2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Intuitive Robot Integration via Virtual Reality Workspaces

Minh Q. Tram, Joseph M. Cloud, and William J. Beksi

***Abstract*— As robots become increasingly prominent in diverse industrial settings, the desire for an accessible and reliable system has correspondingly increased. Yet, the task of meaningfully assessing the feasibility of introducing a new robotic component, or adding more robots into an existing infrastructure, remains a challenge. This is due to both the logistics of acquiring a robot and the need for expert knowledge in setting it up. In this paper, we address these concerns by developing a purely virtual simulation of a robotic system. Our proposed framework enables natural human-robot interaction through a visually immersive representation of the workspace. The main advantages of our approach are the following: (i) independence from a physical system, (ii) flexibility in defining the workspace and robotic tasks, and (iii) an intuitive interaction between the operator and the simulated environment. Not only does our system provide an enhanced understanding of 3D space to the operator, but it also encourages a hands-on way to perform robot programming. We evaluate the effectiveness of our method in applying novel automation assignments by training a robot in virtual reality and then executing the task on a real robot.**

***Index Terms*— Virtual Reality and Interfaces; Human-Centered Automation; Human-Robot Collaboration**

I. Introduction

Robots have become the cornerstone of many industrial operations due to their efficiency, productivity, and reliability. The dependency on robots is ultimately rooted in the need for an autonomous workforce that can achieve high throughput with low downtime in order to meet increasing production demands. With applications spread across large industries (e.g., agricultural operations, automotive manufacturing, pharmaceutical packaging, etc.) robots have become irreplaceable and serve a critical role in the supply chain workflow. This aspect is further emphasized during times of national emergency such as the recent COVID-19 pandemic [1]. Thus, the demand for more robust and highly-integrated robotic systems increases as the industrial sector intensifies its growth in automation.

Although the deployment of robots continues to rise, the task of determining the feasibility of using a new robotic component, either in a novel work environment or adding more robots to collaborate within an existing infrastructure, remains extremely difficult [2]. Acquiring and implementing capable industrial robots is often undesirable for small and medium-sized enterprises. This is due to high initial investment costs and a nontrivial integration process that requires expert knowledge with no guarantee of profitable returns. Meanwhile, the expanding virtual, augmented, and mixed reality (VAMR) industry has yielded incredibly useful tools and techniques to enhance the robotic development process.

The authors are with the Department of Computer Science and Engineering, The University of Texas at Arlington, Arlington, TX, USA. Emails: minh.tram@mavs.uta.edu, joe.cloud@mavs.uta.edu, william.beksi@uta.edu.

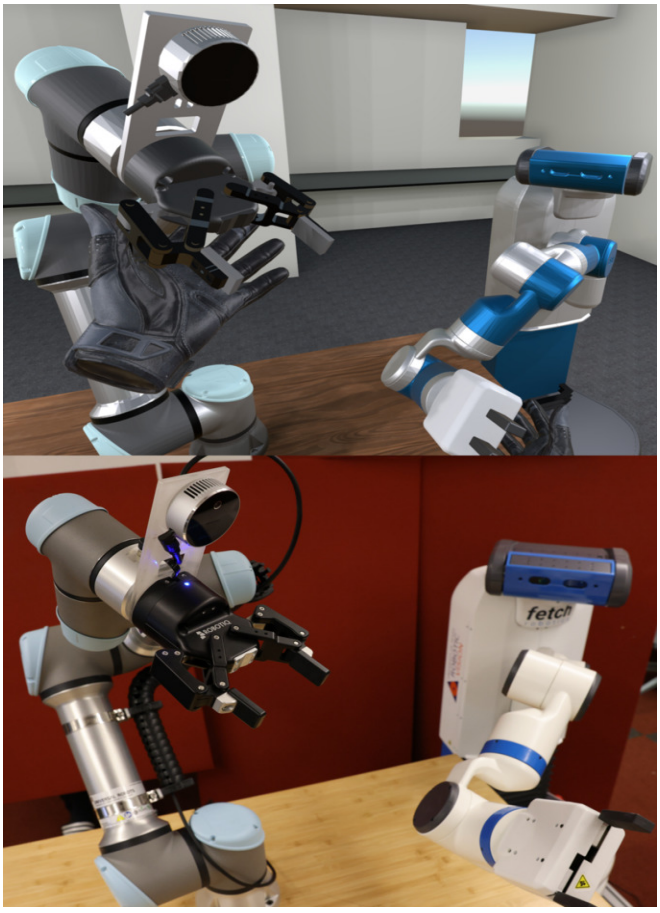


Fig. 1: A sample scene captured in our VR workspace from the point of view of the operator. The operator can directly interact with individual robot joints simply by grabbing or pushing them as they normally would in a real environment.

