Humanoid Manipulation with Vision-Language-Action Models

ECE 398 Final Report

Aidan Andrews
University of Illinois at Urbana-Champaign
aidansa2@illinois.edu

December 2025

Abstract

This report presents the development of a simulation-to-reality pipeline for humanoid robot manipulation using vision-language-action models (VLAMs). Working with the Unitree G1 humanoid and NVIDIA's GR00T N1.5 foundation model, I investigated fine-tuning strategies for enabling language-conditioned manipulation from limited demonstration data. The key finding is that parameter-efficient fine-tuning via LoRA dramatically outperforms full fine-tuning on small datasets—achieving successful generalization with only 14 teleoperation trajectories while full fine-tuning led to catastrophic overfitting despite 100,000 training steps. Diagnostic experiments on an SO-101 robotic arm validated the pipeline and confirmed data quality as the primary factor affecting model performance. The project establishes a complete infrastructure including Meta Quest 3 VR teleoperation, ROS2 integration, and Isaac Sim environments that enables continued research toward autonomous humanoid operation in laboratory settings.

1 Introduction

This project began with an ambitious goal: deploying vision-language-action models on the Unitree G1 humanoid to automate chemical laboratory tasks—specifically, an autonomous pick-and-pour sequence for elephant toothpaste. The motivation was a belief that embodied AI could extend machine intelligence beyond text-based automation into physical domains where real scientific discovery happens.

What I assumed would be a straightforward deployment revealed that current VLAs lack the out-of-box generalization I expected. Internet-sourced training data failed on my setup, teleoperation infrastructure for humanoids did not exist, and every layer of the pipeline—from simulation to data collection to fine-tuning—required building from scratch.

Through this struggle, I came to understand that physical intelligence may be more consequential than text-based AI. Just as scientists move from reading papers to working in labs where knowledge emerges through trial and error, enabling machines to reason and learn through embodied experience could push the frontier of all machine intelligence in ways that language models alone cannot.

The core research questions that emerged were:

1. How should VLAMs be fine-tuned when demonstration data is scarce?

2. What data collection methods are most effective for humanoid manipulation?

3. How can we diagnose and resolve performance issues in complex simulation pipelines?

This semester, I delivered a complete VLA deployment pipeline—teleoperation via Meta Quest, GR00T fine-tuning with LoRA on minimal data, and successful manipulation in simulation—that works across both the G1 humanoid and SO-101 arm, establishing the foundation for a hierarchical System-1/System-2 architecture that future work will build upon.

2 Project Timeline

The project progressed through seven distinct phases over the Fall 2025 semester. Each phase built on previous work while addressing emerging challenges.

2.1 Phase 1: Infrastructure and Initial Research (Sept 15–Oct 3)

I set up NVIDIA Isaac Sim and Isaac Lab with the G1 humanoid model and trained a basic locomotion policy to enable walking. Research during this phase revealed a critical finding: no single model currently handles both locomotion and dexterous manipulation well for humanoid robots. This led to the decision to develop a multi-component system consisting of a high-level planner ("System 2"), a locomotion policy, and GR00T N1.5 for manipulation.

I loaded multiple G1 embodiment configurations into Isaac Sim (Dex3 vs. Inspire hands) and settled on GR00T N1.5 as the primary VLAM for manipulation. Initial fine-tuning on G1 data with Dex3 hands was conducted to understand how GR00T works.

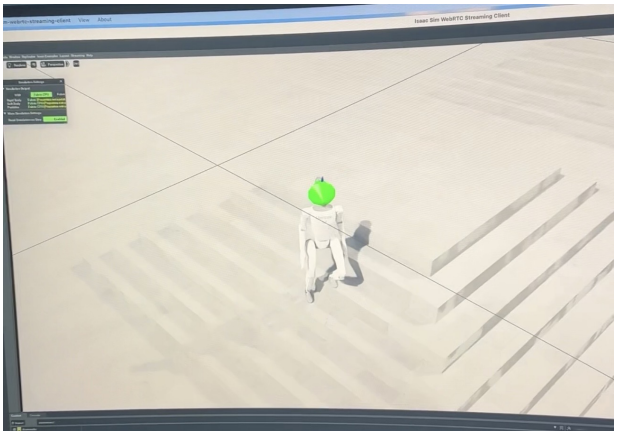

Figure 1: G1 humanoid executing trained locomotion policy in Isaac Sim.

