Lucid-XR: An Extended-Reality Data Engine for Robotic Manipulation

Yajvan Ravan[1,2]*, Adam Rashid[1]*, Alan Yu[1,2], Kai McClennen[1,2], Gio Huh[2,3], Kevin Yang[2,4], Zhutian Yang[1], Qinxi Yu[5], Xiaolong Wang[5], Phillip Isola[1†], Ge Yang[1,2†*]

[1]MIT CSAIL, [2]FortyFive Labs, [3]Caltech, [4]Harvard University, [5]UC San Diego

[*]*Equal contribution* [†]*Equal advising*

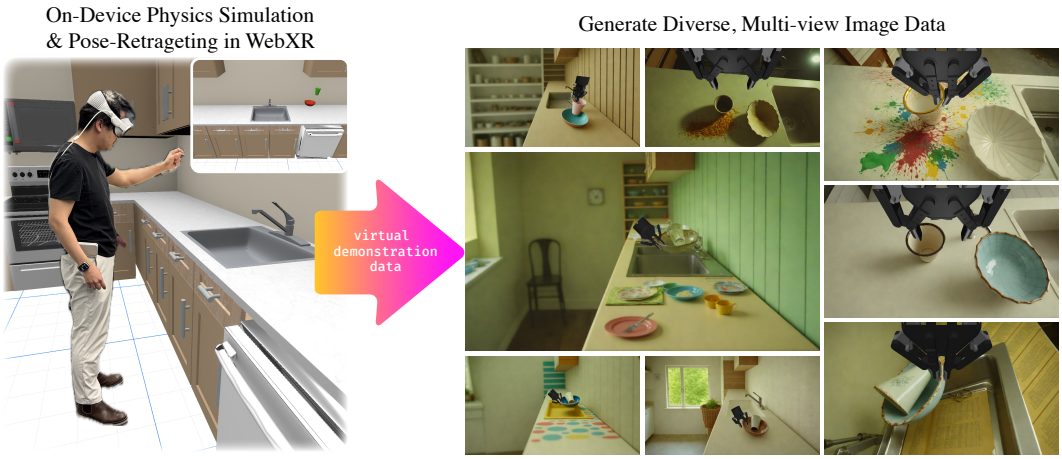


Figure 1: **An Extended Reality Data Engine for Robotic Manipulation.** *Left*: we deliver physics simulation to run directly on the XR devices via the web browser, to enable internet-scale crowd-sourcing of demonstration data collection. *Right*: Our GenAI-powered synthetic data engine creates steerable, diverse, and realistic multi-view visual data to train real-world robots.

Abstract: We introduce Lucid-XR, a generative data engine for creating diverse and realistic-looking multi-modal data to train real-world robotic systems. At the core of Lucid-XR is vuer, a web-based physics simulation environment that runs directly on the XR headset, enabling internet-scale access to immersive, latency-free virtual interactions without requiring specialized equipment. The complete system integrates on-device physics simulation with human-to-robot pose retargeting. Data collected is further amplified by a physics-guided video generation pipeline steerable via natural language specifications. We demonstrate zero-shot transfer of robot visual policies to unseen, cluttered, and badly lit evaluation environments, after training entirely on Lucid-XR's synthetic data. We include examples across dexterous manipulation tasks that involve soft materials, loosely bound particles, and rigid body contact. Project website: https://lucidxr.github.io

Keywords: Extended-reality, world-model, synthetic data

1 Introduction: From Atoms to Bits

When viewed from afar, training a robot controller is not too different from making a movie, in that both involve carefully curating content. Feature films over the years have transitioned steadily from practical special effects in the physical world to digitally created virtual special effects, driven primarily by creators' demand for even greater freedom in storytelling [1, 2, 3, 4]. Hardening robotic

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

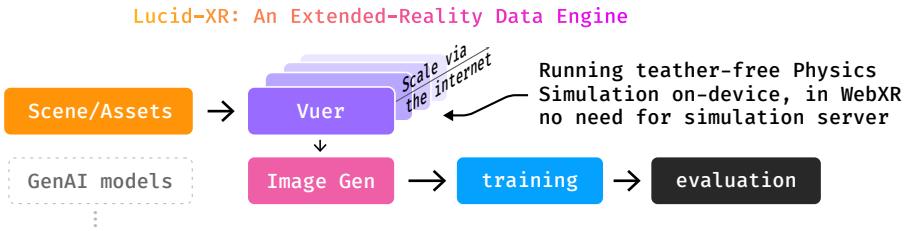


Figure 2: **System Schematic of the Lucid-XR Pipeline.** The results in this paper require hand-crafted, but basic 3D scenes. The data collection is done collectively by the authors. The resulting simulated datasets are augmented by a generative pipeline powered by language and text-to-image models.

systems for real-world deployment requires a similar level of control over the training environment to cover rare but mission-critical events that, by definition, are scarce in real-world datasets. What makes robotics more difficult is that digitally creating millions of realistic-looking virtual worlds at the scale required for our robots to generalize to the real world is infeasibly expensive.

In this work, we introduce Lucid-XR, an extended-reality (XR) data engine for robot manipulation. Lucid-XR uses a generative image pipeline to convert human demonstration data collected in low-fidelity virtual environments into diverse and visually realistic data to train the robot. Key to our vision is to enable internet scale deployment of real-time multi-physics simulations through the web browser to enable the crowd-sourcing of unlimited human demonstration data. Furthermore, virtual demonstrations in sparsely populated 3D environments alone are insufficient for training real-world computer vision systems. We leverage language and text-to-image generative models to construct a physics-guided video generation pipeline that amplifies a small number of simple designs into millions of diverse and realistic-looking multi-view images for the robot.

One challenge that makes crowd-sourcing teleoperation difficult is that retargeting human poses to robot form factors that are kinematically different involves writing custom computer code for each robot and setting up a server. We solve this problem by leveraging an inverse kinematics solver, built-in to MuJoCo, and allowing users to define bindings between motion capture (MoCap) sites and robot parts in the markup schema.

Our contributions are three-fold: first, moving physics simulation onto the XR device, to deliver a latency-free multi-physics simulation in an immersive environment via the open internet; second, a way to retarget human pose data to virtual robots, without requiring a custom computer program; and third, a demonstration of a policy trained on Lucid-XR's synthetic data deployed on scanned digital twins of real environments.

2 A Touch of Physics in Extended Reality (XR)

Modern internet browsers are de facto operating systems built according to a commonly agreed-upon programming interface. The key limitation is the constraint on compute and memory, as the device's operating system needs to throttle the browser process to maintain a smooth user experience while handling a multitude of system and application tasks. In the past decade, however, the competition between vendors has significantly improved browser performance; on newer extended-reality devices, the browser receives first-class support in features sets, system resource limits, and performance.

Lucid-XR builds on three web standards. First, to ensure simulations are *fast*, we compile MuJoCo into WebAssembly, bypassing the single-threaded V8 engine and achieving near-native speed directly on XR hardware. Second, we built upon the web-XR standard, a user-interaction programming interface that is shared across device vendors. This enabled us to support *hands*, *motion controllers*, and XR devices from as low as $300 to $4000. The third web standard is webGL. We wrote a performant rendering, interaction, and programming interface from scratch using react-three/fiber, a modern lightweight and performant 3D framework with strong community backing. This has en-

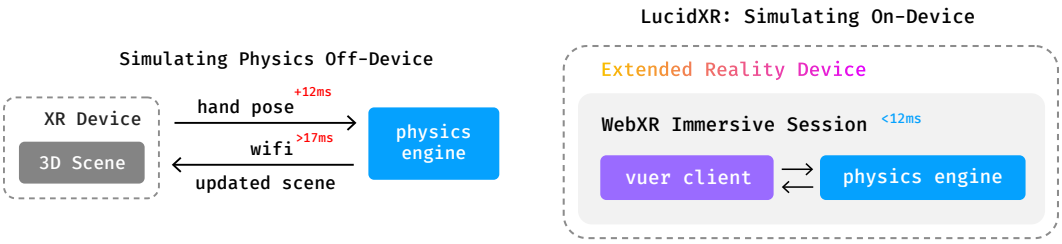


Figure 3: **Moving physics simulation on-device enables untethered access to immersive simulations.** The key benefits are twofold: first, it enables the simulation of deformable objects that involve modifying a large amount of mesh data that are too slow to send over WiFi. Second, it *eliminates* delays due to network latency, allowing the simulation to run at the device's native frame rate.