VAMR provides an *intuitive understanding* of 3D space while simultaneously providing *additional information* to the operator. This leads to an overall *enhanced* operator experience and *encourages* a hands-on approach to robot programming [3]. Nonetheless, VAMR-based human-robot interaction (HRI) research is generally focused on teleoperation for task-specific applications [4], [5], human-in-the-loop digital twins [6], [7], or machine learning with densely overlayed information interfaces [8], [9], [10]. However, these frameworks have the following *disadvantages*: (i) an assumption of a pre-installed, functioning robotic system, (ii) a lack of support for high-dexterity interaction, and (iii) a focus on single-robot use cases.

979-8-3503-2365-8/23/$31.00 ©2023 IEEE

In this work, we propose a simulated robotic environment capable of representing and integrating operator-configurable workspaces for *multiple* collaborative robots, Fig. 1. The primary objective of this research is to create a virtual reality (VR) workspace that allows an operator to *immersively explore* and *efficiently assess* the integration of a desired robotic system into a physical environment. We also aim to make this work a *foundation* for future robot programming and operator training. In summary, our contributions are the following.

- We eliminate the dependency on a physical robotic system by moving visualization and interaction into a purely virtual environment.
- We simulate multiple collaborative robots in realistic setup and usage scenarios (e.g., direct imitation learning, learning from demonstration).
- We provide operators with an intuitive and informative representation and control of virtual robots.

The source code and Docker image associated with this project are publicly available at [11].

The remainder of the paper is organized as follows. We provide an overview of related research in Section II. The details of our system are presented in Section III. Our evaluation use cases and results are discussed in Section IV. In Section V, we conclude and provide directions for future work.

II. Related Work

The practice of leveraging VAMR to enhance the experience of robot programming, and provide interaction and information through either see-through displays or head-mounted displays (HMD), is very popular. However, the use of augmented or mixed reality far exceeds that of VR. Surveys have shown that for effective HRI, it is crucial to provide sufficient information not only to the robot, but also to the operator [12]. To this extent, the following common setup is favorable for many researchers: (i) an overlaying interface, provided by VAMR, communicating with real robot hardware through a Robot Operating System (ROS) [13] backend; (ii) motion planning delegated to either their own implementation of forward kinematics (FK) and inverse kinematics (IK), or MoveIt [14], a well-known robot motion planning framework.

Visualizing a planned robot action, before it executes, plays an important role in a collaborative workspace in terms of human safety and correct execution. Quintero *et al.* [15] highlighted the advantages of having additional information and visualization for robot motion planning, while remaining relatively free from the use of any handheld devices, by facilitated on-board hand tracking of their HMD using the Microsoft HoloLens. They also explored the use of multiple instruction input sources, such as speech and gesture, as an alternative means of robot control. Perez *et al.* [16] demonstrated a different, more VR-oriented approach, where a warehouse was projected onto the operator's point of view by the use of 3D scanners and scene mapping through post-processing via third-party software. Control of their robot was rudimentary in that it indirectly communicated to the manufacturer's interface through a VR headset rather than directly manipulating the scene.

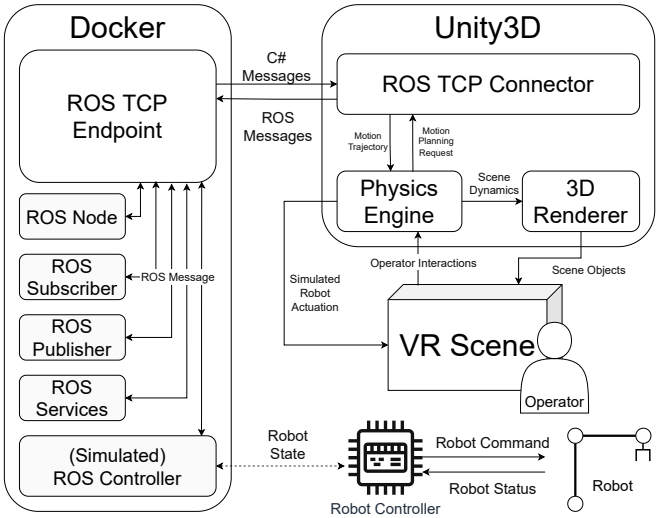


Fig. 2: An overview of the system components including the underlying communication and rendering pipelines.

In general, related work on this topic has one or more of the following objectives: (i) programming or operating a robot through VAMR [17], [15], [18], [19], [20], or (ii) leveraging VAMR as a foundation for other purposes such as imitation learning [8], [10]. There has also been significant effort made towards the use of VAMR to analyze complex behaviors of multiple collaborative robots (e.g., if they can accomplish a common goal with human partners rather than working around them). These works often target mobile rather than stationary robots [21], [22], and they typically use augmented or mixed reality. In contrast, our work is focused solely on utilizing VR as a simulation, training, and integration analysis framework without the need for physical robots. Nevertheless, our system is still capable of transferring execution over to a real platform. Moreover, we prioritize direct hands-to-object interactions rather than alternative means of control. This allows us to emulate ultra realistic interplay between the operator and the robots.