2.2 Phase 2: GR00T-Isaac Sim Integration (Oct 6–17)

This phase marked a significant technical milestone: establishing a streaming connection between GR00T N1.5 and Isaac Sim for real-time control. I constructed a task environment with the G1 standing in front of a table with a cylinder to manipulate, using assets from Unitree's official Isaac Lab repository.

A critical issue emerged immediately: the G1 would fall over while executing GR00T's manipulation commands because GR00T only controls the upper body (arms and hands). Without lower-body stabilization, the robot's center of mass shifted beyond its support polygon. I addressed this by significantly increasing joint stiffness and damping parameters for the lower body and pelvis, and zeroing commanded velocities for leg joints. I also repositioned the robot closer to the table to ensure manipulation targets were within the workspace.

2.3 Phase 3: Multi-GPU Configuration (Oct 13–24)

I installed an RTX 5090 GPU alongside my existing RTX 4090 to accelerate training and inference. This introduced substantial integration challenges—Isaac Lab had difficulty reconciling with the multi-GPU setup, requiring modifications to source code and custom CUDA kernel compilation. I built a custom Flash Attention wheel and adjusted PyTorch configurations. Isaac Lab now ran on the 4090 due to PCIe bandwidth limitations (Gen 4 x4 on the 5090 slot).

During testing, I discovered a critical problem: the retrained GR00T model was producing identical actions regardless of the language prompt. The robot would perform arbitrary motions rather than purposeful manipulation. Despite thorough code review, I could not identify the cause. This motivated the diagnostic phase.

2.4 Phase 4: SO-101 Diagnostic Project (Oct 20–Nov 7)

To isolate whether issues originated from GR00T itself or my G1 integration, I acquired an SO-ARM101 leader-follower robotic arm. The strategy was clear: collect real-world teleoperation data on simpler hardware, fine-tune GR00T, and deploy directly to the physical robot. If GR00T worked on the SO-101, the problem was my G1 pipeline or data quality; if it failed similarly, the issue was with GR00T's training procedure.

I developed ROS2 bridge scripts for teleoperation data collection:

- Real-world leader arm → ROS2 joint position publisher → Isaac Sim subscriber → articulation controller commands to virtual follower

One of my follower arm motors failed unexpectedly, but this led to developing a leader-only ROS2 bridge that proved valuable for continued development while waiting for the replacement.

2.5 Phase 5: GR00T Validation on SO-101 (Nov 10–21)

After replacing the motor and fully fine-tuning GR00T N1.5 on SO-101 data, I built an inference script that streams webcam frames to the GR00T

server and executes actions in real-time on the physical robot.

Initial deployment revealed a critical hardware issue: the robot was not executing GR00T commands properly despite the software working correctly. Root cause analysis identified that Feetech STS3215 servos require `Goal_Time=0` to be set for responsive movement in position mode. Without this configuration, motors were essentially frozen despite receiving position commands. I added this fix to the LeRobot SO-100/SO-101 robot classes.

With the fix applied, the robot became fully functional for policy evaluation—I ran 150+ inference steps without crashes and all 6 joints tracked policy commands correctly. This validated that GR00T works correctly when provided with compatible data, confirming that my G1 issues stemmed from data quality rather than the model architecture.

Figure 2: GR00T deployed on SO-101.

2.6 Phase 6: VideoMimic and Data Collection Research (Nov 24–Dec 8)

After validating GR00T on the SO-101, a key realization emerged: current VLAs lack complex generalization. If the setup is not nearly identical between training and testing, or if the task differs, the model will not generalize. The extent of VLA capabilities is limited to small perturbations in the task or scene.

My original approach—training GR00T on precollected G1 data then deploying on my G1 in simulation with a completely different scene and task space—was fundamentally flawed. I needed to collect teleoperation data on the exact setup I would use for inference.

Since I did not have a VR headset at the time and keyboard/controller usage would result in jerky, uncoordinated motions, I implemented VideoMimic—a pipeline to convert webcam RGB video into G1 trajectories. I set up:

- **SAM2:** Human segmentation and video tracking

Figure 3: SAM2 output.

- **ViTPose:** 2D whole-body pose estimation (133 keypoints)

Figure 4: ViTPose output.

- **VIMO:** Temporal 3D human mesh reconstruction (SMPL parameters)
- **Grounding DINO:** Text-prompted object detection

The pipeline works and produces outputs for every frame of video, but converting these outputs to actual G1 joint positions proved complex. Retargeting

human poses to robot configurations requires solving inverse kinematics with different joint limits, link lengths, and degrees of freedom.

