanoid

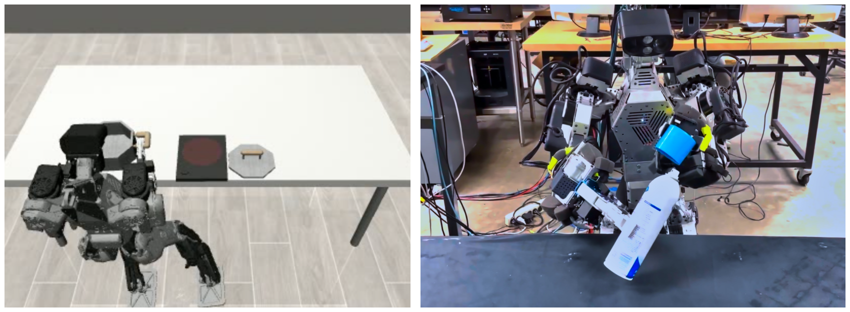


Figure 1.1: The proposed interface can be used to teleoperate a simulation environment (left) and a real robot (right).

robots to perform locomotion and bi-manipulation tasks, as far as I'm aware. There has been work to teach fixed-base robot arms through VR demonstrations [27], but humanoid robots are significantly more difficult to train due to the need to consider robot dynamics and ground reaction forces. Also, although there is a large body of work on teleoperating humanoids using VR and motion capture devices, I have not found any attempt to use the teleoperation data to train a neural network policy to control the humanoid autonomously. Furthermore, these "telepresence" systems are usually very complicated and require expensive hardware, which makes them unsuitable for scaling up data collection in a distributed manner.

In this thesis, I present a simple VR interface that uses the commonplace Oculus Quest 2 headset. I believe that compared to other humanoid teleoperation systems, the software architecture used in this project is uniquely positioned to be both adaptable to VR games and accessible to the public. Although the interface is simple, it is nonetheless powerful enough to teach a humanoid robot to perform some simple tasks. In simulation, I hypothesize that this setup can be incorporated in VR video games to massively scale up data collection. VR games contain rich interactions with the virtual world, have built-in rewards to label successes and failures, and integrate with powerful physical simulation engines. As a result, they are perfect for collecting human demonstrations to potentially teach humanoids to perform the same tasks in the real world. For example, Cook-Out is a VR game that requires players to cook in a kitchen, which is a valuable skill for humanoids to learn. In addition, on the real-robot side, we could potentially distribute the data collection by letting users with VR headsets control the robot remotely. This is similar to the RoboTurk crowd-sourcing platform for fixed-base arm robots [18]. I hope that by open-sourcing the teleoperation codebase later this summer, research in humanoid robot learning can be made more accessible.

To show that we can use the collected demonstrations to train a humanoid to perform simple tasks, we present a hierarchical approach that learns a motion policy from teleoperation demonstrations with an underlying controller. The policy outputs the desired poses of the hands, while the whole-body controller takes care of balancing the robot and tracking the desired trajectory. An overview of this design is presented in Figure 1.2. First, we take advantage of the similar kinematic structures between the robot and demonstrators and collect demonstrations using immersive VR. We only record the SE(3) pose of the hands and abstract away the other joint configurations. This lets us easily retarget human motions to the humanoid robot, which has fewer degrees of freedom in the arms. This also makes it easier to learn the

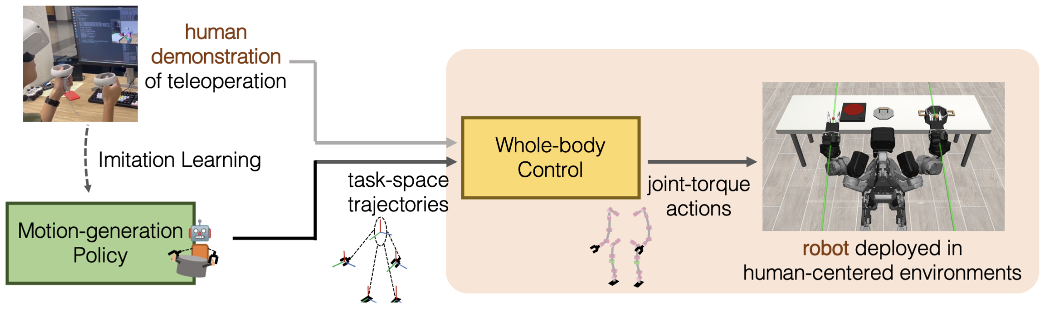


Figure 1.2: An overview of the hierarchical learning framework used in this work. Diagram credit to Mingyo. First, we get the desired hand trajectories and walking commands from the demonstrator. Second, we send these commands to the whole-body controller to produce joint-torque actions of the robot. Third, a behavior cloning policy is trained to imitate the human demonstrations.