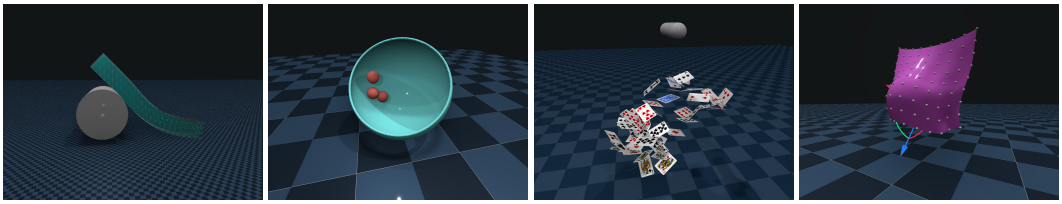


Figure 4: **Multiple types of physics running natively inside the browser in real-time.** From left to right: flexible material interacting with a solid; Signed-distance function (SDF) based collision solver for non-convex shapes; A deck of cards interacting with air/wind; Soft skin material interacting with a solid. Rendering and simulation are both native in a web browser.

abled real-time visualization of complex scenes and deformable objects. The current technological landscape is fast-evolving — in the near future, when webGPU becomes fully supported, we will be able to parallelize the physics simulation using compute shaders with hardware acceleration. This will enable more complex physics that involve a large quantity of interacting particles, deformable vertices, or liquid.

2.1 Multi-Physics Simulation in Vuer

At the core of Lucid-XR is *vuer*, a simulator-agnostic XR framework running in the browser. We use MuJoCo [5] compiled to WebAssembly with a custom react-three/fiber front-end, achieving real-time simulation at 90 fps on Apple Vision Pro. Data is collected at 25 fps; each step takes under 12 ms to simulate, so latency is negligible unless overloaded. The MuJoCo *decimation* parameter (5–20 steps per frame) controls simulation fidelity. Rendering runs natively on the XR device at 90 fps for user comfort. Like Blender, *vuer* is simulator-agnostic and benefits as physics engines improve.

Flexible materials. Prior VR setups run physics on an external computer [6], which introduces latency when simulating flexible or particle-rich scenes. Running physics on-device eliminates this bottleneck and enables interactive deformable-object simulation (Fig. 4).

Fluid forces. MuJoCo's integrated fluid model [7] supports wind and fluid interactions with rigid and deformable objects. For example, Fig. 4c shows poker cards falling against a capsule under air resistance.

Collision without convex decomposition. We inculde the signed-distance function (SDF) collision solver in Mujoco by default, removing the need for convex decomposition during modeling. This simplifies setup, at the cost of higher runtime compute and memory.

2.2 Precise Interactions at A Distance: Hitchhiking Controllers

The virtual world presents an opportunity to define new ways of interacting with robots beyond the constraint of real-world physicality. When the robot is situated far from the user's home location in VR, naïvely controlling those grippers by "grasping" them from afar yields poor user experience

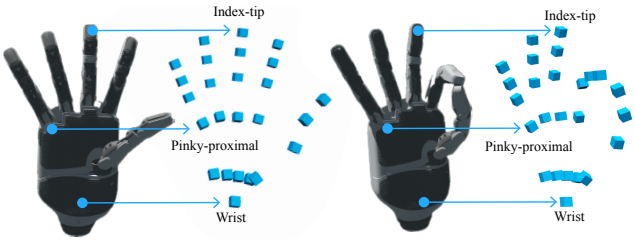


Figure 5: **Controlling Hand Pose via Motion Caption Sites.** We specify mocap sites by first aligning the proximal joints, and scale the hand so that fingers are similar in size as the robot hand. We then weld the SE(3) pose of the fingertip to similar sites on the robot hand and the wrist. We adjust the torque scale to balance tracking of the position and the rotation portion of the pose.

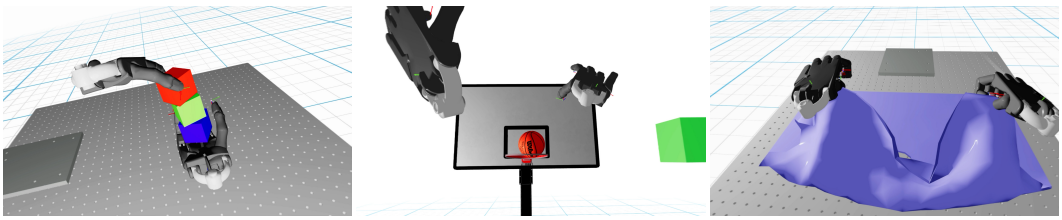


Figure 6: **Handling dynamic tasks and deformable objects.** The on-device retargeting is accurate enough to balance three blocks on one hand; is fast enough to handle dynamic tasks such as throwing a basketball, and handle deformable objects for cloth folding.

because the hand tracking error gets amplified as the distance increases. We solve this problem by applying SE(3) transformation to the target gripper in the local frame of the mocap site. This design took inspiration from the *hitchhiking hand* [8], and let the activate MoCap sites at a distance by looking at the object and clicking on it. Once activated, the user can engage with the site via a natural grasping gesture, where they close the lower three fingers.

2.3 On-Device Retargetting for Dexterous Hand Control

The problem of retargeting kinematic poses between humans and robots across different body kinematics is a common problem shared between dexterous manipulation, locomotion and humanoid whole-body control [9]. Existing solutions rely on running the kinematics solver separately on a server [10], making it hard to scale. Our solution involves binding mocap sites to the tip of each finger and utilizing the relative pose to the wrist joint, as well as the action space.

This hand control scheme is very general. We found that it worked well for all of the robot hands that we experimented with. We designed a schema-based programming interface for users to specify custom bindings between mocap sites or geometric bodies with landmarks and gestures of the hand in python.

2.4 Porting Existing Environments

Take Robocasa as an example – you can extract the XML from MuJoCo by calling `env.get_xml`, and then iterate through the `'file='` attributes to collect all of the assets. You can then load this file bundle by simply dragging and dropping it into Vuer. We were able to extract scenes from RoboHive, RoboCasa, RoboSuite, and MuJoCo Menagerie (see 7) .

3 Synthesizing Diverse Manipulation Data from Virtual Demonstrations

The third component of Lucid-XR is a generative engine that converts the virtual human demonstrations collected in the sim into diverse and realistic-looking multiview image datasets for the robot.

Figure 7: (a) Spot robot from Menagerie (b - c) RobotCasa Scenes

3.1 Generating Realistic Images from Virtual Demonstrations.

Figure 8 shows our setup for generating realist-looking images. We follow the LucidSim [11] recipe that starts with a collection of diverse text prompts collected from chatGPT, and use the semantic mask labels from the physics simulation to control the image generation process. Prompts for generation are sourced en masse from ChatGPT via a meta-prompt (see appendix). Prompts for the background tend to be more complex. In alignment with observations made by prior works [11], we found it is key to generate these images from a diverse set of text prompts.

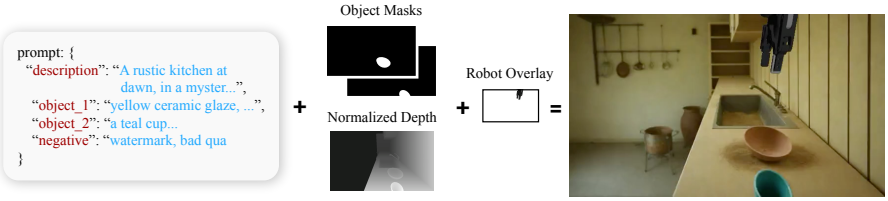


Figure 8: **Image generation pipeline.** We apply semantic masking and depth conditioning to precisely control the scene.