2.7 Phase 7: Meta Quest Teleoperation and Final Training (December)

Rather than completing the VideoMimic retargeting (which required substantial additional work), I acquired a Meta Quest 3 VR headset and integrated Unitree's XR_teleoperate package. This provided direct VR control of the G1 in Isaac Sim with hand and arm tracking mapped to robot joint positions.

I collected 14 teleoperation episodes (15,791 frames) for a pick-and-place task: moving a red block to a yellow target zone. Using this data, I conducted the central experiment of the project: comparing full fine-tuning versus LoRA fine-tuning on identical data.

Figure 5: Meta Quest 3 VR teleoperation setup for G1 data collection.

3 Team Roles

This was an individual research project conducted in KIMLAB under the supervision of Prof. Joohyung Kim, with Sankalp Yamsani as graduate mentor. As the sole researcher, I was responsible for all aspects:

- **Infrastructure:** Setting up simulation environments, multi-GPU configuration, and compute resource management on my personal AI server
- **Software Development:** Implementing ROS2 bridges, teleoperation systems, VideoMimic adaptation, and GR00T integration
- **Data Collection:** Designing tasks, collecting demonstrations via physical teleoperation and VR, and curating datasets
- **Experiments:** Training models, running ablations, and analyzing results
- **Hardware:** Configuring SO-101 arm, debugging servo issues, and integrating Meta Quest

All code was run on my personal AI server (RTX 5090 + RTX 4090). Building this server was not part of this project but significantly contributed to my ability to pursue this research.

4 Methods

4.1 Hardware Configuration

The primary robot platform is the **Unitree G1** 26-DOF humanoid robot with Inspire DFX hands. The diagnostic platform is the **SO-ARM101** 6-DOF leader-follower arm system with Feetech STS3215 servos (joint names: shoulder_pan, shoulder_lift, elbow_flex, wrist_flex, wrist_roll, gripper). Data collection uses a **Meta Quest 3** VR headset for hand and arm tracking.

The compute infrastructure consists of dual GPUs: an RTX 5090 and RTX 4090. Isaac Lab runs on the 4090 due to PCIe bandwidth limitations, while training utilizes the 5090.

4.2 Software Architecture

The software stack integrates:

- **NVIDIA Isaac Sim/Lab:** Physics simulation and robot learning framework
- **GR00T N1.5:** Diffusion transformer outputting target joint positions for arms and hands
- **ROS2:** Inter-process communication and sim-to-real bridge
- **LeRobot:** Teleoperation framework for SO-101
- **XR_teleoperate:** Unitree's VR teleoperation package

4.3 Training Methodology

GR00T N1.5 uses a diffusion transformer architecture that predicts action sequences conditioned on visual observations and language instructions. I compared two fine-tuning strategies:

Full Fine-Tuning: All model parameters are updated during training. This approach has high capacity but risks overfitting on small datasets.

LoRA Fine-Tuning: Low-Rank Adaptation [4] freezes pretrained weights and adds small trainable

rank decomposition matrices. For rank r, a weight matrix $W \in \mathbb{R}^{d \times k}$ is modified as:

$$W' = W + BA \tag{1}$$

where $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$ with $r \ll \min(d, k)$. This reduces trainable parameters from dk to $r(d+k)$.

4.4 Challenges Encountered

Multi-GPU Integration: The 5090 + 4090 configuration required custom CUDA kernel compilation, Flash Attention wheel building, and PyTorch configuration adjustments.

Servo Configuration: Feetech STS3215 servos appeared unresponsive until discovering the `Goal_Time=0` requirement for position mode—a critical fix that was not documented.

Data Compatibility: Pre-collected internet data with different normalizations, action spaces, and camera configurations failed completely. Data must match the exact inference setup.

VLA Generalization Limits: Current VLAs cannot generalize across substantially different scenes or task spaces—they handle only small perturbations from training distribution.

5 Results

5.1 SO-101 Diagnostic Results

Fine-tuning GR00T on 20 physical teleoperation trajectories from the SO-101 and deploying to hardware confirmed that the model learns effectively from compatible data. After fixing the servo configuration issue, the robot successfully executed 150+ inference steps, validating the pipeline and ruling out fundamental issues with GR00T's architecture.