III. Virtual Reality Robotic Workspace

In this section, we present our virtual reality robotic workspace (VRRW) architecture and how each component facilitates the proposed framework.

A. System Overview and Design Choices

Our VRRW is comprised of the following three major components: (i) VR HMD, (ii) VR rendering software, and (iii) robotic simulation and planning backend. This section describes each component and the functionality it provides for our proposed approach. Fig. 2 details the communication pipeline across the hardware and software components.

For the VR headset, we developed our system using the Steam Valve Index [23]. The choice of headset is dependent on the display resolution and refresh rates. This is a vital

point of consideration since we aim to reduce the effects of initial real-to-sim and sim-to-real transitions such as motion sickness and disorientation. A high-resolution display (1440×1600 pixels per eye), high refresh rate (up to 144 Hz), low pixel illumination periods (0.330 ms to 0.530 ms), and accurate live tracking of the headset position all contribute to reducing these VR-related issues.

We utilized the Unity [24] game engine as our environment rendering software. In recent years, due to an increased interest in VAMR for robotics, Unity development has branched off into providing a software suite capable of rendering and representing robots and robotic environments. This software suite, commonly known as the Unity Robotics Hub[1], not only provides proper two-way communication between Unity and ROS, but it also contributes to how robots are represented within Unity itself.

For robot simulation and motion planning, we made use of ROS 1 Noetic running on Ubuntu 20.04 from a Docker image. We loosely use the term *simulation* since we do not run a full-scale Gazebo [25] simulation where the simulated robot is practically indistinguishable from a real setup, nor are we communicating with the official simulation backend. Instead, ROS serves the following three purposes: (i) communication, (ii) joint-state correspondence, and (iii) motion planning through MoveIt.

B. Unity 3D Game Engine

Unity3D (Unity) was originally a cross-platform game engine targeting developers rather than VAMR roboticists. As VAMR usage in the gaming industry dramatically increased, Unity has developed a sophisticated infrastructure to accommodate high-fidelity games. This includes both visual and interactive support, along with the addition of more VAMR devices from different manufacturers. Consequently, VAMR software diverges into other categories beyond just games as it is now capable enough to render visually appealing and physically accurate environments. Most prominently, Unity has been used for architecture model tours, automotive showrooms, cinematic works, and more recently robotic development.

Unity provides three primary tools for robotic development. The first tool is a graphic rendering framework capable of supporting VR devices. Not only is Unity prepackaged with support for in demand, commercially available VR headsets, but the game engine itself is also very capable of rendering 3D graphics. This allows us to display the high-fidelity rendering of desired workspaces and correctly communicate and send frames to the operator's HMD without having the burden of low-level graphics programming and controller communication.

The second tool is the standardization of ROS communication messages. ROS is written in C++ (or Python) and communicates through TCP via standardized serializable messages, while Unity is written in C#. Previously, this presented difficulties as developers needed to handle network communication with discrepancies between ROS and Unity messaging standards. This was done using *ROSBridge* (aka *ROS#*) [26], [27]. *ROS#* has been further developed by Unity into a *ROS TCP Connector* and a *ROS TCP Endpoint*, and integrated into their software suite as additional packages.

[1]https://github.com/Unity-Technologies/Unity-Robotics-Hub

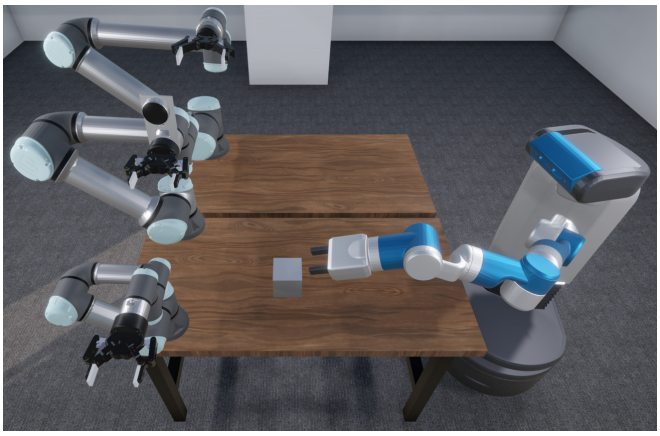


Fig. 3: A sample virtual workspace with multiple robot variants generated from URDFs. On the left is a Universal Robots UR3e, UR5e, and UR10e, each equipped with a different Robotiq 2-Finger (85 mm or 140 mm stroke) end-effector and sensor. On the right is a Fetch Robotics mobile manipulator. The gray cube on the table is 10 cm in all dimensions.