3.2 Demonstration Augmentation

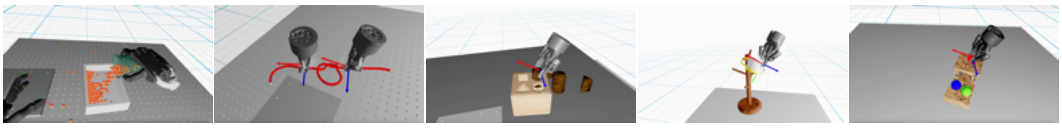
We designed an interface to generate MuJoCo XMLs procedurally in Python, allowing for a user to quickly compose scenes in LucidXR and multiply data. This allows robust control of the distribution of initial scene configurations during data collection.

Repositioning cameras post-demonstrations Camera poses can be modified very quickly in the Python script without needing data to be recollected. Collected trajectories can simply be played back and rendered from the new camera. This allows for easy visualization and quick iteration of image views for training. The simulator also allows for rendering the ground truth depth, which we use to warp images post-rendering. Given the exact camera extrinsics, intrinsics, and depth we can compute the optical flow for nearby poses. This is a crucial part of the sim-to-real pipeline to prevent sensitivity to the camera pose.

Trajectory warping for repositioning objects and robots Similar to MimicGen [12] we select keypoints in the trajectories, and move them around within a specified distribution. With linear interpolation for pose + spherical interpolation for rotation, this creates an entirely new demonstration. This allows us to reposition objects, robots, and change the distribution of initial scene setup post-demonstration. We use this augmentation to make trained policies more robust to variations in object position.

4 Results

We created contact-rich environments to evaluate various types of on-device physics simulation in Lucid-XR. Each environment tests a distinct type of physical interaction (see Fig. 9). For each task the initial state is randomized within a small area.



(a) Pouring cup. (b) Tying a knot. (c) Tight insertion (d) Mug tree (e) Ball sorting toy

Figure 9: **Lucid-XR can simulate contact-rich manipulation of diverse physics.** (a) granular materials to simulate pouring liquid, (b) deformable materials when tying a knot, (c) tight tolerances between objects in contact, in a shape insertion task. (d) placing a mug onto a drying rack (e) a ball sorting toy.

- **Block Stacking:** This task requires creating a three block stack with a dextrous hand.
- **Pour Liquid:** This task requires picking up a cup with a dextrous hand, handing it off to another hand, and pouring into a sink – involves a large scene, and granular particles flow.
- **Ball Sorting:** This task requires sorting three balls by color into a toy – Involves mixed rigid–particle collision in a toy sorter.
- **Knot Tying:** This task requires tying a knot on a suspended rope with a single two-fingered gripper – Involves deformation of a flexible string into a knot with self-contact
- **Kitchen-Sink:** This task requires picking up a cup, placing it onto a bowl, and then placing both into the kitchen sink – Involves a much larger scene, using SDFs, and is longer-horizon.
- **Mug Tree:** This task requires picking up a mug with a single two-fingered gripper and placing it on a tree rack – Involves modeling collisions with concave shapes using the SDF plugin,

4.1 Data Collection and Learning Setup

We record observations and actions as SE(3) mocap poses at 25 Hz; rotations use the 6D representation of [13]. Demonstrations are embodiment-agnostic, so behavior-cloned policies can deploy on any two-finger gripper. Policies take proprioception as well as either a wrist RGB view or three fixed RGB views of the workspace and output chunked absolute end-effector poses; we train with batch size 64.

ACT. We use standard ACT [14]: 15k updates, chunk size 25, learning rate 10^{-4}, and a DETR-style VAE backbone [15] (4 encoders, 1 decoder, head dim 128, FFN 256). Images are color-jittered. At evaluation we apply temporal aggregation [14].

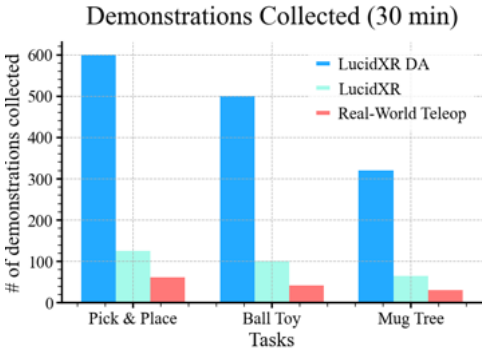
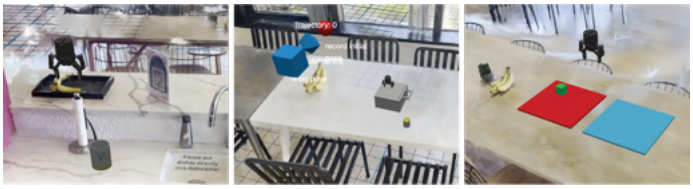


Figure 10: **Amount of data collected for LucidXR vs Real-World Teleop.**

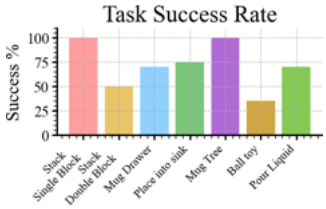
Diffusion. We also train a score-based action denoiser [16, 17] with a 1D U-Net [18] conditioned on image features via FiLM [19]. Optimization uses AdamW with an exponential LR schedule $10^{-3} \rightarrow 10^{-5}$. Inference performs 1000 denoising steps and executes actions by chunks.

4.2 Comparing Data Collection Speed in Virtual and the Real-world

Resetting the scene in the real world involve manually replacing the object, moving the gripper, and safety checks. In vuer the entire physics simulation and reset logic runs on-device, so the user can reset the environment by pressing a button. This provides an uninterrupted data capture for the full thirty minutes. We collected 30 minutes of demonstrations for each of three tasks in both settings. As shown in 10, participants collected roughly $2\times$ more demonstrations in Lucid-XR than with real-world teleoperation. When we further applied the augmentation pipeline, the effective dataset size increased to about $5\times$ that of the real-world baseline.



(a) **Real-to-sim evaluation.** We made 3D Gaussian scans of real-world kitchen environments to stage the evaluation environments, as a rough estimate of the generalization capability of the trained policy.



(b) Success rate for each manipulation task, as measured in our extended-reality data engine.

Figure 11: Evaluation results from real-to-sim transfer (left) and per-task performance in simulation (right).

Kitchen Clearing	**Base Env.**	**Low Clutter**	**High Clutter + Noise**
ACT Policy	100%	0%	0%
ACT + LucidSim	100%	90%	25%

Table 1: Evaluation scores for the kitchen clearing task with unseen real-life meshes overlaid on the original environment. We score the Kitchen Clearing task based on two pick-and-places, allowing for 4 points per run.

4.3 Real-to-Sim Evaluation

We made a small number of 3D Gaussian models of kitchens with real-life clutter, and use them as staging environments for our simulated eval. Although the Lucid-XR policy has never seen data from these environments, it out-performed the same policy trained on simulated data (see Fig.11).

4.4 Sim-To-Real Evaluation

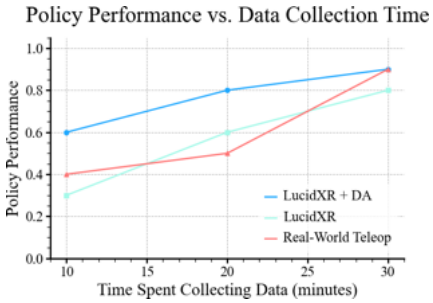
We trained three policies to accomplish pick and place using datasets collected over 10, 20, and 30 minutes. For each duration, one dataset was collected in LucidXR and rendered with generative imagery and another via real robot teleoperation where an Oculus headset was used to control the end-effector position. We further augmented the LucidXR dataset with methods from 3.1. We then evaluated all policies on a real robot in the same environment that the real dataset was collected on. Results showed that policies trained on purely synthetic data performed comparably to those trained on real data (see Fig. 12).

