Integration of Real and Simulated ALOHA Robots Using Imitation Learning Techniques

Lukas Probst
Karlsruhe Institute of Technology
Karlsruhe, Germany
Email: lukas.probst@student.kit.edu

Jonathan Fricke
Karlsruhe Institute of Technology
Karlsruhe, Germany
Email: ucioq@student.kit.edu

***Abstract*—This paper explores the integration of real and simulated ALOHA robots using Imitation Learning (IL) techniques to teach robotic systems new tasks through demonstrations. A human expert controls the ALOHA robot via teleoperation in a simulated environment provided by NVIDIA Isaac Lab, which offers a robust and safe platform for training and testing. By recording demonstrations in simulation, we collect large datasets for training IL models. The project focuses on task learning through a series of manipulation tasks, including table sweeping, block picking, and block transferring. A novel approach, Movement Primitive Diffusion (MPD), is employed to train the robots in performing complex, fine-grained manipulations. The use of a combination of camera views, such as gripper-mounted and bird's-eye perspectives, enhances the monitoring and control of the robot's actions. Our findings demonstrate that simulated environments and IL can effectively accelerate the development of robotic systems capable of performing intricate tasks with high precision.**

I. Introduction

Enabling robots to learn new skills and tasks through demonstrations is a fundamental objective in the field of robot learning and human-robot interaction. Imitation Learning (IL) stands out as a powerful approach by observing and replicating expert demonstrations. In this context, a human expert controls a simulated version of the ALOHA [1] robot using the real custom teleoperation interface of the robot, within NVIDIA Isaac Lab [2]. It provides a robust simulated environment for training and testing. The expert can demonstrate solutions to a variety of tasks, which are then recorded and used to train the robot.

Simulated environments offer significant advantages for IL, particularly in high-stakes applications like robot-assisted surgery (RAS). They provide a safe and controlled setting where robots can practice and refine their skills without the risks associated with real-world errors. Moreover, simulated environments facilitate the collection of large datasets of expert demonstrations, which are crucial for training robust models. This data can be used to develop and fine-tune methods like Movement Primitive Diffusion (MPD), a novel approach designed specifically for IL in RAS, focusing on the delicate manipulation of deformable objects [3]. By leveraging these simulated environments, researchers can accelerate the development of advanced robotic systems that are capable of performing intricate tasks with high precision and reliability.

II. Related Work

A. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware

A Low-cost Open-source HArdware System for Bimanual Teleoperation (ALOHA) is the underlying system with off-the-shelf robots and 3D printed components [1]. Two follower robots mirror the teleoperated movement by the expert using two freely moveable joysticks. It is capable of precise and dynamic movements and gripping to accomplish everydays tasks.

B. Movement Primitive Diffusion

This method for imitation learning in robot-assisted surgery [3] combines the versatility of diffusion-based imitation learning with the high-quality motion generation capabilities of Probabilistic Dynamic Movement Primitives (ProDMPs) [4]. ProDMPs serve as a unified framework that merges the strengths of dynamics-based and probabilistic movement primitives, enabling the generation of smooth, adaptable trajectories while capturing the variability and higher-order statistics inherent in demonstrated movements.

C. Benchmark

The evaluation of imitation learning algorithms' ability to capture and reproduce diverse human behaviors is a crucial aspect of robotics research. The D3IL benchmark, presented in "Towards Diverse Behaviors: A Benchmark for Imitation Learning with Human Demonstrations" [5], addresses this need by providing a suite of tasks designed to assess an algorithm's capacity to learn from multi-modal data distributions. These tasks incorporate human demonstrations, involve multiple sub-tasks and object manipulation, and necessitate closed-loop sensory feedback, thus promoting behavioral diversity. In contrast, other benchmarks like RLBench and ManiSkill2, while offering a variety of tasks, lack diversity in their demonstrations due to their reliance on motion planning. Our project aligns with the spirit of D3IL by emphasizing the importance of learning from diverse human demonstrations to achieve more natural and adaptable robot behavior.

D. Augmented Reality Demonstrations

The utilization of Augmented Reality (AR) for robot learning through demonstration is an area of active research, as evidenced by the work of [6]. Their study focused on evaluating the effectiveness and intuitiveness of various AR-based interfaces for collecting demonstrations for a virtual robot. They explored five distinct interfaces: Hand Tracking, Virtual Kinesthetic Teaching, Gamepad, Motion Controller, and Kinesthetic Teaching, with the latter involving the manipulation of a physical robot to control its virtual counterpart. Their findings underscored the superior performance of Kinesthetic Teaching in terms of both objective metrics (success rate, task completeness, completion time) and subjective user experience. This research aligns with our project's focus on leveraging intuitive interfaces for effective demonstration collection, highlighting the potential of AR and physical interaction for enhancing the robot learning process.