Support for direct importation of the Unified Robot Description Format (URDF) is the third tool. URDF is a standardized XML format that represents a robot model along with its articulation. More specifically, Unity will correctly parse and define the geometry, visual meshes, kinetics, and dynamics attribute of any robot given its URDF description. Subsequent robot dynamics and kinematics are handled by PhysX 4.0 [28], a full-featured physics engine that Unity physics is based on.

Internally, each robot joint generated by the URDF description will spawn with an appropriate articulation type (i.e., fixed, prismatic, revolute, or spherical) and it will be governed by their respective joint-drive properties. These properties are defined as

$$\text{Effect} = \text{Stiffness} \cdot (\Delta\text{Position}) - \text{Damping} \cdot (\Delta\text{Velocity}), \quad (1)$$

where Effect is the force or torque applied to the joint in any given time frame, ΔPosition is the difference between a set joint target and current joint angle, and ΔVelocity is the difference between the set-joint velocity and current-joint velocity. Thus, if the stiffness is zero, then the joint becomes a velocity driven joint. Conversely, if damping is zero, then the joint will only attempt to reach a position driven joint. The forward and inverse kinematic chain will not be affected since physical interaction and articulation are handled as a chain defined by the URDF, and Unity respects that definition.

C. Scene and Robot Description

While it is desirable to capture as many physical and visual dynamics of a workspace as possible, such a task is often infeasible due to the infinitely complicated and chaotic

nature of real-world environments. We strive to achieve a balance between a visually attractive and intuitive workspace, and still retain a representative physical interaction between the robot and its surrounding simulated environment. To do this, we present a set of novel solutions for defining unique workspaces.

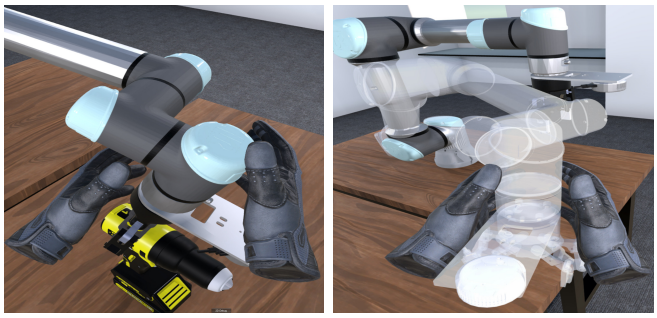
Unity, as a 3D game engine, natively supports applicable 3D modeling and rendering formats. Leveraging this feature, developers can define their workspace to the level of detail they want with the help of commercial 3D modeling software. These models can then be manually or programmatically imported into Unity for visualization. Furthermore, simulated assets can be redefined and reused on demand to allow flexible customization of an object's physical and visual characteristics. Alternatively, rather than defining the entire scene as an exported 3D scene, the developer may also export individual 3D objects (e.g., objects from relevant datasets) and programmatically generate them on demand into the scene. This is useful for data generation and machine learning use cases as the base scene can remain static for many iterations, while the distribution and orientation of the objects can be changed. These techniques can also be useful for larger scene definitions such as room-scale or house-scale scenes targeting mobile robots.

Our preferred approach is to define robots in URDF, and then let Unity handle the 3D generation parsing (Section III-B). Traditionally, to describe a robot in Unity, developers must manually create and attach all the individual links and joints of the robot while keeping track of their physical and articulation properties. Hence, two unique, but semantically identical robot descriptions, are required for Unity to work properly with ROS. Unity now supports URDF parsing, thus allowing robot descriptions to be directly imported into the desired simulated scene. Furthermore, ROS supports the use of XML macros. This allows developers to define their desired robots once and then subsequently reference that definition in other robot descriptions. As a result, large and complicated robot descriptions (e.g., numerous robots with multiple end-effectors) are more achievable as shown in Fig. 3.

Lastly, once imported into Unity, a robot model can now be saved directly as an asset. This further reduces the complexity of introducing additional robots into a simulated scene. Moreover, it decreases the load time for extensive robot collaboration projects since Unity does not need to sequentially parse the URDFs and reconstruct the robot kinematic chain at every launch. Instead, it just instantiates the complete, pre-generated robot into the scene on the fly.

D. ROS Backend and Motion Planning

We utilize ROS 1 Noetic due to its wide support for a range of industrial robots from various manufacturers (e.g., Universal Robots, Fetch Robotics, Clearpath Robotics, etc.), as well as a number of end-effector providers (e.g., Robotiq). Although, we considered moving to ROS 2 Humble and Rolling for this work, we ultimately decided to stay with ROS 1 due to incomplete robot support in ROS 2. ROS 1 Noetic is modern enough to support Python 3, while still providing proper support for the majority of the robots active in industry. Furthermore, although ROS 2 is officially supported for Unity packages, it is still under heavy development and continues to roll out bug fixes and updates.