To further test robustness, we repeated the evaluation under modified visual conditions. Specifically, we varied the lighting, color, and replaced the original wooden tabletop with either a textured tablecloth or a black tablecloth. Under these changes, the policy trained only on real-world demonstrations failed to generalize, while the LucidXR-trained policies maintained high success rates and outperformed the real-world baseline.(see Fig. 12).

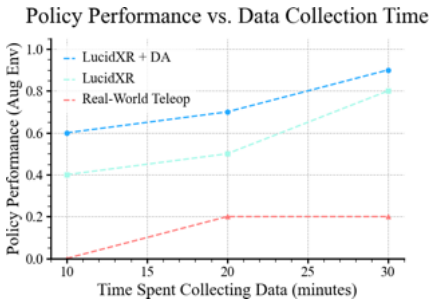
5 Related Works

Extended reality for robot teleoperation and data collection.

XR environments have been used to scale robot learning via teleoperation, yet each faces bandwidth or realism limits. [8] introduced "hitchhiking hands" for remote interaction, letting users control virtual avatars by gaze in VR. Our hitchhiking controller builds on this idea, enabling distant robot gripper control through XR hands. Hand-overlay methods [20, 21, 22, 23] and mobile-AR approaches [24, 25] overlay virtual robots on physical objects, but lack rich contact. Other AR/VR frameworks [10, 26, 27, 28] enable seamless teleoperation of physical robots, but throughput and environment diversity remain limited. Most closely related to our work are [29, 30], toolkits for VR demo collection. A key difference is that LucidXR runs simulations on-device, avoiding cloud



(a) Success rates for sim-to-real and real-to-real policies



(b) Success rates for sim-to-real and real-to-real policies on Augmented environment

Figure 12

latency that hinders dexterous, dynamic control in systems like DART and IRIS. We envision LucidXR as a platform for scaling imitation learning with generative models. Works such as [14] provide foundations for architectures like the Action Chunking Transformer.

Generative AI for Synthetic Data Augmentation . Recent works use generative AI across text, vision, and 3D assets to expand robot training data. Our image-generation pipeline builds on Lucid-Sim [11], which showed that locomotion policies trained entirely from generated images can deploy zero-shot in the real world. LucidSim uses text prompts and diffusion models to produce diverse scenes; we extend this with more semantic classes and randomized object geometry sampled from 3D assets. Other works [31, 32, 33, 34] generate diverse tasks but still lack realistic textures. These can be integrated into our pipeline to further expand data realism. In parallel, [35, 36, 37, 38] apply text-to-image generation to hallucinate new scenarios directly on real data, maintaining semantic consistency with robot objectives.

Large-scale imitation learning and data aggregation. Multi-institution efforts aggregate demonstrations but remain embodiment-specific. Open X-Embodiment collects 50k+ demos for RT-X policies [39]; RH20T [40] and DROID [41] add 100k+ crowdsourced trajectories. Despite scale, diversity across robots, environments, and tasks remains limited, falling short of what is needed for truly generalist policies analogous to vision or NLP foundation models.

6 Conclusion

In this work, we present Lucid-XR, a generative-AI-powered learning pipeline for producing generalizable visual policies for manipulation. We demonstrate the potential for virtual demonstrations in a simulated world to produce real-world robot policies that generalize across object instances, appearance, and lighting conditions. We also present new designs for interacting with virtual robots through VR controllers and hand gestures. We believe virtual demonstration data has the potential to scale across the internet and close the data gap for training a generally capable robot foundation model.

Deploying Across Embodiments. Our observation is that as long as the data is collected on the full embodiment, the learned controller will work, and in many cases data collected with embodiment-free (a floating gripper) transfer readily. We note that cross-embodiment transfer in Lucid-XR is limited only by inverse-kinematics and mobility. This is an avenue that future work can explore.

Acknowledgments

This work was supported by Amazon.com Services LLC, Award #2D-06310236. We would like to thank the reviewers for their constructive criticism and for helping make this paper more comprehensive.

7 Limitations

This work leaves a few rocks untouched. For instance, we rely on the controlling power of the same text prompt to generate consistent views. A key benefit of generating visual data is that it comes with paired text labels. Future iterations can benefit from the additional supervision that this pairing provides.

References

[1] J.-H. Ryu. Reality & effect: A cultural history of visual effects. *Communication Dissertations*, 2007.

[2] J. Turnock. Before Industrial Light and Magic: the independent Hollywood special effects business, 1968–75: Research Article. *New Rev. Film Telev. Stud.*, 7(2):133–156, June 2009.

[3] S. Das. The evolution of visual effects in cinema: A journey from practical effects to CGI. *Journal of Emerging Technologies and Innovative Research*, 10(11):303–309, 2023.

[4] B. Murodillayev. The impact of visual effects on the cinema experience: A comprehensive analysis. *Art Des. Rev.*, 2024.

[5] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. doi:10.1109/IROS.2012.6386109.

[6] Y. Park, J. S. Bhatia, L. L. Ankile, and P. Agrawal. Dexhub and dart: Towards internet scale robot data collection. *ArXiv*, abs/2411.02214, 2024. URL https://api.semanticscholar.org/CorpusID:273821640.

[7] MuJoCo Documentation. Fluid forces. https://mujoco.readthedocs.io/en/latest/computation/fluid.html, 2025. Accessed: 2025-August-29.

[8] R. Ban, K. Matsumoto, and T. Narumi. Hitchhiking hands: Remote interaction by switching multiple hand avatars with gaze. In *SIGGRAPH Asia 2023 Emerging Technologies*, pages 1–2. 2023.

[9] X. Cheng, Y. Ji, J. Chen, R. Yang, G. Yang, and X. Wang. Expressive whole-body control for humanoid robots, 2024. URL https://arxiv.org/abs/2402.16796.

[10] X. Cheng, J. Li, S. Yang, G. Yang, and X. Wang. Open-television: Teleoperation with immersive active visual feedback. *arXiv preprint arXiv:2407.01512*, 2024.

[11] A. Yu, G. Yang, R. Choi, Y. Ravan, J. Leonard, and P. Isola. Learning visual parkour from generated images. In *8th Annual Conference on Robot Learning*, 2024.

[12] A. Mandlekar, S. Nasiriany, B. Wen, I. Akinola, Y. Narang, L. Fan, Y. Zhu, and D. Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations, 2023. URL https://arxiv.org/abs/2310.17596.

[13] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li. On the continuity of rotation representations in neural networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2019.

[14] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[15] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers, 2020. URL https://arxiv.org/abs/2005.12872.

[16] T. Garipov, S. D. Peuter, G. Yang, V. Garg, S. Kaski, and T. Jaakkola. Compositional sculpting of iterative generative processes, 2023. URL https://arxiv.org/abs/2309.16115.

[17] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2024.

[18] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis, 2022. URL https://arxiv.org/abs/2205.09991.

[19] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer, 2017. URL https://arxiv.org/abs/1709.07871.

[20] S. Chen, C. Wang, K. Nguyen, L. Fei-Fei, and C. K. Liu. Arcap: Collecting high-quality human demonstrations for robot learning with augmented reality feedback. *arXiv preprint arXiv:2410.08464*, 2024.

[21] N. Nechyporenko, R. Hoque, C. Webb, M. Sivapurapu, and J. Zhang. Armada: Augmented reality for robot manipulation and robot-free data acquisition, 2024. URL https://arxiv.org/abs/2412.10631.

[22] 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 *2024 19th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 333–342, 2024.

[23] Y. Yang, B. Ikeda, G. Bertasius, and D. Szafir. Arcade: Scalable demonstration collection and generation via augmented reality for imitation learning, 2024. URL https://arxiv.org/abs/2410.15994.

[24] J. Duan, Y. R. Wang, M. Shridhar, D. Fox, and R. Krishna. Ar2-d2: Training a robot without a robot. 2023.

[25] J. Wang, C.-C. Chang, J. Duan, D. Fox, and R. Krishna. Eve: Enabling anyone to train robots using augmented reality, 2024. URL https://arxiv.org/abs/2404.06089.