5.2 Full Fine-Tune Experiment (Failed)

Training configuration:

- Training steps: 100,000
- Epochs: ∼50
- Training time: 10 hours
- Batch size: 8

The loss dropped 250× in early training (steps 0–850), then flatlined near zero with no improvement from step 850 to 100,000. At inference, the model produced identical trajectories every run regardless of language prompt or object position—a clear indicator of catastrophic overfitting. The model had memorized the 14 training trajectories rather than learning generalizable manipulation skills.

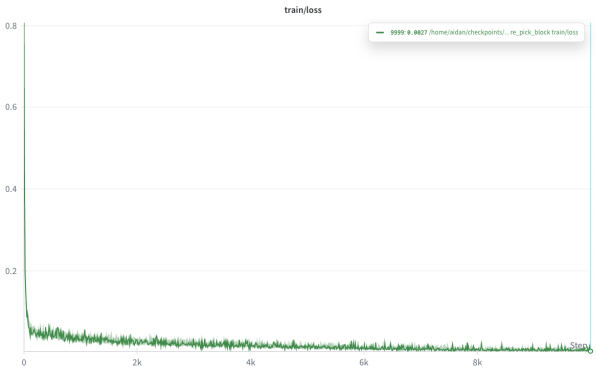

Figure 6: Training loss for full fine-tuning. Loss dropped 250× in early training but flatlined near zero after step ∼850, indicating severe overfitting.

5.3 LoRA Fine-Tune Experiment (Successful)

Training configuration:

- Training steps: 8,000–10,000
- Batch size: 8
- LoRA rank: 16
- LoRA alpha: 32
- Warm-up ratio: 0.1

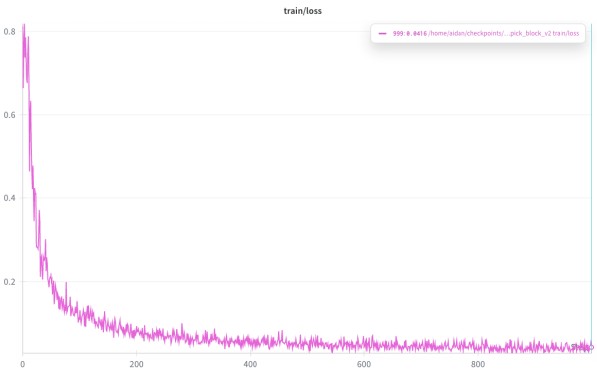

Figure 7: Training loss for LoRA fine-tuning showing proper decay without premature convergence.

The LoRA-trained model exhibited qualitatively different behavior:

- Distinct trajectories for each inference run
- Appropriate responses to different initial object positions
- Correct behavior changes based on language prompts
- Successful task completion (red block to yellow target zone)

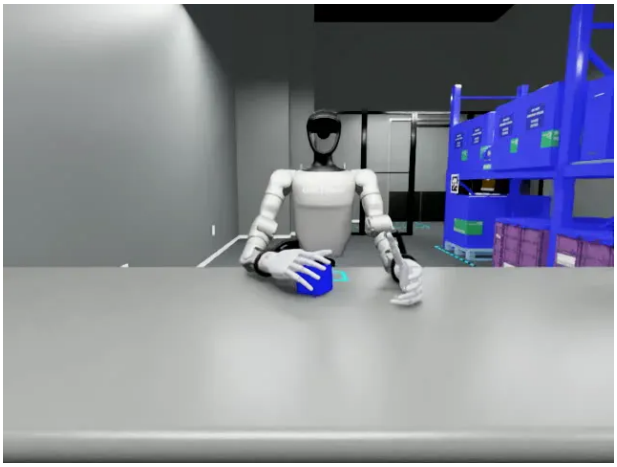

Figure 8: G1 successfully moving block to target zone using LoRA fine-tuned model. Front view.

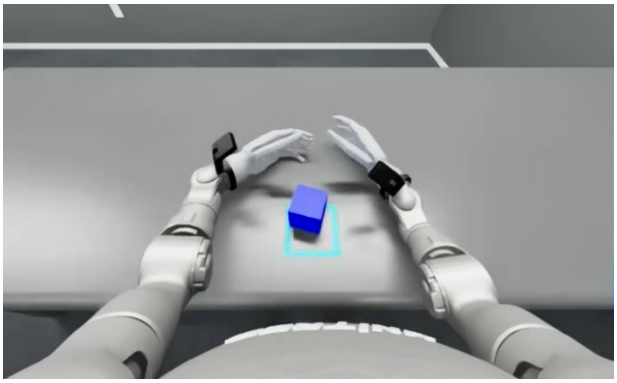

Figure 9: G1 successfully moving block to target zone using LoRA fine-tuned model. Head view.