(a) Free-drive mode (b) Ghost-drive mode

Fig. 4: Different modes of robot operation. (a) Direct-body interaction between the controlling operator and the body of the virtual robot to pull it along. (b) Ghost images indicating and recording where the robot will be without actually moving the body of the robot.

Motion planning is broken up into the following categories: planning from Unity and planning from MoveIt. These categories are separate, but they communicate in between the ROS and Unity backends. Fig. 4 shows the point of view of the operator controlling the simulated robot. From within the Unity visual scene, the operator can directly drag individual joints into position thus performing FK or pseudo-FK, whereby they can specify if the robot should follow along (free-drive mode). Conversely, the operator can overlay a representative semi-transparent double to indicate the intended final pose (ghost-drive mode). The operator can also exclusively specify the end-effector pose (position and orientation) and request IK from MoveIt.

The underlying motion planning is configurable, commonly employing the Open Motion Planning Library [29], a collection of state-of-the-art motion planning algorithms. FK is therefore trivial and can be actuated from within Unity itself, while more difficult motion planning can be delegated to MoveIt. The position of the robot is constrained by the state provided by the fake joint controller generated by MoveIt. This prevents discrepancies between ROS joint states and Unity joint states. Moreover, it is coupled with the robot representation between Unity and ROS to ensure synchronous behavior.

IV. Evaluation

In this section, we showcase the capability of our proposed VRRW to visualize and train simple robotic tasks. To demonstrate the ability to train a robot in VR and then transfer the training to a real robot, we utilize a physical robot setup. The simulation directly mirrors our real workspace and consists of a Universal Robots (UR) UR5e with a Robotiq 2F-85 2-Finger gripper attached to its tool port. This adds an additional layer of software communication between the ROS backend, the physical robot, and Unity, which is made

possible using our open-source UR-Robotiq integrated driver [30].

A. Use Cases

We present the following use cases for VRRW. In the first use case, we perform direct imitation learning on a robot solely through virtual interaction for a pick-and-place task. In the second use case, we generalize the pick-and-place behavior by learning from a virtual demonstration using a framework called dynamic movement primitives (DMPs) [31].

1) Virtual Direct Imitation Learning: This use case targets offline, exact replay of recorded robot motion provided by an operator directly manipulating the body of the robot. It should not be confused with other forms of learning from demonstration where only the initial motion is recorded and fed to a machine learning algorithm to generate a different, but behaviorally similar, trajectory. These direct prerecorded robot motions are often used as part of a larger, collaborative workflow where individual robots repeatedly perform their assigned task. Therefore, while the individual instructions may be simple, when enough of them are properly orchestrated a collection of mutually dependent motions can achieve complex results.

A standard, high-level definition of the robot instruction set for a pick-and-place task can be enumerated as follows.

i) Move robot from current position to near-pick position
ii) Move gripper to pick position
iii) Close (or activate) gripper
iv) Return to near-pick position for clearance (optional)
v) Move to final position
vi) Open (or deactivate) gripper

An example of a system exercising this use case is an industrial assembly line (e.g., automotive factory). The independent robot tasks can be as simple as moving an object from one location to another location, i.e., picking and placing tools or parts. While the individual robot tasks are straightforward, their end result is an intricately assembled vehicle. Thus, to demonstrate the efficacy of our workspace in terms of the ability to train rudimentary robot tasks from VR, we selected the job of programming a pick-and-place instruction. From within the VRRW, an operator will manually set and move the simulated arm through a desired trajectory. This recorded instruction will be replayed using our real, identical setup to pick up a similar object in the VRRW.

2) Virtual Learning from Demonstration: In this use case, we save the interactions between the VR operator and the robot as trajectories in the Unity environment to facilitate learning from demonstration [32]. To do this, we make use of motor primitives, i.e., complex sequences of muscle movements that have been theorized by neuro-biologists to be composed of *building block* movements. DMPs attempt to present this motor primitive theory within an elegant mathematical framework represented by a spring-damper system [33], [31]. Concretely, a DMP system is parameterized by the start and goal locations, desired velocities, and additional forcing terms that can be appended to perturb the behavior in response to arbitrary stimuli (e.g., sensed obstacles).