motion, since the policy can delegate joint-space controls to the underlying controller. Then, we send the desired hand trajectories and walking commands to the whole-body controller to produce joint-torque actions of the robot. The controller prioritizes the robot's stability while tracking the hand and feet trajectories, so it may not track the trajectories perfectly. However, the human demonstrator can adapt to the tracking error by observing the effects of their actions in the VR headset. Finally, a behavior cloning policy is trained to imitate the human demonstrations. The policy needs to learn the state-action distribution of whole-body control behaviors and adapt to it in a closed-loop manner similar to the human. During deployment, the generated trajectories are passed to whole-body control just like during demonstration. The policy achieves reasonable success rate in simple simulation tasks, but it fails for more complex tasks involving manipulation while walking and precise motions. Future research directions to improve the policy will be discussed in the end.

In the below sections, I go into more details about the motivations for this project.

1.1 Humanoid Robots

Humanoid robots have gained a lot of attention in recent years. After Boston Dynamic's Atlas made headlines by jumping and dancing with human-like dexterity [9], Tesla also entered the market by developing the cheap and mass-producible humanoid named Optimus. If the cost of the robot could be kept below \$20,000 like Elon Musk promised [25], we would be entering a world where general purpose humanoids could replace humans for unsafe and repetitive tasks. Many startups are also getting a lot of funding recently to pursue humanoid robots. Apptronik is an Austin-based startup that aims to create humanoids that can work alongside people. One of their prototypes, DRACO 3, is the platform used for this thesis.

This interest in humanoids is justified by their versatility and social capabilities. They can be used as personal assistants, as companions for the elderly, as workers in factories, and as first responders in disaster zones. The morphology of humanoids enables them to easily adapt to the human-centered world that we live in. Every tool, every environment, and every task in our society is designed for the human form. It wouldn't make sense to redesign power tools or get rid of stairs for the convenience of robots, so creating robots that can take advantage of the existing infrastructure made for humans is extremely valuable. Also, as humans, it's easier to provide demonstrations to a robot that has a similar form as us, rather than having to train our brain to adapt to the morphology of the robot.

1.2 VR Teleoperation Interface

In order for imitation learning to succeed on robots, high-quality demonstrations are essential. Using a VR interface is desirable since it closes the observation and embodiment gap. Instead of looking at the robot externally, the human will have the same perception as the robot. Instead of having to map the human body's joints to the robot's joints, the human is directly controlling the robot's joints. Because of the human brain's impressive capability to adapt, the demonstrator can quickly learn to treat the arms of the robot as their own arms, and perform tasks intuitively as they do in their own body. Indeed, there are various experiments that shows that humans can start to take ownership of their virtual body, even if the body is very different from their own, if appropriate multisensory correlations are provided [14] [24].

In addition, since humanoid robots have to maintain balance while following hand trajectories, the whole-body controller may fail to track the hand trajectory perfectly. So, in order to move the robot hand to a desired position, one needs to constantly observe the effects of their actions before deciding where to move next. In experiments, we notice that humans are great at adapting to the whole-body control's tracking error in this closed-loop manner. By using the VR interface, we are essentially borrowing the human brain's power to solve the embodiment mismatch issue and perform closed-loop actions.

1.3 Scaling up Demonstration Dataset for Humanoids

Recently, we have seen the successes of training deep learning algorithms on mind-blowingly huge datasets. For example, GPT-4 [19] is believed to be trained on most texts on the internet, which enables its scalable transformer architecture to produce human-like texts. The ability of the transformer to scale can also impact the robot learning field, since many transformer architectures for robotics has been proposed with impressive results [29] [13]. Notably, researchers at Google demonstrated that their transformer architecture is able to absorb a large dataset of multi-task demonstrations from different robots and even simulations, and the resulting model is able to generalize to unseen tasks, environments, and objects in a zero-shot manner [10]. Importantly, they demonstrated that by incorporating simulation data with the real-robot data, not only was the real-robot policy not degraded, but the generalization performance also improved significantly on objects only seen in simulation. This shows that even if the simulation isn't very realistic, the model can still absorb useful information that helps with generalization. The researchers stated that their dataset "consists of over 130k episodes, which contain over 700 tasks, and was collected with a fleet of 13 robots over 17 months." The scale of this dataset is very impressive, but the collection procedure seems very costly, and it is still small in scale compared to the billions of images used to train today's image generation models. In addition, the dataset isn't publicly available, and they are collected on single-arm wheeled robots instead of humanoids.