E. Robot Learning for Manipulation

The field of robot learning for manipulation encompasses a wide array of research, tackling challenges such as defining state spaces, learning environmental transition models, and acquiring motor control policies, as highlighted in Kroemer et al.'s comprehensive review [7]. The review emphasizes the critical role of learning in enabling robots to interact effectively with their environments, particularly in unpredictable real-world scenarios. It underscores the importance of generalization in manipulation learning, advocating for a multi-task formulation that allows robots to adapt and apply learned skills across a diverse range of tasks. The review also delves into various technical challenges, including state space definition, transition model learning, and motor skill acquisition, providing a structured overview of the current state-of-the-art and highlighting promising avenues for future research.

III. Methods

A. Simulation Setup

1) Introduction to Simulation Environment: We utilize NVIDIA's Isaac Lab [2] for both physics simulation and as a platform to train policies for our robotics tasks. This environment offers a high-fidelity and flexible simulation space and realistic rendering, which is crucial for developing and testing complex robotic behaviors in a controlled and safe setting.

2) Robot Model Integration and Configuration: For our specific setup, we updated a model of the ViperX 300 S with a simplified robot description (MJCF) of the bimanual ALOHA 2 robot from Google DeepMind [8]. It is part of MuJoCo Menagerie, which offers a curated library of well-designed models that work well right out of the box [9]. Given the need to work with Universal Scene Description (USD) files in Isaac Lab, the MJCF Importer was employed to convert the MuJoCo representation into a USD format, allowing seamless integration into the simulation environment. Further adjustments to the robot configuration were done by converting the binary USD format to ASCII encoded, human-readable USDA format. The OpenUSD library by Pixar Animation Studios made it possible. This gave us the ability to precisely adjust the robot configuration to our needs.

Figure 1 shows the simulated view of the robot.

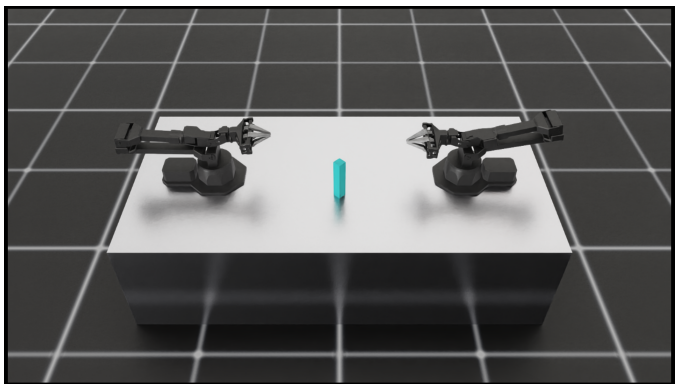

Fig. 1. Initial view of the simulated robot in NVIDIA's Isaac Lab.

3) Controller Implementation: The simulation is managed using a *ManagerBasedEnv*, which handles the overall environment operations. Custom controllers, specifically the *AlohaController* control the kinematics and dynamics of the ALOHA robot. The controller employs forward kinematics to execute actions based on the desired end-effector positions and gripper commands, ensuring precise control of the robotic system. We adjusted the Proportional–Integral–Derivative (PID) controller parameters to achieve a responsive and stable handling. With limited computing power the simulation physics can get unstable, therefore it is always a trade-off between responsiveness and stability when tuning the stiffness and damping factors.

B. Teleoperation and Interaction

1) Teleoperation Interface: To facilitate real-time interaction with the simulation, our teleoperation system is largely based on the *one_side_teleop.py* script from the ALOHA project. With this implementation the left and right puppet robots have to be initialized in order to activate the master. However, they are not activated or used in our setup and stay idle.

This system enables a human operator to seamlessly send commands to the robot while receiving immediate feedback from the simulation. The teleoperation interface plays a critical role in demonstrating the tasks that the robot needs to learn, thereby generating the essential data required for training models through imitation learning.

Figure 2 depicts our ALOHA setup, including the leader robots controlled by the user, the simulation displayed on a monitor, and the inactive follower robots.

To get the ALOHA master inputs into the simulation without having to merge both repositories, we used the *multiprocessing* Python package. By instantiating a client listener connection, joint angles of the master robot can be send to the simulation script in real-time over the local network.

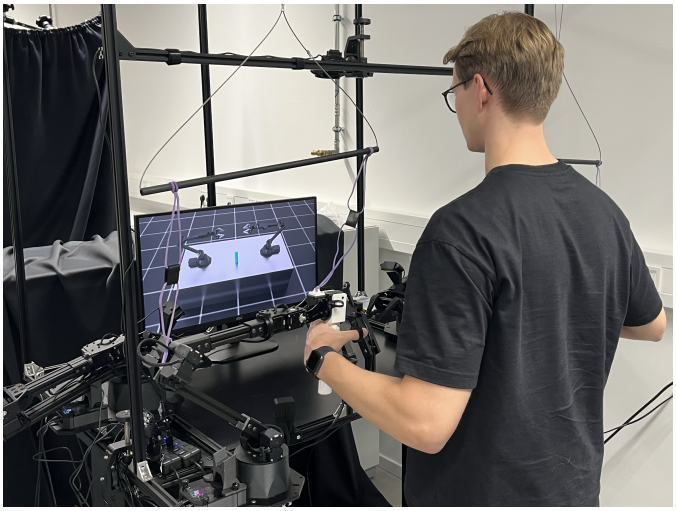

Fig. 2. The user teleoperates the simulated ALOHA robot which can be seen on an external monitor by using the two joysticks. Each joystick consists of a 3D-printed "handle and scissor" mechanism. The rubber band load balancing mechanism partially counteracts the gravity on the master side.