[26] J. van Haastregt, M. C. Welle, Y. Zhang, and D. Kragic. Puppeteer your robot: Augmented reality leader-follower teleoperation, 2024. URL https://arxiv.org/abs/2407.11741.

[27] A. Iyer, Z. Peng, Y. Dai, I. Guzey, S. Haldar, S. Chintala, and L. Pinto. Open teach: A versatile teleoperation system for robotic manipulation, 2024. URL https://arxiv.org/abs/2403.07870.

[28] A. Naceri, D. Mazzanti, J. Bimbo, Y. T. Tefera, D. Prattichizzo, D. G. Caldwell, L. S. Mattos, and N. Deshpande. The vicarios virtual reality interface for remote robotic teleoperation: Teleporting for intuitive tele-manipulation. *J. Intell. Robotics Syst.*, 101(4), Apr. 2021. ISSN 0921-0296. doi:10.1007/s10846-021-01311-7. URL https://doi.org/10.1007/s10846-021-01311-7.

[29] X. Jiang, Q. Yuan, E. U. Dincer, H. Zhou, G. Li, X. Li, J. Haag, N. Schreiber, K. Li, G. Neumann, and R. Lioutikov. Iris: An immersive robot interaction system, 2025. URL https://arxiv.org/abs/2502.03297.

[30] Y. Park, J. S. Bhatia, L. Ankile, and P. Agrawal. Dexhub and dart: Towards internet scale robot data collection, 2024. URL https://arxiv.org/abs/2411.02214.

[31] P. Katara, Z. Xian, and K. Fragkiadaki. Gen2sim: Scaling up robot learning in simulation with generative models, 2023. URL https://arxiv.org/abs/2310.18308.

[32] Y. Wang, Z. Xian, F. Chen, T.-H. Wang, Y. Wang, K. Fragkiadaki, Z. Erickson, D. Held, and C. Gan. Robogen: Towards unleashing infinite data for automated robot learning via generative simulation, 2024. URL https://arxiv.org/abs/2311.01455.

[33] Y. J. Ma, W. Liang, H.-J. Wang, S. Wang, Y. Zhu, L. Fan, O. Bastani, and D. Jayaraman. Dreureka: Language model guided sim-to-real transfer, 2024. URL https://arxiv.org/abs/2406.01967.

[34] Z. Chen, A. Walsman, M. Memmel, K. Mo, A. Fang, K. Vemuri, A. Wu, D. Fox, and A. Gupta. Urdformer: A pipeline for constructing articulated simulation environments from real-world images. *arXiv preprint arXiv:2405.11656*, 2024.

[35] T. Yu, T. Xiao, A. Stone, J. Tompson, A. Brohan, S. Wang, J. Singh, C. Tan, D. M, J. Peralta, B. Ichter, K. Hausman, and F. Xia. Scaling robot learning with semantically imagined experience, 2023. URL https://arxiv.org/abs/2302.11550.

[36] Z. Mandi, H. Bharadhwaj, V. Moens, S. Song, A. Rajeswaran, and V. Kumar. Cacti: A framework for scalable multi-task multi-scene visual imitation learning. *arXiv preprint arXiv:2212.05711*, 2022.

[37] J. Jang, S. Ye, Z. Lin, J. Xiang, J. Bjorck, Y. Fang, F. Hu, S. Huang, K. Kundalia, Y.-C. Lin, L. Magne, A. Mandlekar, A. Narayan, Y. L. Tan, G. Wang, J. Wang, Q. Wang, Y. Xu, X. Zeng, K. Zheng, R. Zheng, M.-Y. Liu, L. Zettlemoyer, D. Fox, J. Kautz, S. Reed, Y. Zhu, and L. Fan. Dreamgen: Unlocking generalization in robot learning through video world models, 2025. URL https://arxiv.org/abs/2505.12705.

[38] Z. Chen, S. Kiami, A. Gupta, and V. Kumar. Genaug: Retargeting behaviors to unseen situations via generative augmentation. *arXiv preprint arXiv:2302.06671*, 2023.

[39] O. X.-E. Collaboration, A. O'Neill, A. Rehman, A. Gupta, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, A. Tung, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Gupta, A. Wang, A. Kolobov, A. Singh, A. Garg, A. Kembhavi, A. Xie, A. Brohan, A. Raffin, A. Sharma, A. Yavary, A. Jain, A. Balakrishna, A. Wahid, B. Burgess-Limerick, B. Kim, B. Schölkopf, B. Wulfe, B. Ichter, C. Lu, C. Xu, C. Le, C. Finn, C. Wang, C. Xu, C. Chi, C. Huang, C. Chan, C. Agia, C. Pan, C. Fu, C. Devin, D. Xu, D. Morton, D. Driess, D. Chen, D. Pathak, D. Shah, D. Büchler, D. Jayaraman, D. Kalashnikov, D. Sadigh, E. Johns, E. Foster, F. Liu, F. Ceola, F. Xia, F. Zhao, F. V. Frujeri, F. Stulp, G. Zhou, G. S. Sukhatme, G. Salhotra, G. Yan, G. Feng, G. Schiavi, G. Berseth, G. Kahn, G. Yang, G. Wang, H. Su, H.-S. Fang, H. Shi, H. Bao, H. B. Amor, H. I. Christensen, H. Furuta, H. Bharadhwaj, H. Walke, H. Fang, H. Ha, I. Mordatch, I. Radosavovic, I. Leal, J. Liang, J. Abou-Chakra, J. Kim, J. Drake, J. Peters, J. Schneider, J. Hsu, J. Vakil, J. Bohg, J. Bingham, J. Wu, J. Gao, J. Hu, J. Wu, J. Wu, J. Sun, J. Luo, J. Gu, J. Tan, J. Oh, J. Wu, J. Lu, J. Yang, J. Malik, J. Silvério, J. Hejna, J. Booher, J. Tompson, J. Yang, J. Salvador, J. J. Lim, J. Han, K. Wang, K. Rao, K. Pertsch, K. Hausman, K. Go, K. Gopalakrishnan, K. Goldberg, K. Byrne, K. Oslund, K. Kawaharazuka, K. Black, K. Lin, K. Zhang, K. Ehsani, K. Lekkala, K. Ellis, K. Rana, K. Srinivasan, K. Fang, K. P. Singh, K.-H. Zeng, K. Hatch, K. Hsu, L. Itti, L. Y. Chen, L. Pinto, L. Fei-Fei, L. Tan, L. J. Fan, L. Ott, L. Lee, L. Weihs, M. Chen, M. Lepert, M. Memmel, M. Tomizuka, M. Itkina, M. G. Castro, M. Spero, M. Du, M. Ahn, M. C. Yip, M. Zhang, M. Ding, M. Heo, M. K. Srirama, M. Sharma, M. J. Kim, M. Z. Irshad, N. Kanazawa, N. Hansen, N. Heess, N. J. Joshi, N. Suenderhauf, N. Liu, N. D. Palo, N. M. M. Shafiullah, O. Mees, O. Kroemer, O. Bastani, P. R. Sanketi, P. T. Miller, P. Yin, P. Wohlhart, P. Xu, P. D. Fagan, P. Mitrano, P. Sermanet, P. Abbeel, P. Sundaresan, Q. Chen, Q. Vuong, R. Rafailov, R. Tian, R. Doshi, R. Mart'in-Mart'in, R. Baijal, R. Scalise, R. Hendrix, R. Lin, R. Qian, R. Zhang, R. Mendonca, R. Shah, R. Hoque, R. Julian, S. Bustamante, S. Kirmani, S. Levine, S. Lin, S. Moore, S. Bahl, S. Dass, S. Sonawani, S. Tulsiani, S. Song, S. Xu, S. Haldar, S. Karamcheti, S. Adebola, S. Guist, S. Nasiriany,