Within our VRRW, we leverage DMPs to learn trajectories demonstrated by the VR operator. This is done by using the handheld controller to provide the robot with generalized virtual skills, which can then be executed in both the virtual and real environments. Formally, a DMP is defined as

$$\begin{aligned} \tau \dot{v} &= K(g-x) - Dv + (g-x_0)f(s), \\ \tau \dot{x} &= v, \end{aligned} \tag{2}$$

where $x, v \in \mathbb{R}$ are the position and velocity, $x_0, g \in \mathbb{R}$ are the initial and goal positions, and $K, D \in \mathbb{R}^+$ are constants for the spring and damping terms. In (2), $D = 2\sqrt{K}$ to render the system critically damped, τ is a positive speed scaling factor, and f is the s dependent forcing function to be learned. Exponentially decaying from 1 to 0, s abstracts away time using the canonical system, i.e.,

$$\tau \dot{s} = -\alpha s. \tag{3}$$

The forcing term is written as

$$f(s) = \frac{\sum_{i=0}^{N} \omega_i \psi_i(s)}{\sum_{i=0}^{N} \psi_i(s)} s, \tag{4}$$

where N Gaussian basis functions, represented by ψ_i, are sum weighted against the learned weights ω. With this, (2) can be rewritten to calculate the target forces. Then, the weights can be computed using locally-weighted regression. Once the weights have been learned, we can then execute the primitive in novel contexts. We demonstrate this capability on a pick-and-place task as displayed in Fig. 5.

B. Discussion

In the pick-and-place task, we focused on the motion of the robot itself rather than the actuation of the gripper. We were able to manipulate the simulated robot directly using natural hand motions (e.g., holding or pushing) on specific parts of the robot. The trajectory of the robot was recorded as a function of time for each subtask. We then directly executed the recorded trajectories on a physically identical robot. Fig. 6 shows the correspondence and dynamic adjustments between the virtually recorded and DMP-generated trajectories.

Being able to record robot trajectories and other relevant joint state information naturally allows us to capture training data for machine learning purposes. Using the joint state information collected exclusively from our VRRW, we were able to provide sufficient training information for a DMP framework to adapt a model capable of recreating characteristically similar motions with a variable goal state. The DMP was able to learn from our VR demonstration and produce an appropriate new trajectory. The general form of the motion was retained while being executed from different start and end joint states. Moreover, the execution of the generated trajectory on the real robot exhibited similar motion and velocity behavior.

To summarize, the results of the physical execution indicates that there was a proper transition between the simulation recorded joint states and the physical joint states, even

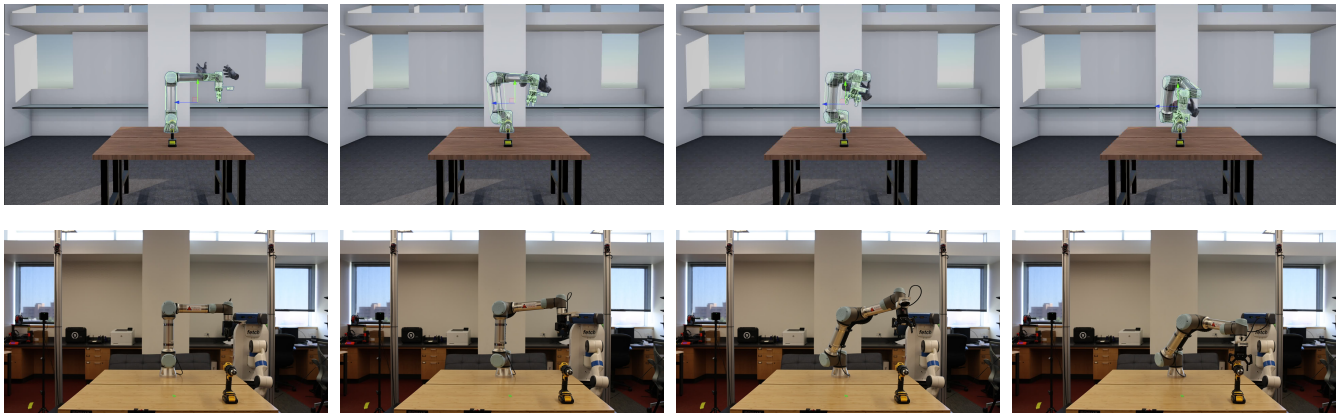
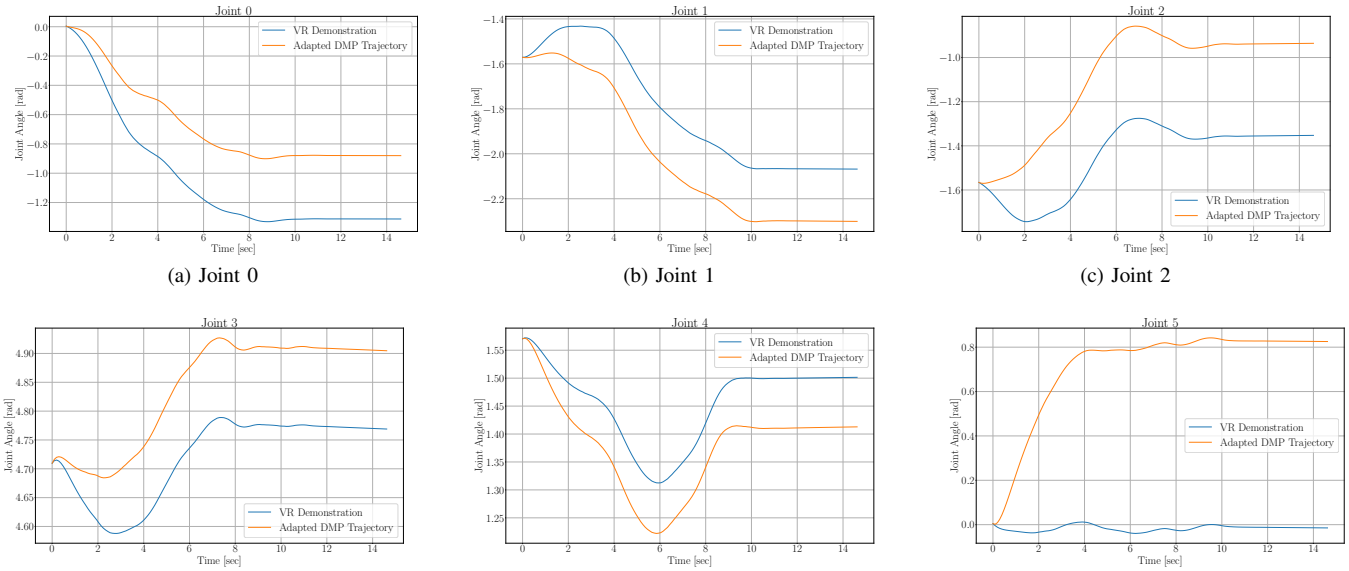