5.4 Quantitative Analysis

To understand why LoRA succeeds where full fine-tuning fails, consider the relationship between model capacity and dataset size. Let N be the number of training samples, P the number of trainable parameters, and $\mathcal{L}$ the training loss. For a dataset with limited diversity, full fine-tuning with large P allows the model to memorize training examples:

$$\mathcal{L}_{\text{train}} \to 0 \quad \text{as} \quad P/N \to \infty \tag{2}$$

However, memorization destroys the pretrained representations that enable generalization. With LoRA, the effective parameter count is bounded:

$$P_{\text{LoRA}} = r \cdot \sum_i (d_i + k_i) \ll P_{\text{full}} \tag{3}$$

For rank $r = 16$ applied to transformer attention layers, this reduces trainable parameters by approximately 100–1000×, preventing memorization while allowing task-specific adaptation.

Metric	Full FT	LoRA
Training Steps	100,000	8,000
Training Time	10 hours	<1 hour
Final Loss	∼0	Normal decay
Unique Trajectories	No	Yes
Language Response	No	Yes
Task Success	0%	non-zero %

Table 1: Comparison of full fine-tuning vs. LoRA on 14 teleoperation episodes (15,791 frames).

5.5 Stabilization Solution

GR00T N1.5 only controls the upper body. Without lower-body stabilization, manipulation actions shift the center of mass beyond the support polygon, causing falls. The implemented solution increases joint stiffness K_p and damping K_d for lower-body joints:

$$\tau_i = K_p(q_i^* - q_i) + K_d(\dot{q}_i^* - \dot{q}_i) \tag{4}$$

with $q^* = q_{\text{init}}$ and $\dot{q}^* = 0$ for leg joints, effectively fixing the lower body in place.

6 Discussion

6.1 Key Finding: Parameter Efficiency is Essential

The central insight from this project is that parameter-efficient fine-tuning is not just computationally convenient—it is essential for small-dataset regimes. Full fine-tuning's 250× loss drop followed by stagnation indicates the model rapidly memorized the 14 training trajectories rather than learning generalizable manipulation skills.

This has practical implications for robotics research. Collecting thousands of demonstrations for every new robot or task is often infeasible. LoRA and similar methods enable foundation models to transfer their pretrained knowledge while adapting to new embodiments with minimal data.

6.2 Data Quality Over Quantity

The SO-101 experiments and subsequent G1 work demonstrated that GR00T works correctly when provided with compatible data. The initial G1 failures stemmed not from model limitations but from data incompatibility—using externally-collected demonstrations with different normalizations, action spaces, and camera configurations.

For small datasets, quality and consistency matter more than quantity. The 14 VR-teleoperated episodes succeeded because they were collected with the exact simulation setup used for inference.

6.3 VLA Generalization Limits

A key realization: current VLAs lack complex generalization. They cannot transfer across substantially different scenes or task spaces—only small perturbations from the training distribution. This means sim-to-real transfer requires high-fidelity simulation matching, and any deployment scenario must be well-represented in training data.

6.4 Diagnostic Methodology

The SO-101 "side project" proved invaluable. By validating GR00T on simpler hardware first, I confirmed the model architecture was sound before investigating G1-specific issues. This systematic approach—isolating variables by testing on progressively simpler systems—is transferable to debugging any complex robotics pipeline.

6.5 Lessons Learned

1. **Start simple:** Validate on easier platforms before tackling complex ones
2. **Match train/test exactly:** Data collection setup must match inference setup
3. **Prefer parameter efficiency:** LoRA should be default for small datasets
4. **Check hardware configs:** Undocumented settings (like `Goal_Time=0`) can block progress
5. **VLAs need compatible data:** Pre-collected internet data rarely works out-of-box

7 Future Work

7.1 Immediate Extensions

LeVERB Integration: The current lower-body stabilization is static. Integrating LeVERB [5] would enable whole-body locomotion, allowing the G1 to walk to manipulation targets while maintaining balance.

Sim-to-Real Transfer: Once the pipeline is ready for physical G1 deployment I would like to try sim-to-real transfer to the physical G1.

Expanded Task Diversity: Additional tasks (pouring, tool use, bimanual manipulation) would test generalization.