S. Schaal, S. Welker, S. Tian, S. Ramamoorthy, S. Dasari, S. Belkhale, S. Park, S. Nair, S. Mirchandani, T. Osa, T. Gupta, T. Harada, T. Matsushima, T. Xiao, T. Kollar, T. Yu, T. Ding, T. Davchev, T. Z. Zhao, T. Armstrong, T. Darrell, T. Chung, V. Jain, V. Kumar, V. Vanhoucke, V. Guizilini, W. Zhan, W. Zhou, W. Burgard, X. Chen, X. Chen, X. Wang, X. Zhu, X. Geng, X. Liu, X. Liangwei, X. Li, Y. Pang, Y. Lu, Y. J. Ma, Y. Kim, Y. Chebotar, Y. Zhou, Y. Zhu, Y. Wu, Y. Xu, Y. Wang, Y. Bisk, Y. Dou, Y. Cho, Y. Lee, Y. Cui, Y. Cao, Y.-H. Wu, Y. Tang, Y. Zhu, Y. Zhang, Y. Jiang, Y. Li, Y. Li, Y. Iwasawa, Y. Matsuo, Z. Ma, Z. Xu, Z. J. Cui, Z. Zhang, Z. Fu, and Z. Lin. Open X-Embodiment: Robotic learning datasets and RT-X models. https://arxiv.org/abs/2310.08864, 2023.

[40] H.-S. Fang, H. Fang, Z. Tang, J. Liu, J. Wang, H. Zhu, and C. Lu. Rh20t: A robotic dataset for learning diverse skills in one-shot. In *RSS 2023 Workshop on Learning for Task and Motion Planning*, 2023.

[41] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, P. D. Fagan, J. Hejna, M. Itkina, M. Lepert, Y. J. Ma, P. T. Miller, J. Wu, S. Belkhale, S. Dass, H. Ha, A. Jain, A. Lee, Y. Lee, M. Memmel, S. Park, I. Radosavovic, K. Wang, A. Zhan, K. Black, C. Chi, K. B. Hatch, S. Lin, J. Lu, J. Mercat, A. Rehman, P. R. Sanketi, A. Sharma, C. Simpson, Q. Vuong, H. R. Walke, B. Wulfe, T. Xiao, J. H. Yang, A. Yavary, T. Z. Zhao, C. Agia, R. Baijal, M. G. Castro, D. Chen, Q. Chen, T. Chung, J. Drake, E. P. Foster, J. Gao, D. A. Herrera, M. Heo, K. Hsu, J. Hu, D. Jackson, C. Le, Y. Li, K. Lin, R. Lin, Z. Ma, A. Maddukuri, S. Mirchandani, D. Morton, T. Nguyen, A. O'Neill, R. Scalise, D. Seale, V. Son, S. Tian, E. Tran, A. E. Wang, Y. Wu, A. Xie, J. Yang, P. Yin, Y. Zhang, O. Bastani, G. Berseth, J. Bohg, K. Goldberg, A. Gupta, A. Gupta, D. Jayaraman, J. J. Lim, J. Malik, R. Martín-Martín, S. Ramamoorthy, D. Sadigh, S. Song, J. Wu, M. C. Yip, Y. Zhu, T. Kollar, S. Levine, and C. Finn. Droid: A large-scale in-the-wild robot manipulation dataset. 2024.

Appendix

We include additional simulated environments; details on how we engineer the physics involved; in addition to details on the generative workflow in this appendix. We also include examples of the virtual demonstrations that we collected, and examples of the synthetic images used for training. A live version of the presentation is included in The accompanying video.

7.1 Examples of Dexterous Virtual Demonstrations

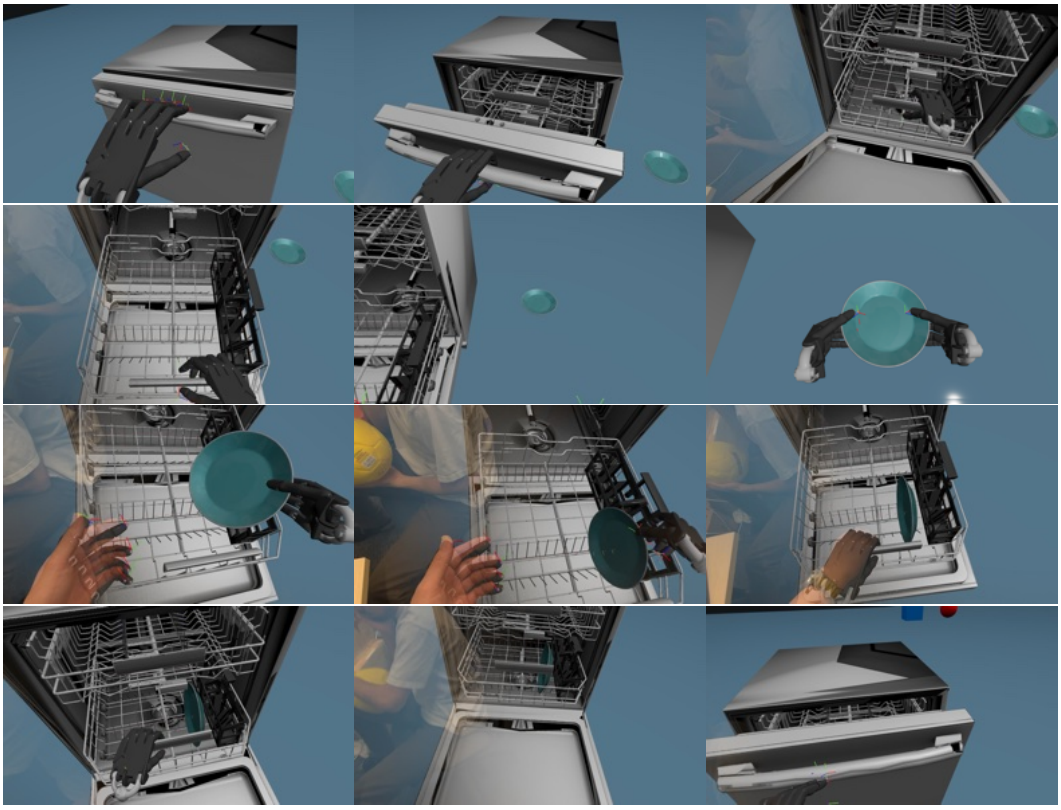


Figure 13: **Loading a Dishwasher.** Showing dexterous manipulation of a dishwasher with articulated doors a plate, and its interaction with the racks inside the dishwasher.

7.2 Examples of Policy Unroll in Simulation

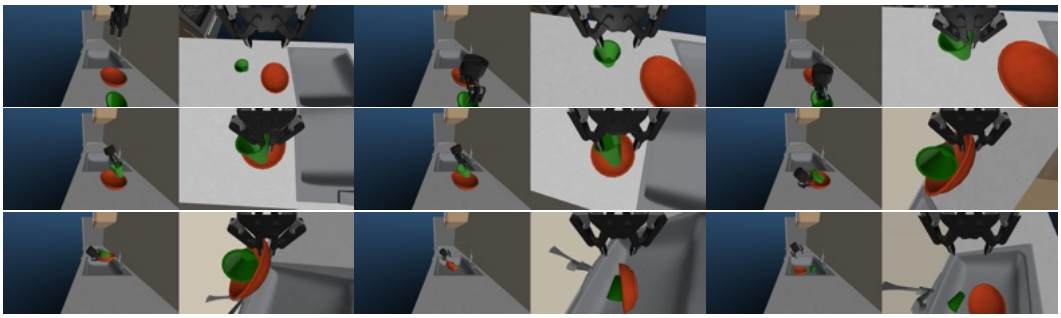


Figure 14: **Cleaning The Kitchen.** The robot learns to pick up a cup and stack it on top of the bowl, followed by placing both objects into the kitchen sink.

7.3 Examples of Learned Re-try Behavior

We noticed robust, re-try behavior from the learned policy. We present the image sequence in Figure. 15.