Achieving a low latency and realistic control scheme. And keeping the two repositories separate.

2) Recording and Data Collection: Within the teleoperation setup, we incorporated a *Recording* class that captures essential data such as joint positions, velocities, and gripper actions during the simulation. This data is first aggregated in memory and after stopping the recording, it is saved in NumPy *npz* file formats. Thereby enabling subsequent analysis and usage for training the robot models. The system is designed to allow users to start, stop and discard recordings, and reset the simulation environment interactively. Recordings are collected in every simulation iteration and only capped by the physics delta time which is $\frac{1}{60}$ seconds. This implementation has the advantage of collecting all the data that is generated regardless of computing power or delays in simulation loop.

C. Training with Movement Primitive Diffusion (MPD)

1) Introduction to MPD: Movement Primitive Diffusion (MPD) is a novel approach used for imitation learning in robot-assisted surgery (RAS) [3]. It focuses on the gentle manipulation of deformable objects, a crucial requirement in surgical tasks. The MPD method is trained using the data collected from the teleoperation interface, where a human expert demonstrates the tasks.

2) Connecting Simulation to MPD Training: The captured training data from simulation demonstrations can be directly imported as NumPy *npz* files (with the correct folder structure) into the MPD training procedure. For the ALOHA tasks we used the obstacle avoidance configuration as a template.

D. Execution and Model Deployment

1) Simulation Execution with Trained Models: Once the MPD model is trained, it is deployed within the same simulation environment to validate its performance. We used the same multiprocessing implementation as in the *teleop_sim.py* scripts to connect the simulation repository with the MPD repository. By keeping them separate, we obtain a more elegant and flexible integration. The simulation allows for thorough testing of the model's capabilities before it is used in real-world applications. The model only outputs a short trajectory of 12 actions before it gets 3 new observations. Therefore enabling interactive behaviour based on the simulation state.

2) Final Deployment and Applications: Following successful testing in the simulation, the trained model is ready for deployment in actual robot-assisted tasks. The integration of ProMPD within the simulation ensures that the robot is adequately prepared to handle delicate operations, with a focus on safety and precision. Furthermore, while being cost-effective, flexible and rapid to iterate, the model is able to achieve this without compromising the aforementioned safety and precision.

E. Code Availability

To ensure transparency and facilitate further research, the codebase, including the detailed implementation of the teleoperation interface as well as the extended movement-primitive-diffusion repository, are made publicly available on GitHub. At this moment, the *aloha_platform_simulation* repository remains private as part of ALRhub's intellectual property. In the future, it may be possible to get public access to this repository too.

- Teleoperation interface with a connection to the simulation
- Extended movement-primitive-diffusion repository
- ALOHA simulation (currently private).

IV. Experiments

The experiments are divided into recording, training and execution stages. Each stage is separate and only connected with each other when it is necessary to exchange data. The recording stage uses the teleoperation actions and executes them in simulation, thereby creating observations. These observations are used in the training stage to create the respective model. This model is then used in the final execution stage to send actions and receive resulting observations from the simulation.

Figure 3 illustrates the overview of the pipeline.

A. Simulation

NVIDIA Isaac Lab [2] which is based on NVIDIA Isaac Sim, is specialized for creating robot environments. We use it to perform physics simulation and as a platform to execute policies for our robot tasks.

B. Rendering

To get a more realistic view of the scene we adjusted the materials, lighting and perspectives.

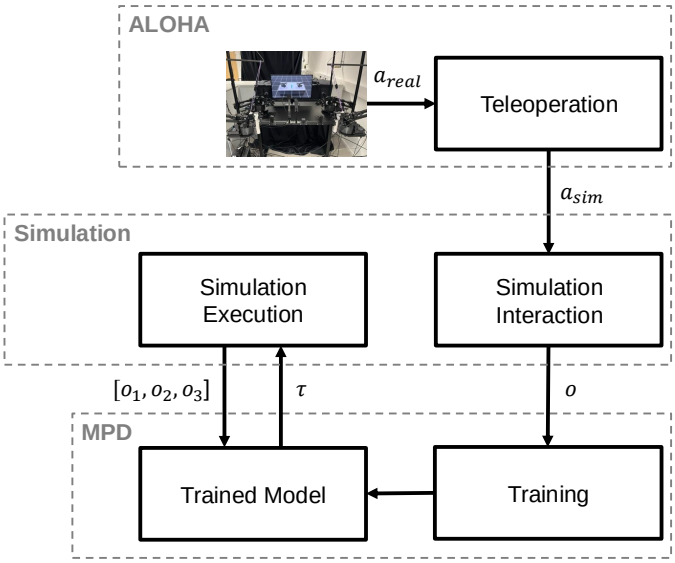

Fig. 3. **Overview of the Pipeline:** A human expert controls the robot using teleoperation, generating real-world actions a_{real} that are applied to the simulation as a_{sim}. These actions are simulated in NVIDIA Isaac Lab and provide an observation o. By harnessing the MPD method, the model is trained with the collected data. The final simulation execution is run by trajectories τ and the last three observations $[o_1, o_2, o_3]$ are used as feedback.