So, how can we scale up humanoid demonstration collection in a cheaper manner than investing manpower to collect demonstrations on the real robot? One proposal is to learn from the plethora of YouTube videos online. For example, [7] shows that by watching YouTube videos of house tours, an off-policy Q-learning algorithm can learn the semantic cues in a human environment to improve navigation efficiency. To make it possible to learn dexterous manipulation skills from YouTube videos, [23] trained a neural network to retarget human finger poses from a video to a robotic hand. However, due to the difficulty of inferring 3D poses from video, the lack of proprioceptive view from the eyes of the human, and the ignorance of the person's internal state such as their joint positions, the wealth of information that online videos possess still remain out of reach for practical applications of robot learning.

Between learning from YouTube videos and collecting demonstrations on the real robot, taking advantage of the rich manipulation data from VR applications might just be practical enough to scale up humanoid data to satiate the appetite of today's deep learning methods. In VR, the human sees exactly what the robot sees, the hand poses are measured precisely by the headset and controllers, and the simulated robot provides full access to its internal states. As realistic simulation games such as Microsoft Flight Simulator rise up in popularity, we are presented with a valuable opportunity to collect high-quality human hand trajectories at scale for robot-learning research. This is not unreasonable since Meta is most likely already collecting information about the users' surroundings and actions for targeted ads [26].

Chapter 2

Background

In this chapter, the necessary background information for this work is introduced. Then, existing literature that relates to our goal will be surveyed.

2.1 Imitation Learning

In imitation learning, a task is demonstrated by a human instructor multiple times until a robot can imitate the demonstrated behavior and perform the task autonomously. There are many forms of imitation learning. The simplest form is Behavioral Cloning, where the robot learns to imitate the motion without inferring the actual goal. This becomes a supervised learning problem with the inputs being the observations and the outputs being the actions taken. This simple method is often effective [27], but it is prone to covariant shift [21], in which the errors in actions accumulate and the state deviates from the ones shown in demonstration. Another class of algorithms is inverse reinforcement learning, in which the agent tries to reverse-engineer the reward function from the demonstrations. However, this is usually computationally expensive due to the requirement to run reinforcement learning as an inner-loop.

The specific imitation learning algorithm used in our work is Behaviorial Cloning with Recurrent Neural Networks (RNN) and Gaussian Mixture Models (GMM). RNN maintains information about the past and can be viewed as injecting the inductive bias of sequentiality. Namely, it preserves time translation invariance by using the same neural network weights at each time step [5]. GMM uses a mixture of Gaussians with different means to approximate a multimodal distribution, such as the multiple ways that a human may do a specific task.

2.2 Whole-body Control

Whole-body control is a robotics approach that aims to control the entire robot's body, including its arms, legs, torso, and head, to achieve a desired task. The specific form we are using is called implicit hierarchical whole-body controller [1], which uses an implicit hierarchy of tasks to allow for different prioritization of control depending on whether the robot is performing locomotion or manipulation. For example, we would like the robot to prioritize foot motion over hand motion when the robot is walking.

Whole-body control can be formulated as finding the optimal joint acceleration $\ddot{q}^*$ and contact forces f_r^* to minimize a loss function:

$$\min_{\ddot{q},f_r} \sum_{i=1}^{n} w_i \|J_i\ddot{q} + \dot{J}_i\dot{q}_m - \ddot{x}_i^d\|^2 + w_{f_r}\|f_r^d - f_r\|^2 + \lambda_q\|\ddot{q}\|^2 + \lambda_{f_r}\|f_r\|^2$$

subject to the constraints of robot dynamics, contact, maximum reaction force, joint position limit, and torque limit. The first term in the loss function represents the task space error, or the deviation of the robot's end-effector position and orientation from the desired values. The second term represents the contact force error. The third term regularizes the joint acceleration to control the trade-off between task performance and joint motion smoothness. The fourth term regularizes the contact force to prevent damage to the robot or the environment.