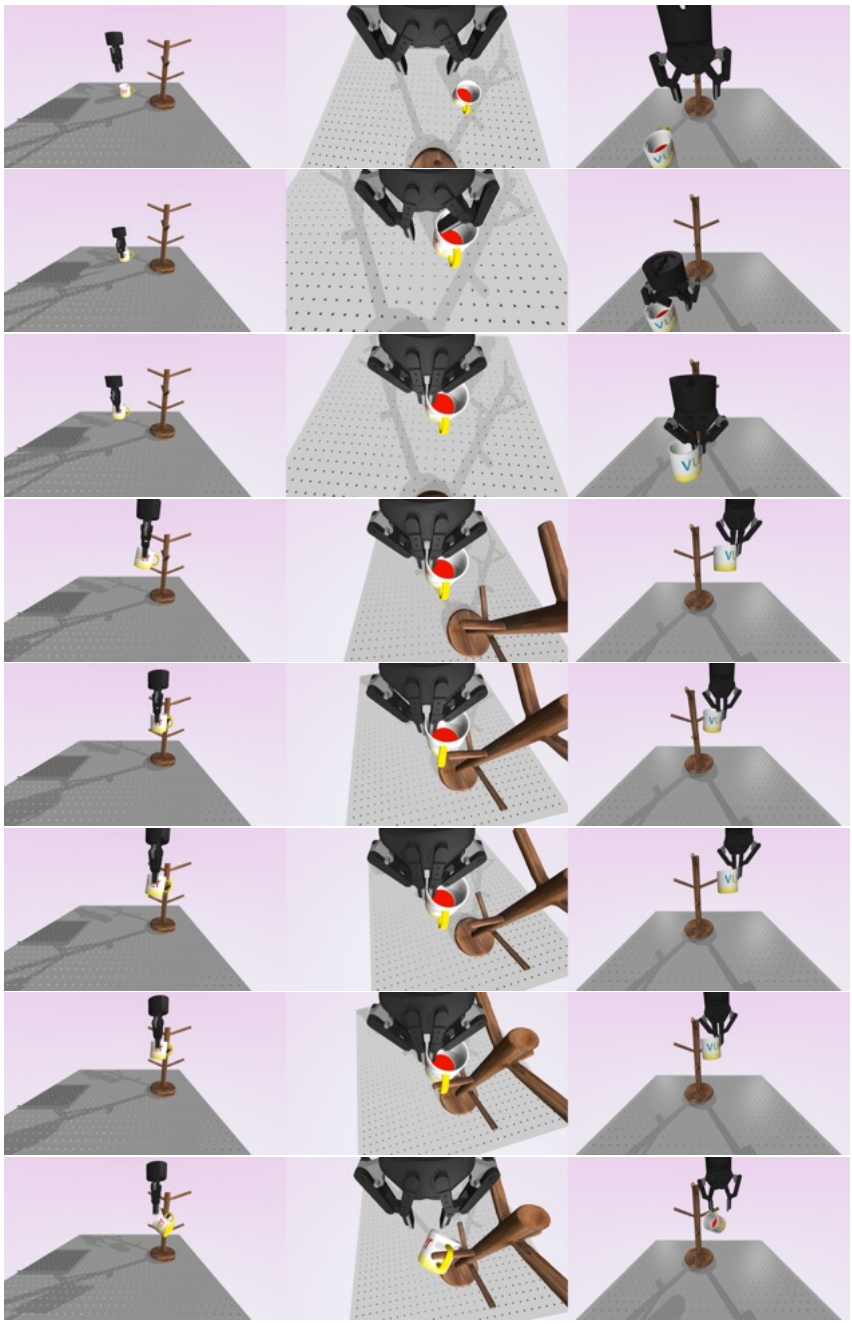


Figure 15: **Re-try Behavior in the Mug Tree Environment.** The policy came into contact twice: First during the picking up of the mug; the second time is when trying to hang the mug onto the arm of the mug tree drying rack.

7.4 Additional Examples of Synthetic Data

We include additional examples of synthetic images generated by the LucidXR pipeline in Figure. 16.

Figure 16: **Additional Examples of Generated Images from Lucid-XR.** Notice the control over lighting, geometry, and content diversity.

7.5 The Vuer Scene Description Language

Usage of the MuJoCo engine API in Python tends to follow an imperative pattern, where objects, material texture, and lighting are changed by mutating MuJoCo's physics and modeling buffers. To improve the readability and reusability of the MuJoCo simulation, we developed a declarative scene description language that treats the scene as a nested set of scene components that form a tree. This module, `vuer-mujoco`, enables the user to sketch out a scene via the following:

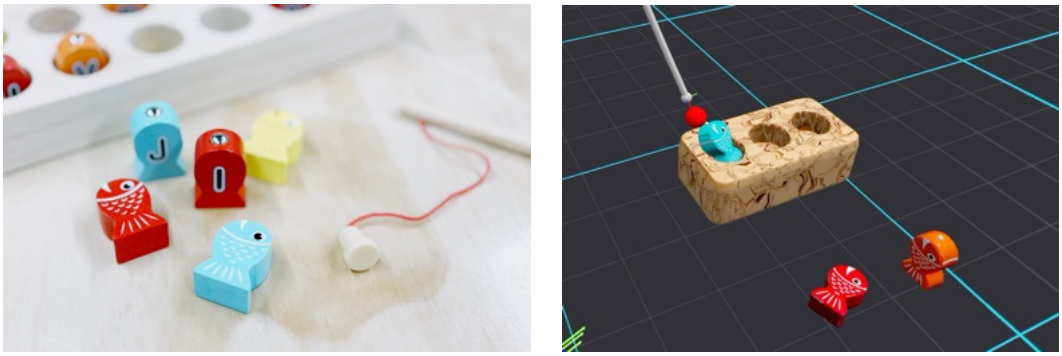
```
from vuer_mujoco import Box, DefaultScene, SimpleTable
from your_code import make_camera_rig

camera_rig = make_camera_rig()
lighting_rig = make_lighting_rig()

table = SimpleTable(pos=[0.0, 0.0, 0.0])

# a 10cm cube, initialized slightly above a table.
cube = Box(size=[0.01, 0.01, 0.01], pos=[0., 0.1, 0.02] + table.
    surface_origin)

scene = DefaultScene(
    *camera_rig.get_all_cameras(),
    lighting_rig.key, lighting_rig.fill, lighting_rig.back,
    table, cube, robot, robot.mocap_points,
)
```



(a) Real-world photo of the toy. (b) Details of simulated learning environment.

Figure 17: The real-world fishing toy set (left), versus the simulated learning environment used to collect data (right). The fishing rod has a magnetic hook attached to the end of the line that interacts with a ferromagnetic nail head embedded in the fish. We simulate this by defining adhesion actuators on a small sphere body attached to the fish geometry.

7.6 Tips on Engineering Synthetic Environments

Adhesion. Adhesive interactions are involved in many physical processes, including surface tension, magnetic interaction with ferro- and para-magnetic materials, interaction with sticky surfaces, such as an object against the silicone rubber material in a gripper.

We provide a fishing toy environment that simulates a toy fishing pole with a magnetic hook that the robot can use to pick up a wooden fish. This simultaneously involves interaction with flexible material – a rope. We set the weight of the fish pieces so that the adhesion is strong enough to lift them off the table, but still weak enough to detach with a slight shake of the fishing rod.

Generating 3D Assets. All of the 3D assets shown above are generated either from a text description of the object or stock images of the physical item found on Amazon.com. We use a free version of the meshy.ai service. The resulting 3D meshes usually contain a large number of faces, with physical dimensions that are off by two orders of magnitude. We post-process these 3D assets by first simplifying them in MeshLab, and then centering and rescaling using a custom script.