1) Materials: We changed the material of the robot to be closer to the real robot. The applied material is the *OrbitPBR*, which is Physically Based Rendering. Therefore it does not only change the color, but also the surface properties like metallic and roughness. As a result, we get more realistic rendering.

2) Lighting: We also improved the background environment texture, by creating a equirectangular image of the real life environment and importing it into simulation. The background acts as a spherical light source, making additional lights unnecessary.

3) Perspectives: The main camera view was chosen to be as close as possible to the view of a standing user in front of the robot-attached table (see Figure 1). This view provides an intuitive perspective that closely mirrors how a human operator would observe the interaction, making it easier to monitor the task progress and the robot's actions.

We virtually placed cameras onto the grippers, similar to those on the real robot, to offer a detailed, close-up view of the manipulative actions. This allows for precise tracking of object handling and can be critical for evaluating the fine movements of the grippers (see Figure 4). However, gripper-mounted cameras may have limited field of view, making it difficult to assess spatial awareness, such as where objects or other obstacles are located relative to both arms.

In addition, there is a bird's-eye view, which offers a comprehensive overhead perspective (see Figure 5).

This combination of views enhances both the monitoring and control of the robot's actions. Unfortunately, from our understanding and this point in time it is only possible to view more then two viewports (cameras) at the same time in NVIDIA Isaac Lab.

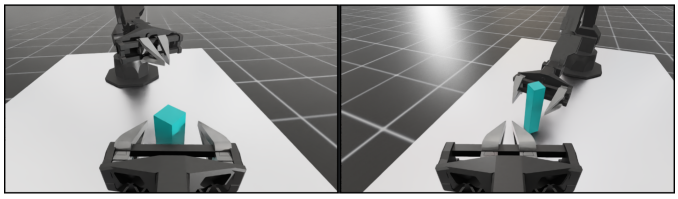

Fig. 4. Each gripper has an attached camera. ***Left:*** The gripper camera view of the left robot arm. ***Right:*** The gripper camera view of the right robot arm.

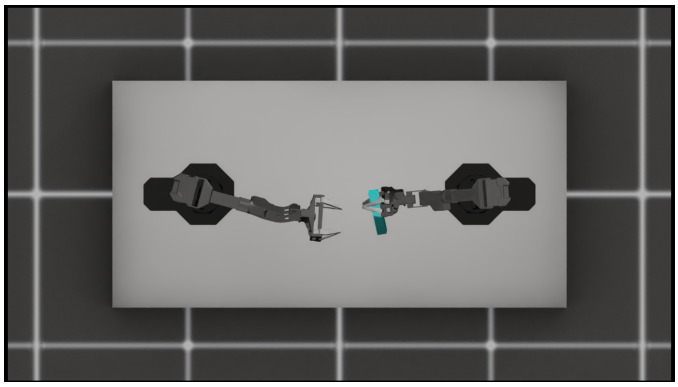

Fig. 5. This top-down view shows the two robotic arms interacting with the block on the table. The right arm is grasping the block, while the left arm is in position, preparing for the next action.

C. Tasks

1) Table Sweep: The goal of this task was to create a simple baseline which can be learned quickly to test and evaluate our pipeline. The task is to sweep a block off the table, starting from the initial position. Having only one mode and a short trajectory makes it suitable to evaluate the functionality of the system. Because this movement does not require the gripper and the interaction with the block is trivial, we used a reduced dataset in which the gripper and block-pose keys are removed from the data.

2) Pickup Block: This task is meant to be a significant step up in difficulty from the trivial Table Sweep task. It not only includes the joint angles of the robot but also the gripper, the position, and orientation of the block in 3D-space. To complete the task the robot is supposed to grip the block placed on the table and pick it up by moving it into the air.

3) Block Transfer: The final task of handing over the block is the most difficult one. Besides picking up the block, it requires bimanual manipulation and interaction between both robot arms. Due to time constraints we could not train a functional model for this task.

These three tasks, including their initialization and subtasks, are illustrated in Figure 6, Figure 7, and Figure 8.

D. Recording

Recordings are exclusively done with NVIDIA Isaac Lab and the *teleop_sim.py* script from our simulation repository.