Fig. 5: The temporal execution of the robot pick-and-place task (left to right). The simulation trajectory was captured using the VRRW (top row). The second row depicts the DMP execution of the trajectory. Note the novel location of the object (yellow cordless drill).



(a) Joint 0 (b) Joint 1 (c) Joint 2

(d) Joint 3 (e) Joint 4 (f) Joint 5

Fig. 6: The UR5e joint trajectories for the virtually trained DMP.

though there was a slight amount of jitter due to the discrete time step recording of the simulated trajectory. Overall, the motion characteristics of the pick-and-place demonstration were preserved and the task was successfully executed without any deviations from the learned trajectory or object collisions in the workspace. This shows the ability of our system to provide meaningful robot access and evaluation. Furthermore, it allows operators to properly interact with and assess robot integration and programming without the constraints of obtaining or being in proximity of the physical robot.

V. Conclusion and Future Work

In this paper, we introduced a novel VR framework for easing the burden of robot integration and programming. Our VRRW is capable of simulating multiple robots in a visually captivating and intuitively interactable workspace. We described the architecture of our system and evaluated its effectiveness against various robot programming scenarios. Moreover, we showed the ability of VRRW to simulate desired workspaces and their accompanying robots, as well as its consistency in simulation to reality transference. For future work, we have additional use cases (e.g., educational outreach, workforce training, etc.) where VRRW can act as a foundation for providing access to a robotic workspace. In particular, we aim to further increase the quality of the visualization and interaction capabilities of our current approach thus allowing for more natural and dexterous interactions between operators and virtual environments.

Acknowledgments

The authors acknowledge Stanley Black & Decker for providing the power tools used to obtain the research results reported within this paper. Joseph M. Cloud was supported by a National Science Foundation Graduate Research Fellowships Program grant (#1746052).

REFERENCES

[1] Y. Shen, D. Guo, F. Long, L. A. Mateos, H. Ding, Z. Xiu, R. B. Hellman, A. King, S. Chen, C. Zhang, and H. Tan, "Robots under covid-19 pandemic: A comprehensive survey," *IEEE Access*, vol. 9, pp. 1590–1615, 2021.

[2] L. Sanneman, C. Fourie, and J. A. Shah, "The state of industrial robotics: Emerging technologies, challenges, and key research directions," *Foundations and Trends in Robotics*, vol. 8, no. 3, pp. 225–306, 2021.

[3] S. Arévalo Arboleda, T. Dierks, F. Rücker, and J. Gerken, "There's more than meets the eye: Enhancing robot control through augmented visual cues," in *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 104–106.

[4] Z. Gharaybeh, H. Chizeck, and A. Stewart, "Telerobotic control in virtual reality," in *Proceedings of the OCEANS Conference & Exposition*, 2019, pp. 1–8.