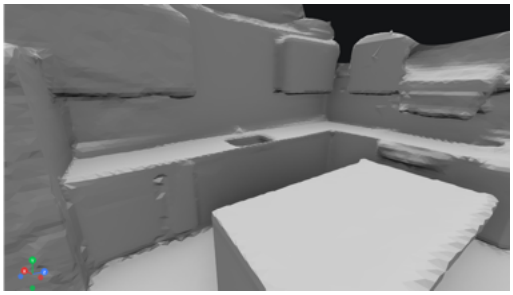
```
from vuer_mujoco import MuJoCoRope

rope = MuJoCoRope(
  postamble="""
    <actuator>
      <adhesion body="f1-magnet" ctrlrange="0.15 0.16" gain="1"/>
      <adhesion body="f2-magnet" .../>
      <adhesion body="f3-magnet" .../>
    </actuator>
  """,
)
```

7.7 Real-to-sim Evaluation Setup

We include two realistic testing environments: a) Clean Kitchen and b) Messy Kitchen. These environments serve as a more controllable proxy of the real-world experiments, to enable faster iteration on the data and learning pipeline.

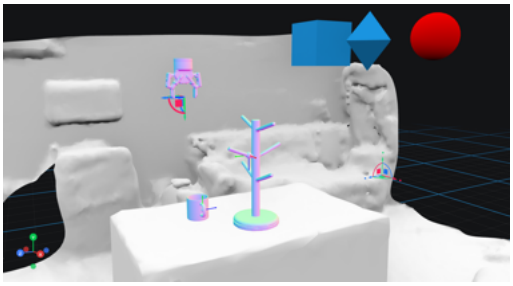
We align the 3D mesh from PolyCam to the MuJoCo environment (see Figure. 19) manually, shown in Figure 18.



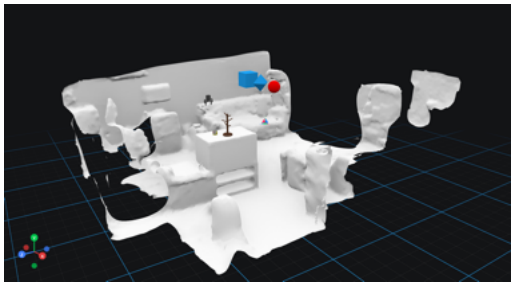
(a) Collision Geometry

(b) Render

Figure 18: **Clean Kitchen.** Contain no clutter. We place 3D object assets programmatically.



(a) Aligning the 3D mesh with the MuJoCo scene.



(b) The scene after alignment.

Figure 19: **Aligning scan with the physics environment.**

7.8 Image Generation Workflow

We provide the complete image generation workflow in JSON form in the supplementary material. This workflow can be loaded into ComfyUI as-is. A screenshot fo this workflow can be found in Figure. 20.

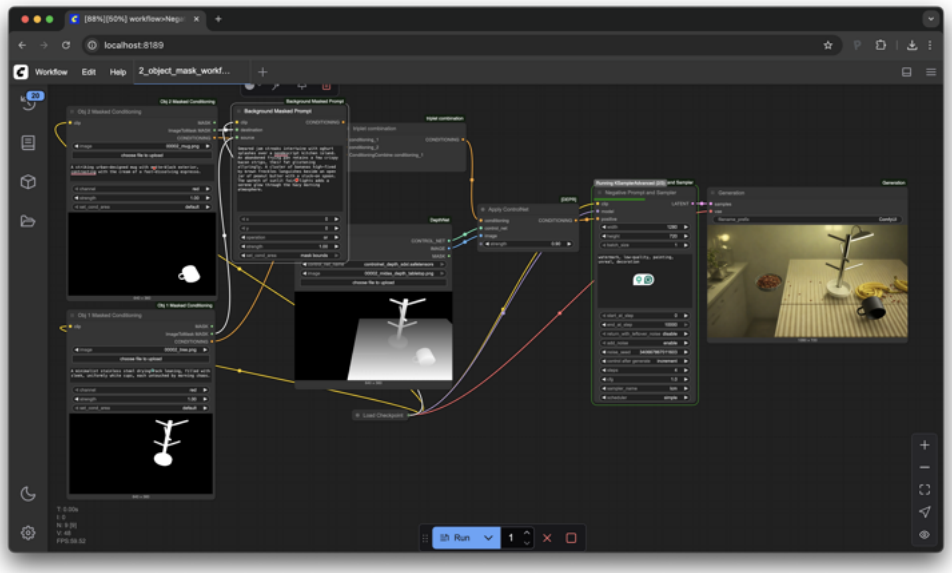


Figure 20: **Image Generation Workflow.** Using two object masks plus a normalized inverse depth image, we are able to control the geometry, lighting, and the composition of the generated images.

7.9 Training Details

We use the Action-Chunking transformer architecture from [14]. Our policy uses multiple backbones (for each input camera feed). The hyperparameters we used are in Table. 2.

Table 2: Training Hyperparameters.

learning rate	5e−5
batch size	32
number of encoder layers	4
number of decoder layers	7
feedforward dimension	3200
hidden dimension	512
number of heads	8
chunk size	10
KL-weight	10
dropout	0.1

7.10 Example Image Prompts

We produce a small number of randomly selected text prompts below.

Prompts for the kitchen cup/bowl Scene:

```
bowl = "A rich green patinated copper bowl, its surface embellished
    with intricate embossing echoing the craftsmanship of generations
    past, each detail wrought with precision."

cup = "A minimalist acrylic cup, hard-edged and visually weightless,
    mimicking the ethos of contemporary transparency."

background = "A chaotic ensemble of baking supplies lie scattered
    around, flour dusts the edges of a marble countertop, while
    reflective surfaces from an overhead light palette interact
    sporadically creating patterns on the brushed steel faucet."
```

Listing 1: **The Crossroad of Time.** Illustrating the intersection of aesthetic past and present through culinary objects. (Minimalist cup, antique bowl)

```
bowl = "A ceramic bowl with swirling patterns of cobalt blue,
    sunflower yellow, and rose pink glazes, with a high-gloss finish."

cup = "A teacup marked by abstract patches of emerald green and
    crimson red, seemingly creating a kaleidoscopic effect."

background = "A backdrop filled with chaotic elements \u2014 a
    countertop cluttered with spatulas, over-ripe fruit, and colorful
    cracked ceramic tiles leaning against the wall."
```

Listing 2: **Mosaic Glaze Fantasy.** A close-up of vibrant ceramics with a dazzling faucet. (Colorful ceramics, shining faucet)

Prompts for the mug tree environment:

```
mug = "An oversized mug with a faded \u2018World's Best Bartender\
    u2019 logo, filled with room-temperature coffee and a tiny
    lipstick mark on its rim."
mug_tree = "A sleek, modern steel drying rack holding a neat row of
    eclectic mugs, each hanging at a slightly different angle."
background = "The counter is scattered with an array of bar staples,
    including an upturned shaker, a jar of maraschino cherries left
    open with sticky syrup pooled at the base, tiny, colorful drink
    umbrellas laying flat, and cocktail recipe cards partially
    obscured by a dishcloth. The dim overhead lighting casts a cozy
```

```
    yet neglected ambiance, reflecting off of glass surfaces,
    highlighting water stains and the subtle sheen of unpolished wood.
     Seasonal drink posters curl at the corners on the walls behind,
    and the muted hum of conversation drifts from unseen patrons."
```

Listing 3: **Afternoon Clutter.** An up-close view of a neglected bar counter amidst a busy afternoon, capturing the hustle and bustle pausing for just a moment. (bar counter)

```
mug = "A tall novelty mug with light visible scuffs, its personality
    yet vibrant, with a traveler's emblem boldly emblazoned.",
mug_tree = "Innovatively designed transparent drying rack, its
    architecture raising daily-use mugs to a dignified height",
background = "The bar counter's imperfections speak of immediate use \
    u2013 glistening droplets from pint glasses left haphazardly,
    several salt grains shining under muted sunlight, and a stray beer
     tab nestled amongst dried orange zest. A sharp, storied scratch
    veers perilously close to a vase of gaudy red carnations.
    Decorative but decisively aged beer mats pattern the visible
    stained wood, while a distant chorus of cheerful banter tingles
    the air.",
```

Listing 4: **Unexpected Afternoon Clutter.** A surprisingly busy scene of a bar counter overtaken by mid-day merriment. (bar counter)