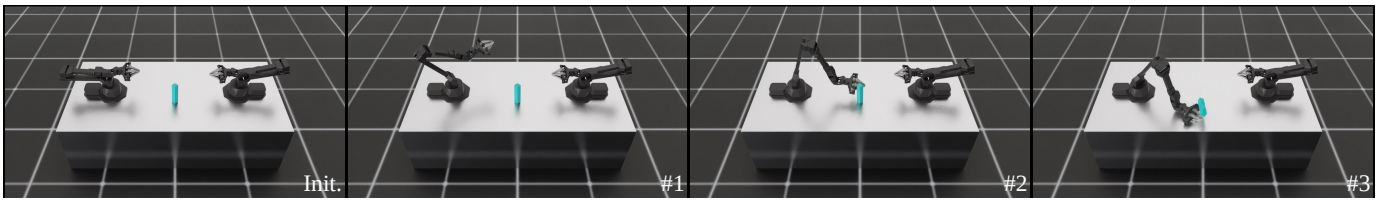

Fig. 6. ***Table Sweep:*** The left arm first moves to the middle of the table (frame #1) followed by moving to the block (frame #2). Then it pushes the block off the table (frame #3).

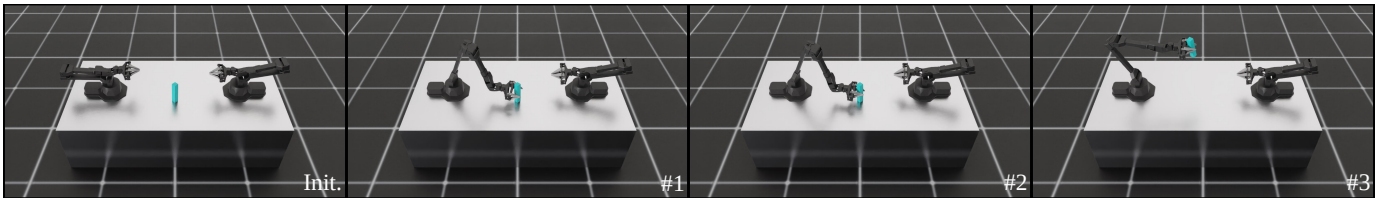

Fig. 7. ***Pickup Block:*** The left arms positions itself so that the block is within the grip area of the gripper (frame #1) and closes it (frame #2). Next, it lifts the block off the table (frame #3).

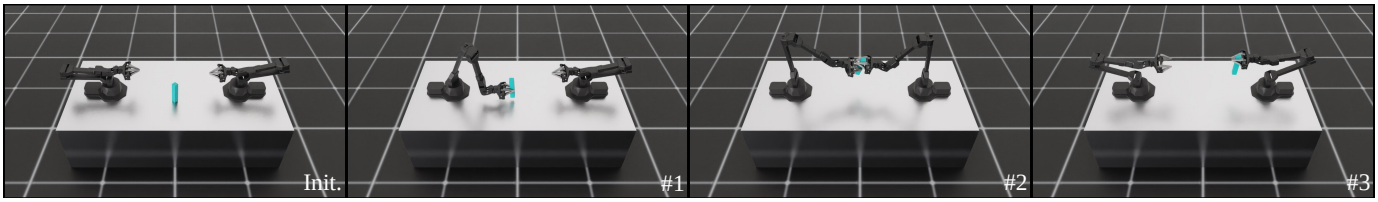

Fig. 8. ***Block Transfer:*** The left arm picks up the block (frame #1) and hands it over to the right arm (frame #2). Then the left arm releases the block and the right arm takes over (frame #3).

This script is based on an existing keyboard teleoperation script developed by the Autonomous Learning Robots (ALR) Lab at the Institute for Anthropomatics and Robotics at KIT. In our script we establish a client listener connection between the *teleop_sim.py* scripts from the simulation and the ALOHA robot. The ALOHA script constantly sends the current position and the simulation processes them as actions. Because of the asynchronous nature of this connection the simulation always empties the build in the network queue and only executes the most recent action. Therefore the simulation will not lag behind but might stutter when experiencing low physics frame-rate.

E. Playback

To replay the recordings we wrote a playback script which takes the recorded actions and simply replays them in a loop. The actions are executed on the simulated robot as they were recorded before This allows a fast analysis of single samples.

F. Training

For training procedure and model creation we used the movement-primitive-diffusion public repository. Our model is based on the Probabilistic Dynamic Movement Primitive Transformer (ProDMP Transformer), which was used for comparable manipulation tasks in the associated paper [3]. Our model uses the state of the simulation and actions as keys, which are noise free and easily obtainable from simulation. We trained locally using an NVIDIA GeForce RTX 2060 Super graphic card and with an NVIDIA Tesla V100 on the bwUniCluster 2.0.

G. Execution

In the execution stage initially a client listener connection is established between the simulation and the model script in the *movement-primitive-diffusion* repository. The simulation starts sending observations which get processed by the model and sends back a sequence of actions. Those actions are executed, thereby creating new observations which restarts the interaction cycle.