2.3 Related Work

2.3.1 VR for Imitation Learning

The work that most closely relates to ours is [27], which also involves using a consumer VR headset to teleoperate a robot. They show that by using only half an hour of demonstrations collected in VR, an imitation learning algorithm can be trained to perform simple manipulation tasks such as grabbing a ball or pushing a block. However, their robot has a fixed base, so they can directly use their robot's built-in Jacobian-transpose based controller instead of a whole-body controller. Instead of streaming stereoscopic images to the headset, they use Unity to render the colorized point cloud from the robot's 3D camera to allow the user to look around freely in the VR world. They argue that since the robot's head has low degrees of freedom and moves slowly, controlling its movement using the headset's movement will cause motion sickness. Indeed, we are faced with the same problem. However, instead of using a point cloud that could look strange for the demonstrator, we simply disallowed movements of the head. This is sufficient for our tasks, and it does not seem to impact immersion. For learning, they also use a Behavior Cloning algorithm. However, they are only controlling one arm of the robot, whereas we are controlling both. They are also using the depth image as an input, but we only provide stereoscopic images. Finally, we are using an RNN to preserve information from previous time steps, while they only provide the 5 most recent end effector poses and no image history.

Another work that utilizes VR for robot learning is [3], which uses the hand-tracking capabilities of the Oculus Quest II headset to teach a robot hand dexterous manipulation skills. They first retarget the human hand joints to a robot hand with 4 fingers, then they use visual self-supervised learning combined with a simple nearest neighbor algorithm to successfully manipulate objects unseen during training. Although they are using a VR headset, they are not using stereoscopic rendering to create depth perception. Instead, they are rendering the robot's camera as a 2D video in Unity. Similarly, there is a senior thesis project teleoperating a small car in VR, and they also use a 2D video displayed through Unity [15].

2.3.2 Demonstration Interface

There are multiple ways to collect demonstrations for imitation learning in existing literature. First, a teleoperation input device with 6 degrees of freedom such as the SpaceMouse can be used to control the position and orientation of the robot's end effector [30]. However, it can be unintuitive for people to translate a 3D motion into the push and turn of a button, especially if there is a need to control 2 arms at once. In addition, since SpaceMouse controls the velocity instead of position, it requires training to perform actions involving precision. Second, humans can directly hold the robot to move it in a desired trajectory by applying force [2] [22]. This is called kinesthetic teaching, but it requires the human to come into the frame to control the robot, which becomes a problem when the policy is trained on vision data. To avoid this, we can also build a replica of the robot and move the replica manually, while the main robot follows its trajectory. For example, [28] used a low-cost replica of their bi-manipulation robot to collect demonstrations for fine manipulation tasks. To do so, a human demonstrator pushes the end-effectors of the replica robot to backdrive its joints, and the resulting joint positions are issued as commands for the actual robot to follow. While this method achieved impressive results for the fixed-base robot, they cannot handle the floating-base dynamics of humanoid platforms, which requires torque control to account for the dynamics of the robot.

Since we can manipulate a replica of the robot to create demonstrations, why can't we use our own body as this replica? After all, humanoids are design to mimic the morphology of humans. Indeed, we can directly record the kinematics of human motions [6]. Using either a camera or a motion capture system, we can measure the angular displacement of the joints precisely, and then we can map the values to the robot's joints. However, robots can have different mass distributions and degrees of freedom than humans. So, the actions that humans perform may be impossible for robots or cause them to lose balance. Therefore, learning a good mapping and using a whole-body controller to maintain balance are crucial to the success of this method.

2.3.3 Learning for Humanoids

As mentioned in the introduction, there does not seem to be prior work on training humanoid robots to perform manipulation tasks. Two works that cut close are [4], which uses Hidden Markov models to imitate a human demonstration, and [12], which uses motion planning to produce fluid transitions between locomotion, loco-manipulation, and manipulation. However, both works only consider a simple simulation with only kinematics and no dynamics, so their results are impossible to be applied to the real world.