7.2 Data Scaling with DreamGen

A promising direction is synthetic data generation via video world models [6]:

1. Collect small real-world dataset
2. Train video world model on captured trajectories
3. Generate imagined trajectories ("dreaming")
4. Filter for physical plausibility
5. Label with inverse dynamics model
6. Train policy on real + synthetic data

This could dramatically expand training data without additional teleoperation.

7.3 System-1/System-2 Architecture

The long-term goal is a hierarchical intelligence architecture:

- **System 2 (High-Level):** LLM-based planner for task decomposition and goal monitoring
- **System 1 (Low-Level):** Parallel execution of specialized reactive policies—GR00T for manipulation, LeVERB for locomotion, and deterministic policies for safety and get-up if fall

This would enable complex multi-step tasks like "prepare a chemical solution" requiring reasoning, navigation, and precise manipulation.

8 Conclusion

This project established a complete simulation-to-reality pipeline for humanoid manipulation using vision-language-action models. The key technical contribution is demonstrating that LoRA fine-tuning enables GR00T N1.5 to generalize from only 14 demonstrations while full fine-tuning fails catastrophically on identical data.

The infrastructure developed—VR teleoperation, ROS2 integration, multi-GPU training, VideoMimic pose estimation, and Isaac Sim environments—provides a foundation for continued research toward autonomous humanoid operation. The diagnostic methodology using the SO-101 arm offers a template for systematic debugging of complex robotics systems.

What began as an attempt to deploy a foundation model on a humanoid robot became an education in the current limits of embodied AI. Physical intelligence remains harder than language-based AI—robots must contend with real-world physics, hardware failures, and the challenge of collecting compatible training data. Yet this difficulty is precisely why embodied AI may be more consequential: enabling machines to learn through physical trial and error could push all machine intelligence forward in ways that text alone cannot achieve.

References

[1] Brohan, A., et al. "RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control." *arXiv:2307.00329*, 2023.

[2] Kim, S., et al. "OpenVLA: An Open-Source Vision-Language-Action Model." Project page: `https://openvla.github.io/`, 2024.

[3] NVIDIA. "Isaac GR00T: Generalist Robot Foundation Model." GitHub repository: `https://github.com/NVIDIA/Isaac-GR00T`, 2024.

[4] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, L., & Chen, W. "LoRA: Low-Rank Adaptation of Large Language Models." *arXiv:2106.09685*, 2021.

[5] Xue, Z., et al. "LeVERB: Humanoid Whole-Body Control with Latent Vision-Language Instruction." *arXiv:2506.13751*, 2025.

[6] Jang, E., et al. "DREAMGEN: Unlocking Generalization in Robot Learning through Video World Models." *arXiv:2505.12705*, 2025.

[7] Physical Intelligence, et al. "$\pi_{0.5}$: A Vision-Language-Action Model with Open-World Generalization." *arXiv:2504.16054*, 2025.

[8] Shukor, M., et al. "SmolVLA: A Vision-Language-Action Model for Affordable and Efficient Robotics." *arXiv:2506.01844*, 2025.

[9] Lepert, M., Fang, K., & Bohg, J. "Phantom: Training Robots Without Robots Using Only Human Videos." *arXiv:2503.00779*, 2025.

[10] Li, Y., et al. "H2R: A Human-to-Robot Data Augmentation for Robot Pre-training from Videos." *arXiv:2505.11920*, 2025.

[11] Chen, X., et al. "ViSA-Flow: Accelerating Robot Skill Learning via Large-Scale Video Semantic Action Flow." *arXiv:2505.01288*, 2025.

[12] Chen, A., et al. "VideoMimic: Real-World Video to Robot Action." University of California, Berkeley, 2024.

[13] Makoviychuk, V., et al. "Isaac Gym: High Performance GPU-Based Physics Simulation for Robot Learning." *arXiv:2108.10470*, 2021.

[14] Jain, A., et al. "Vid2Robot: Video-Conditioned Policy Learning with Cross-Attention Transformers." Project page, 2024.

[15] Cheng, R., et al. "NaVILA: Legged Robot Vision-Language-Action Model for Navigation." Robotics: Science and Systems (RSS), 2025.

[16] Physical Intelligence. "openpi: Open-Source Vision-Language-Action Models." GitHub repository: `https://github.com/physical-intelligence/openpi`, 2025.

[17] Unitree Robotics. "unitree_sim_isaaclab: Unitree Simulation Environment Based on Isaac Lab." GitHub repository, 2025.

[18] Venard, J. "A Step-by-Step Guide to Training and Testing the BB-ACT AI Model." phospho.ai Blog, June 2025.