H. Data

1) Raw Data: Demonstrations consist of multiple trajectories comprising action, agent-position, agent-velocity, box-pose and delta-time keys. The delta time can be used to give more information about the speed of the recording, it records variable temporal data. In the course of our experiments, we choose to disregard the delta-time data in order to streamline the learning process, given that the frame rate was suboptimal but stable. Consequently, the addition of this data yielded no discernible insights in our experiments. The box-pose was also ignored for the Table Sweep task to further reduce the base task complexity.

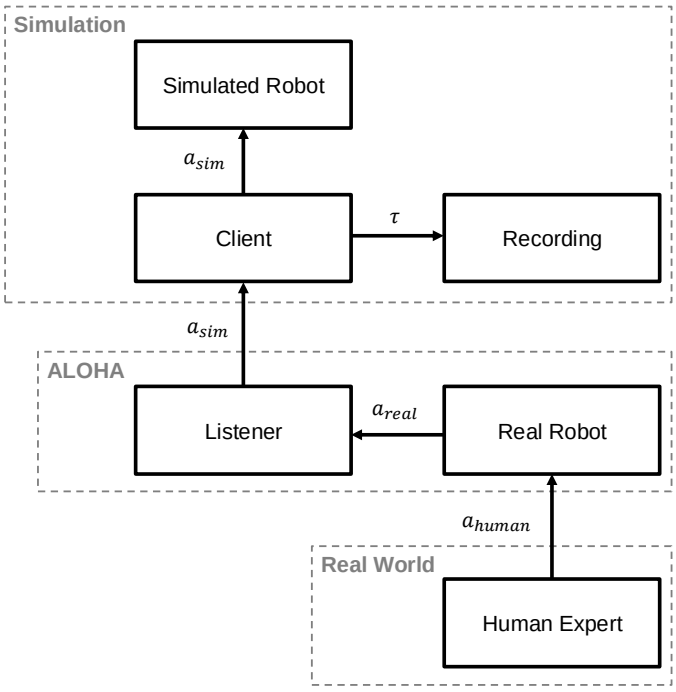

Fig. 9. **Recording Process:** A human expert provides actions a_{human} to the real robot. These actions are received by the real robot, which generates real actions a_{real}. The Listener in the ALOHA repository receives these actions sends them as a_{sim} to the Simulation through a Client. The simulated robot's actions can be recorded by saving the trajectories τ. This closed-loop system enables both real-world and simulated environments to interact.

2) Data Augmentation: Moreover, to ensure effective model training, the quantity of data must be adequate. By interpolating between data points, a single existing trajectory can be transformed into multiple new trajectories. This approach is particularly advantageous in our context, given the constraints on computational resources in our local setup. The sampling rate achieved was approximately 20 frames per second, which was preferable for interpolation purposes when compared to a hypothetical scenario with an ideal number of samples. The augmentation script generates a k-interpolated copy of each of the n original trajectories, with successively increasing interpolation steps $(1, 2, 3, \cdots, k)$.

$$\text{num\_samples} = \sum_{i=1}^{p} \left(\sum_{j=1}^{n} (|\tau_j| - 1) \cdot i \right) \tag{1}$$

This approach not only increases the number of trajectories but also extends the length of each trajectory, thereby creating trajectories with varying velocities. However, it is essential to consider the trade-off between trajectory quality and quantity, given our initial implementation with linear interpolation. To address this limitation, we introduced cubic interpolation, which does not suffer from the issue of the observation used by the model representing a trivial straight line. For our dataset, we have opted to create four interpolation steps, resulting in approximately 120 trajectories.

V. Results

A. Training

The performance of the model was evaluated using the *end_point_deviation* as the primary performance metric, with a goal of minimizing this value. After 1781 epochs of training, the best performance value achieved was **0.0138** (see Figure 10). The training and validation loss values were also monitored throughout the process, with the *mean training loss* reaching **1.085e-05** and the *mean validation loss* being **5.252e-05**.

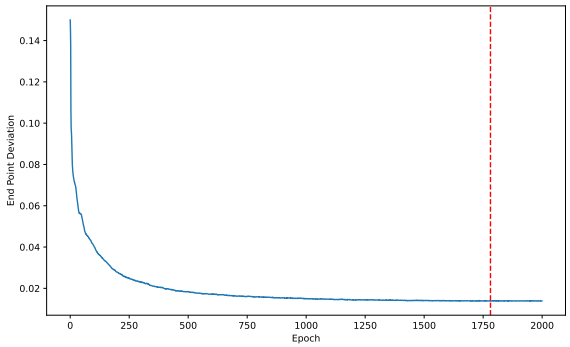

Fig. 10. ProDMP end point deviation while training. The red line indicates the epoch with the best end point deviation.