Nonetheless, there are recent successes on learning humanoid locomotion. [17] adapts the Rapid Motor Adaptation work [16] to bipedal robots. It uses an adaptation module to enable the walking controller to adapt to changing environment in real-time. [20] uses IsaacGym simulation to train a humanoid locomotion controller that responds to velocity commands robustly. Their work seems to be a more robust version of [17] where instead of explicitly adapting to the environment, the transformer implicitly infers the environment through past actions rapidly in every time step. Indeed, they showed that their controller can even maintain balance when objects are thrown at it. There have been attempts at using Isaac Gym and RL to train humanoids to walk in simulation in a brute-force manner without success. What this work did different is that they 1. Use a two-step approach to first train an

RL policy using ground-truth observations and guidance from gait heuristics, during which they can iterate quickly to find the best reward functions and gait parameters, and then train the real RL policy with actual observations. They use the policy from the first step to guide the real policy by adding the KL divergence between the two policies in the loss function and annealing its weight as training goes on. 2. Use a transformer model that is only given the positions and velocities of the body and joints as well as previous commands, so it's practically "blind" when it comes to the physical environment. Just by observing the effect of its previous actions on the robot body, the transformer is able to quickly adapt in-context. Although in their experiment the LSTM baseline got close in success rate, the transformer is able to walk much faster and adapt to new situations much more quickly. They claim that the LSTM baseline is not able to transfer to the real robot. However, they are able to perform zero-shot sim-to-real transfer of the transformer policy since they do aggressive domain randomization from everything in the robot to the environment. Unlike a classical control algorithm, their model on the real robot is able to adapt to a backpack added on its back and even motor malfunctions (simulated by decreasing PD gains for a motor by a half).

2.3.4 Humanoid Teleoperation

There is a recent surge in interest to remotely teleoperate robots. XPRIZE hosted the AVATAR competition in 2022, in which teams compete to develop telepresence technology that allows a human to control a robot remotely to complete a set of tasks. Their overarching goal is for humans to be able to see, hear, touch, and interact with the world in the body of a robot, while the people themselves can be on the other side of the world. While most robots in the competition don't have legs, there are a few humanoids, and most teams are using VR technology to create an immersive visual experience.

There is a recent survey on teleoperation of humanoids [8], some of which competed in AVATAR. The authors also claim that humanoid teleoperation is a new field with high resource demand, so not many labs are working on it. The works they summarized include very complicated motion capture and feedback setups that tracks the movement of users' feet and relays the sense of touch to the users. The survey unifies the teleoperation systems into a framework with 6 components. First, human measurements are taken. Then, the measurements are retargeted to the robot. After a delayed communication channel, the robot executes its controller to produce low-level commands. The robot's sensory input is then retargeted to human feedback, and human teleperception is provided. This is similar to our framework, but we only have vision feedback and hand trajectory plus locomotion commands as actions. They also mention that streaming the camera images to VR can cause motion sickness during locomotion, but this could be fixed by adopting digital image stabilization techniques.

The cockpit interface for NASA's Valkyrie humanoid is a good example of a sophisticated VR teleoperation system [11]. The interface has a complex user interface that allows the operator to directly specify where to place future footsteps. There is a 3D model of the humanoid for visualization, and both a depth point cloud and a 2D RGB footage of the environment are presented to the user.

These complex teleoperation systems make sense if the goal is to create immersive telepresence or to perform critical missions in outer space. However, if the goal is to teach robots common loco-manipulation skills, these systems can be overkill. In order to scale up data collection and lower the barrier of entry for humanoid research, I opted to use a single Oculus Quest II as the interface. Although it is missing many components such as feet tracking and tactile sensing, it is enough for simple loco-manipulation tasks. In addition, VR is a rapidly developing industry, so future household headsets will mostly likely incorporate more sensors. For example, the recently-released Meta Quest Pro supports leg tracking.

Chapter 3

VR Interface

In this chapter, I'll go into the implementation details of the VR interface. The requirements for the interface are as follows:

1. It needs to report the 6-DOF poses of left and right hands as well as additional buttons for locomotion and gripper control.
2. Stereoscopic images need to be streamed and displayed on the VR headset to create depth perception for the wearer.
3. It needs to connect to both the simulation codebase written in Python and the real-robot controller written in C++.
4. The latency should be low and the computation speed should be fast so that the wearer can provide demonstrations with ease. The computation resource consumption should be low because the simulation and whole-body controller are resource-intensive.

First, an overview of the architecture is given. Then, I justify some of the design decisions. Finally, I analyze the performance of the system.