[5] Q. Wang, W. Jiao, P. Wang, and Y. Zhang, "Digital twin for human-robot interactive welding and welder behavior analysis," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 2, pp. 334–343, 2021.

[6] I. A. Tsokalo, D. Kuss, I. Kharabet, F. H. P. Fitzek, and M. Reisslein, "Remote robot control with human-in-the-loop over long distances using digital twins," in *Proceedings of the IEEE Global Communications Conference*, 2019, pp. 1–6.

[7] D. Puljiz, E. Stöhr, K. S. Riesterer, B. Hein, and T. Kröger, "General hand guidance framework using microsoft hololens," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019, pp. 5185–5190.

[8] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2018, pp. 5628–5635.

[9] H. Liu, Y. Zhang, W. Si, X. Xie, Y. Zhu, and S.-C. Zhu, "Interactive robot knowledge patching using augmented reality," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2018, pp. 1947–1954.

[10] J. S. Dyrstad, E. Ruud Øye, A. Stahl, and J. Reidar Mathiassen, "Teaching a robot to grasp real fish by imitation learning from a human supervisor in virtual reality," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 7185–7192.

[11] https://github.com/robotic-vision-lab/Virtual-Reality-Robotic-Workspace.

[12] D. Mukherjee, K. Gupta, L. H. Chang, and H. Najjaran, "A survey of robot learning strategies for human-robot collaboration in industrial settings," *Robotics and Computer-Integrated Manufacturing*, vol. 73, p. 102231, 2022.

[13] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "Ros: An open-source robot operating system," in *Proceedings of the IEEE International Conference on Robotics and Automation Workshop on Open Source Software*, vol. 3, no. 3.2, 2009, p. 5.

[14] D. Coleman, I. Sucan, S. Chitta, and N. Correll, "Reducing the barrier to entry of complex robotic software: a moveit! case study," *arXiv preprint arXiv:1404.3785*, 2014.

[15] C. P. Quintero, S. Li, M. K. Pan, W. P. Chan, H. Machiel Van der Loos, and E. Croft, "Robot programming through augmented trajectories in augmented reality," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 1838–1844.

[16] L. Pérez, E. Diez, R. Usamentiaga, and D. F. García, "Industrial robot control and operator training using virtual reality interfaces," *Computers in Industry*, vol. 109, pp. 114–120, 2019.

[17] J. Lambrecht and J. Krüger, "Spatial programming for industrial robots based on gestures and augmented reality," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 466–472.

[18] T. Kot, P. Novák, and J. Bajak, "Using hololens to create a virtual operator station for mobile robots," in *Proceedings of the International Carpathian Control Conference*, 2018, pp. 422–427.

[19] S. Y. Gadre, E. Rosen, G. Chien, E. Phillips, S. Tellex, and G. Konidaris, "End-user robot programming using mixed reality," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2019, pp. 2707–2713.

[20] M. Ostanin, S. Mikhel, A. Evlampiev, V. Skvortsova, and A. Klimchik, "Human-robot interaction for robotic manipulator programming in mixed reality," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2020, pp. 2805–2811.

[21] F. Ghiringhelli, J. Guzzi, G. A. Di Caro, V. Caglioti, L. M. Gambardella, and A. Giusti, "Interactive augmented reality for understanding and analyzing multi-robot systems," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 1195–1201.

[22] I. Y.-H. Chen, B. MacDonald, and B. Wunsche, "Mixed reality simulation for mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2009, pp. 232–237.

[23] *Steam Valve Index Headset*, 2023. [Online]. Available: https://www.valvesoftware.com/en/index/headset

[24] *Unity 3D Game Engine*, 2023. [Online]. Available: https://unity.com/

[25] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, 2004, pp. 2149–2154.

[26] D. Krupke, F. Steinicke, P. Lubos, Y. Jonetzko, M. Görner, and J. Zhang, "Comparison of multimodal heading and pointing gestures for co-located mixed reality human-robot interaction," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 1–9.

[27] D. Bambušek, Z. Materna, M. Kapinus, V. Beran, and P. Smrž, "Combining interactive spatial augmented reality with head-mounted display for end-user collaborative robot programming," in *Proceedings of the IEEE International Conference on Robot and Human Interactive Communication*, 2019, pp. 1–8.

[28] *NVIDIA PhysX 4.0 Physics Engine SDK*, 2023. [Online]. Available: https://developer.nvidia.com/physx-sdk

[29] I. A. Şucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[30] M. Q. Tram, *UR-Robotiq Integrated Driver*, 2023. [Online]. Available: https://github.com/robotic-vision-lab/UR-Robotiq-Integrated-Driver

[31] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013.

[32] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

[33] A. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 15, 2002.