The following configuration was used during the experiment:

- The keys used in the model were `action`, `agent_pos`, and `agent_vel`.
- The dataset consisted of **144 trajectories**, with a mean trajectory length of **487**.
- The time observed (`t_obs`) was **3**, with **12** time steps for both prediction (`t_pred`) and action (`t_act`).
- The model was trained on a **90% training split**, with the remaining 10% used for validation.
- The training was performed on a `CUDA` device, allowing for efficient GPU-accelerated computation.

The `early stopping` mechanism was disabled, and the model was trained for **2000 epochs** without warmup or patience-based stopping. The training created the best performing model in the **1781. epoch**.

The learning process showed rapid decrease in both training and validation losses during the early stages of training, followed by a stabilization (see Figure 11 and Figure 12).

B. Execution

To evaluate the policy, it was executed in the simulation environment. 30 iterations where executed for one run, in each iteration 12 actions were generated and executed resulting in 360 actions for one test run. An evaluation run was counted as successful execution when the block was moved off the table. However, after many runs the success rate was always poor, showing no skilled behaviour in simulation.

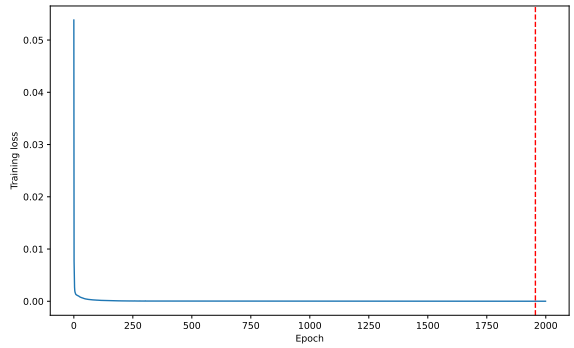

Fig. 11. ProDMP training loss. The red line indicates the epoch with minimal loss.

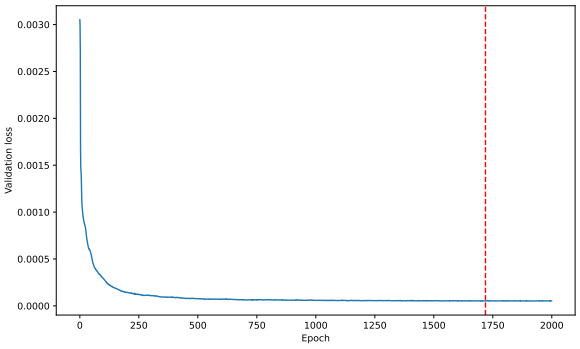

Fig. 12. ProDMP validation loss. The red line indicates the epoch with minimal loss.

VI. Evaluation

A. Training Dynamics

The almost instantaneous drop in loss suggests that the model learned efficiently and converged to a solution. This is probably due to the simple trajectory which is learned. Despite the absence of early stopping, the model did not overfit, as evidenced by the close proximity of training and validation losses (see Figure 11 and Figure 12).

B. Model Performance

The primary metric of *end_point_deviation* was chosen to assess the accuracy of the predicted trajectories. The best performance value of **0.0138** indicates that the model was able to achieve highly accurate predictions, with minimal deviation from the expected end points of the trajectories. This result demonstrates the effectiveness of the ProDMP model in learning and replicating complex trajectories within the given simulation environment.

C. Execution

The model showed no applicable skill in the simulation, it behaved almost randomly and did not replicate the movement. This indicates a problem in either the generation of observations, the training process or the training data.

VII. Conclusion

This project demonstrated the utility of NVIDIA Isaac Lab in facilitating the integration of simulation and real-life robotics. This incorporates the fundamental structure for a comprehensive robot learning pipeline, encompassing data generation, augmentation, policy learning and policy execution. Due to time constraints, this project only includes the simple table sweep task and the obtained model execution results are suboptimal. This may be attributed to a number of factors. The process of normalization and denormalization must be completed for both observations and actions, which could introduce an element of error when implemented wrongly. Additionally, as illustrated in Figure 12, the validation loss declines rapidly, suggesting that the data may be insufficient. Furthermore, the data may lack sufficient diversity, allowing the model to transition into regions that are outside of the distribution range. This ultimately results in an unstable behavioural pattern. In order for the ProDMP model to function correctly, it is necessary to provide it with additional input beyond that which is required for the agent position. In particular, the initial start position and initial start velocity are required for the dynamic movement primitive system. This could be a potential source of error. We attempted to construct a diffusion model to circumvent this issue. However, this proved ineffective, suggesting that the problem may lie elsewhere, either with the model itself or the training process. No further investigation of the diffusion model was possible within the time constraints.

VIII. Future work

The objective of this project was to establish a foundation for the generation of a policy in ALOHA within the context of NVIDIA Isaac Lab. The straightforward tasks illustrated in this paper can be expanded to assess more intricate behaviors with more sophisticated task configurations. NVIDIA Isaac Lab allows for the implementation of stable physics interactions based on rigid bodies. Thus providing a foundation for the development of more complex behaviors within the system. One avenue for further investigation is the creation of tasks based on furniture benches, e.g. a drawer model (see Figure 13 and Figure 14). Moreover, the integration of a virtual reality system could facilitate an enhanced user experience when recording demonstrations. Such an approach would permit more exact control due to the enhanced depth perception. The ability to replay actions using the $play\_recording.py$ script can be leveraged to enhance the quality of recorded data. For example, modifying the speed of action replay will result in a notable discrepancy between the agent position and the original sample. Modifying the environmental lighting allows image-based models to obtain an infinite amount of realistic augmentation data. Incorporation of supplementary camera perspectives can facilitate an even more comprehensive data distribution.