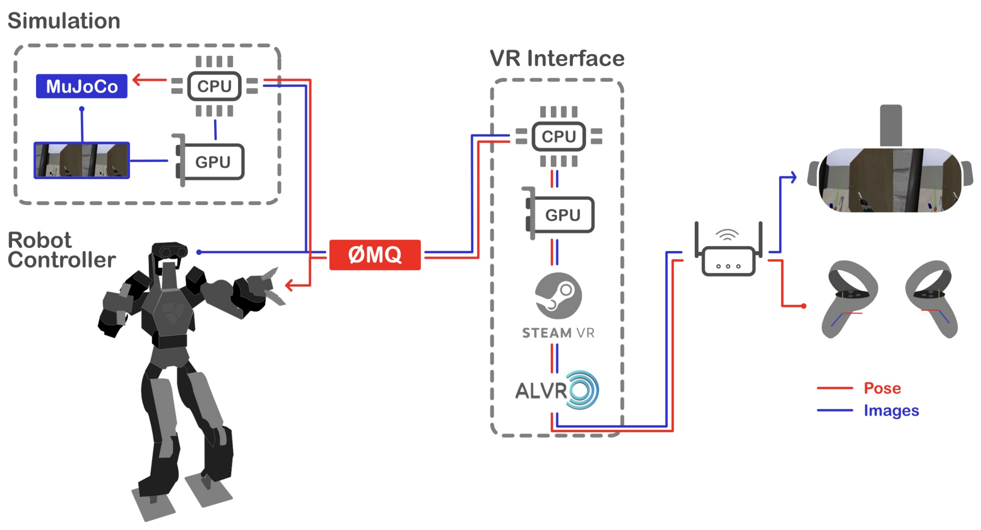


Figure 3.1: Architecture for the VR interface

3.1 Architecture

Figure 3.1 describes the software architecture. The VR interface code written in C++ runs on a separate laptop, which is connected to either the simulation desktop or the robot control station for the real robot. We use OpenVR (implemented in SteamVR) and ALVR (Air Light VR) for the communication between the laptop and the headset. OpenVR is a low-level API for VR applications designed to support a wide range of VR devices. ALVR is an open-source project that allows streaming Steam VR games from the laptop to the headset via Wi-Fi. It implements technologies such as Asynchronous Timewarp and Fixed Foveated Rendering for a smoother experience. ZMQ is an asynchronous messaging library that simplies message-passing between different programs or devices.

For the simulation, we use Mujoco with Python binding for the physical simulation and Robosuite for the objects in the scene. We first render the scene using a virtual stereoscopic camera that is adjusted to match the interpupillary distance of the VR headset. Then, the rendered pixels are copied from the GPU to CPU and sent to the VR interface using ZMQ and ethernet. The interface code listens to the images and writes them into a GPU texture used by OpenVR. Finally, SteamVR and ALVR transmits the images through a router and displays them in the VR headset. At the same time, the interface polls the VR headset for the poses of the headset and controllers through OpenVR. It transforms the controller poses to the local frame of the headset, and they are then mapped to the poses of the robot hands in the robot's local frame. The latter mapping involves aligning the rotation axes of the controller with the axes used in the simulation. Let R_t be the transformation between the VR axes and the simulation axes, R_{vr} be rotation of the controller from its natural orientation, and R_{sim} be the desired rotation to be applied to the robot hands. Their relation can be expressed as

$$R_{sim} = R_t^{-1} R_{vr} R_t$$

The transformed hand poses are then published continuously using a ZMQ pub socket. When the simulation needs a VR command, it pulls the most recent command from the queue and sends it to the whole-body controller.

The setup for the real robot is similar. However, since the robot has a lot more components to manage, I'll describe it in more details in the next chapter.

3.2 Design Choices

3.2.1 OpenVR

Initially, I created a website based on WebXR and JavaScript to stream controller poses over the internet. I wanted to improve the latency by keeping the connection within a local network, but I needed to set up HTTPS certificates in the campus lab, which was difficult. After getting tired of tunneling my connection over the internet, I decided to pursue a more low-level approach. I could use the native Oculus SDK, but this would limit the option to change VR systems in the future. Unity is a good option for cross-platform compatibility, but since I only need to stream images and controller poses, it is an overkill. I also don't think Unity exposes the low-level functionality to directly show the stereoscopic images. Since Unity uses the low-level OpenVR API, I decided to use it directly. Although a newer API called OpenXR is gaining steam, I feel that the performance benefits of OpenXR doesn't justify its complexity compared to OpenVR.