The use of the multiprocessing library and instantiating a client and listener enables the possibility to add remote controlled teleoperation. Where the user and robot are separated from the simulation and are only connected by the internet.

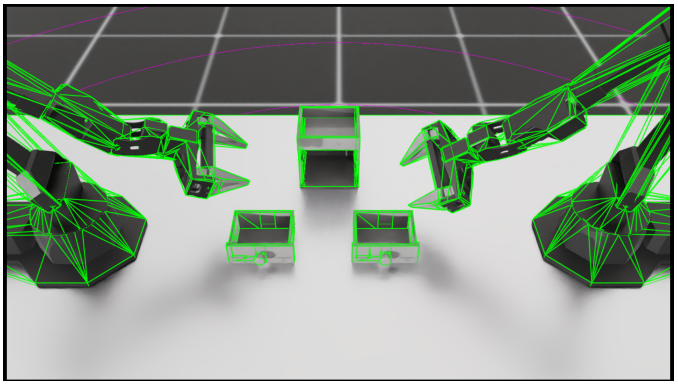

Fig. 13. Isaac Lab rendering of the ALOHA Robot with furniture bench drawer models. The green outlines show the generated collision meshes.

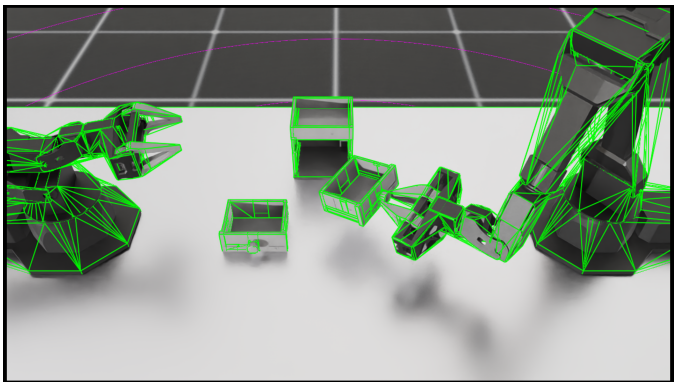

Fig. 14. Isaac Lab rendering of the ALOHA Robot stable interacting with small knob of drawer.

ACKNOWLEDGMENT

The authors acknowledge support by the state of Baden-Württemberg through bwHPC.

REFERENCES

[1] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," 2023. [Online]. Available: https://arxiv.org/abs/2304.13705

[2] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg, "Orbit: A unified simulation framework for interactive robot learning environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3740–3747, 2023.

[3] P. M. Scheikl, N. Schreiber, C. Haas, N. Freymuth, G. Neumann, R. Lioutikov, and F. Mathis-Ullrich, "Movement primitive diffusion: Learning gentle robotic manipulation of deformable objects," *IEEE Robotics and Automation Letters*, vol. 9, no. 6, p. 5338–5345, Jun. 2024. [Online]. Available: http://dx.doi.org/10.1109/LRA.2024.3382529

[4] G. Li, Z. Jin, M. Volpp, F. Otto, R. Lioutikov, and G. Neumann, "Prodmps: A unified perspective on dynamic and probabilistic movement primitives," 2022. [Online]. Available: https://arxiv.org/abs/2210.01531

[5] X. Jia, D. Blessing, X. Jiang, M. Reuss, A. Donat, R. Lioutikov, and G. Neumann, "Towards diverse behaviors: A benchmark for imitation learning with human demonstrations," 2024. [Online]. Available: https://arxiv.org/abs/2402.14606

[6] X. Jiang, P. Mattes, X. Jia, N. Schreiber, G. Neumann, and R. Lioutikov, "A comprehensive user study on augmented reality-based data collection interfaces for robot learning," in *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, ser. HRI '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 333–342. [Online]. Available: https://doi.org/10.1145/3610977.3634995

[7] O. Kroemer, S. Niekum, and G. Konidaris, "A review of robot learning for manipulation: Challenges, representations, and algorithms," 2020. [Online]. Available: https://arxiv.org/abs/1907.03146

[8] A. . Team, "Aloha 2: An enhanced low-cost hardware for bimanual teleoperation," 2024. [Online]. Available: https://aloha-2.github.io/

[9] K. Zakka, Y. Tassa, and MuJoCo Menagerie Contributors, "MuJoCo Menagerie: A collection of high-quality simulation models for MuJoCo," 2022. [Online]. Available: http://github.com/google-deepmind/mujoco_menagerie