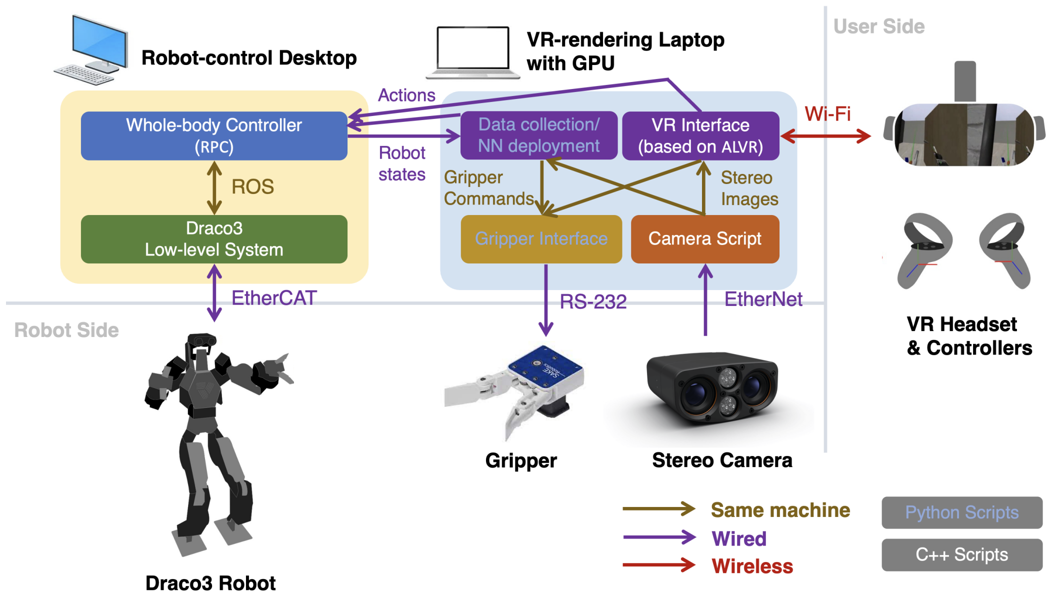
Using OpenVR has several advantages. First, it has great performance since video games depend on it, which means that it has direct support from VR headset manufacturers. Second, it delegates the work of video streaming to SteamVR-compatible software such as ALVR. Third, it works on a variety of Operating Systems and targets many VR headsets. In comparison to [3]

3.2.2 Asynchronous Message Passing

Chapter 4

Real-Robot Experiments

In this chapter, I introduce the details of how our experiments are conducted on the real robot. However, due to the complexity of the robot system, I need to first describe the infrastructure I built for collecting demonstrations and deploying the neural network policy. Specifically, I had build a real-time system to integrate the camera, grippers, whole-body controller, and the VR headset.



Robot-control Desktop
VR-rendering Laptop with GPU
User Side
Actions
Wi-Fi
Whole-body Controller (RPC)
Data collection/ NN deployment
VR Interface (based on ALVR)
Robot states
ROS
Gripper Commands
Stereo Images
Draco3 Low-level System
Gripper Interface
Camera Script
EtherCAT
RS-232
EtherNet
Robot Side
VR Headset & Controllers
Gripper
Stereo Camera
Draco3 Robot
Same machine
Wired
Wireless
Python Scripts
C++ Scripts

Figure 4.1:

Chapter 5

Future Works

Immersive demonstration with force feedback

https://arxiv.org/pdf/2301.09157.pdf

training techniques used in VR fixed-base

representation learning holodex

Appendices

Appendix A

Lerma's Appendix

The source LaTeX file for this document is no longer quoted in its entirety in the output document. A LaTeX file can include its own source by using the command `\verbatiminput{\jobname}`.

Appendix B

My Appendix #2

B.1 The First Section

This is the first section. This is the second appendix.

B.2 The Second Section

This is the second section of the second appendix.

B.2.1 The First Subsection of the Second Section

This is the first subsection of the second section of the second appendix.

B.2.2 The Second Subsection of the Second Section

This is the second subsection of the second section of the second appendix.

B.2.2.1 The First Subsubsection of the Second Subsection of the Second Section

This is the first subsubsection of the second subsection of the second section of the second appendix.

B.2.2.2 The Second Subsubsection of the Second Subsection of the Second Section

This is the second subsubsection of the second subsection of the second section of the second appendix.

Appendix C

My Appendix #3

C.1 The First Section

This is the first section. This is the third appendix.

C.2 The Second Section

This is the second section of the third appendix.

Bibliography

[1] Junhyeok Ahn, Steven Jens Jorgensen, Seung Hyeon Bang, and Luis Sentis. Versatile locomotion planning and control for humanoid robots. *Frontiers in Robotics and AI*, 8, 2021.

[2] Barış Akgün, Maya Cakmak, Karl Jiang, and Andrea Lockerd Thomaz. Keyframe-based learning from demonstration. *International Journal of Social Robotics*, 4:343–355, 2012.

[3] Sridhar Pandian Arunachalam, Irmak Güzey, Soumith Chintala, and Lerrel Pinto. Holo-dex: Teaching dexterity with immersive mixed reality, 2022.

[4] Tamim Asfour, Florian Gyarfas, Pedram Azad, and Rudiger Dillmann. Imitation learning of dual-arm manipulation tasks in humanoid robots. In *2006 6th IEEE-RAS International Conference on Humanoid Robots*, pages 40–47, 2006.

[5] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess,

Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks, 2018.

[6] A. Billard and D. Grollman. Robot learning by demonstration. *Scholarpedia*, 8(12):3824, 2013. revision #138061.

[7] Matthew Chang, Arjun Gupta, and Saurabh Gupta. Semantic visual navigation by watching youtube videos, 2020.

[8] Kourosh Darvish, Luigi Penco, Joao Ramos, Rafael Cisneros, Jerry Pratt, Eiichi Yoshida, Serena Ivaldi, and Daniele Pucci. Teleoperation of humanoid robots: A survey, 2023.

[9] Boston Dynamics, YouTube. Do you love me? `https://www.youtube.com/watch?v=fn3KWM1kuAw`, 2020.

[10] Anthony Brohan et al. Rt-1: Robotics transformer for real-world control at scale, 2022.

[11] Steven Jens Jorgensen et al. Cockpit interface for locomotion and manipulation control of the nasa valkyrie humanoid in virtual reality (vr), 2022.

[12] Paolo Ferrari, Marco Cognetti, and Giuseppe Oriolo. Humanoid whole-body planning for loco-manipulation tasks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4741–4746, 2017.

[13] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot manipulation with multimodal prompts, 2022.

[14] Konstantina Kilteni, Antonella Maselli, Konrad P. Kording, and Mel Slater. Over my fake body: body ownership illusions for studying the multisensory basis of own-body perception. *Frontiers in Human Neuroscience*, 9, 2015.

[15] Alexis Koopmann, Kressa Fox, Phillip Raich, and Jenna Webster. Virtual reality teleoperation robot. `https://my.ece.utah.edu/~kstevens/4710/reports/vr-robot.pdf`.

[16] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots, 2021.

[17] Ashish Kumar, Zhongyu Li, Jun Zeng, Deepak Pathak, Koushil Sreenath, and Jitendra Malik. Adapting rapid motor adaptation for bipedal robots, 2022.

[18] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, Silvio Savarese, and Li Fei-Fei. Roboturk: A crowdsourcing platform for robotic skill learning through imitation, 2018.

[19] OpenAI. Gpt-4 technical report, 2023.

[20] Ilija Radosavovic, Tete Xiao, Bike Zhang, Trevor Darrell, Jitendra Malik, and Koushil Sreenath. Learning humanoid locomotion with transformers, 2023.

[21] Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning, 2011.

[22] John Schulman, Jonathan Ho, Cameron Lee, and P. Abbeel. Learning from demonstrations through the use of non-rigid registration. In *International Symposium of Robotics Research*, 2013.

[23] Aravind Sivakumar, Kenneth Shaw, and Deepak Pathak. Robotic telekinesis: Learning a robotic hand imitator by watching humans on youtube, 2022.

[24] Mel Slater, Daniel Pérez Marcos, Henrik Ehrsson, and Maria Sanchez-Vives. Towards a digital body: the virtual arm illusion. *Frontiers in Human Neuroscience*, 2, 2008.

[25] Tesla, YouTube. Tesla AI day 2022. `https://www.youtube.com/watch?v=fn3KWM1kuAw`, 2022.

[26] XpertVR. Vr data collection: Traditional and contemporary data types. `https://xpertvr.ca/vr-data-collection/`, 2022.

[27] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation, 2018.

[28] Tony Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023.

[29] Yifeng Zhu, Abhishek Joshi, Peter Stone, and Yuke Zhu. Viola: Imitation learning for vision-based manipulation with object proposal priors, 2023.

[30] Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning, 2022.