Robot Utility Models: General Policies for Zero-Shot Deployment in New Environments

Haritheja Etukuru*[1], Norihito Naka[1], Zijin Hu[1], Seungjae Lee[1], Julian Mehu[2], Aaron Edsinger[2]
Chris Paxton[2], Soumith Chintala[3], Lerrel Pinto[1], Nur Muhammad "Mahi" Shafiullah*[1,2]

[1]New York University, [2]Hello Robot Inc., [3]Meta Inc.

Abstract: Robot models, particularly those trained with large amounts of data, have recently shown a plethora of real-world manipulation and navigation capabilities. Several independent efforts have shown that given sufficient training data in an environment, robot policies can generalize to demonstrated variations in that environment. However, needing to finetune robot models to every new environment stands in stark contrast to models in language or vision that can be deployed zero-shot for open-world problems. In this work, we present *Robot Utility Models (RUMs)*, a framework for training and deploying zero-shot robot policies that can directly generalize to new environments without any finetuning. To create RUMs efficiently, we develop new tools to quickly collect data for mobile manipulation tasks, integrate such data into a policy with multi-modal imitation learning, and deploy policies on-device on Hello Robot Stretch, a cheap commodity robot, with an external mLLM verifier for retrying. We train five such utility models for opening cabinet doors, opening drawers, picking up napkins, picking up paper bags, and reorienting fallen objects. Our system, on average, achieves **90% success rate in unseen, novel environments interacting with unseen objects**. Moreover, the utility models can also succeed in different robot and camera set-ups with no further data, training, or fine-tuning. Primary among our lessons are the importance of training data over training algorithm and policy class, guidance about data scaling, necessity for diverse yet high-quality demonstrations, and a recipe for robot introspection and retrying to improve performance on individual environments.

Keywords: Imitation Learning, Manipulation, Robotics

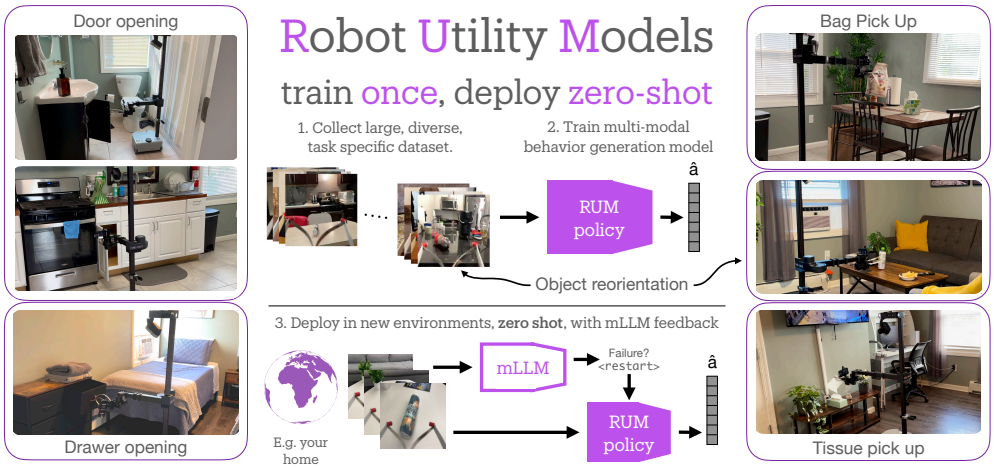


Figure 1: Robot Utility Models are trained on a diverse set of environments and objects, and then can be deployed in novel environments with novel objects without any further data or training.

*Denotes Equal Contribution

8th Conference on Robot Learning (CoRL 2024), Munich, Germany.

1 Introduction

We have seen rapid progress in training manipulation skills recently [1, 2, 3, 4, 5, 6, 7], largely brought about by fitting deep networks on data collected by teleoperating robots [8, 9, 10, 11, 12]. The mechanism for deploying such skills in new environments mimics the pretrain-then-finetune strategy first developed by the vision community circa 2014 [13]. There, models were first pretrained on ImageNet and then finetuned on task-specific data such as detection, segmentation, and pose estimation [13, 14].

In the context of robotics, this strategy involves pretraining on large robot datasets [15, 12, 16, 17] to produce a robot foundation model, which is then fine-tuned on data collected in new environments or tasks [16, 18, 7]. This need to fine-tune the foundation model for each and every new environment is limiting as it requires humans to collect data in the very environment where the robot is expected to perform. So while vision and language models have moved on to zero-shot deployments, i.e. without environment-specific finetuning data, such a capability eludes most robot manipulators. This is not to say that there have not been attempts to create zero-shot manipulation models – several foundational work in grasping and pick-and-place [19, 20, 21] have tackled this problem albeit with a task-specific solution.

Creating a general policy for an arbitrary task that works zero-shot is challenging for several reasons. First, training such a model requires a large amount of data, and collecting robot data is difficult and expensive due to the need for human demonstrations. Second, open-world datasets have diverse and multi-modal behaviors, making it hard to fit models to this data. Third, unlike standardized data in vision and language, robotics lacks uniform hardware setups, complicating real-time model deployment. Finally, zero-shot models in new environments have higher failure rates, necessitating robust error detection and recovery mechanisms.

In this work, we introduce Robot Utility Models (RUMs), a new framework for training focused and functional *utility* models to complete helpful tasks that can be deployed zero-shot without further training or fine-tuning in novel environments. This is done by taking a systems-first approach. To scale up our datasets without compromising on data quality, we develop a new tool, building on prior work in untethered data collection [16, 22]. We train policies on these diverse dataset with state-of-the-art multi-modal behavior learning algorithms [23, 24] and show how they can absorb and scale with large-scale demonstration data. Finally, we deploy the policy in multiple different environments out of the box, with self-critique via mLLMs [25] and retrying, showing how the policy can be robustly executed on cheap, general-purpose hardware. A selection of our trained models are available on the Hello Robot Stretch without much modification.

To validate RUMs, we run a total of 2,950 robot rollouts in real-world environments including homes in New York City (NY), Jersey City (NJ), and Pittsburgh (PA). These experiments reveal the following:

- We show that it is possible to create general Robot Utility Models with a moderate amount of data in the order of 1,000 demonstrations (Section 2). These RUMs achieve a 90% average success rate on zero-shot deployment in 25 novel environments (Section B.1).
- The success of RUMs relies primarily on two key techniques. First, the use of multi-modal policies (Section 2.3) provides a zero-shot success rate of 74.4% (Section B.2). Second, the mLLM based self-critique and retrying system (Section 2.4) further improves the success rate by 15.6% (Section B.6).
- While the overall framework for RUMs is straightforward, the devil is in the details, where we find gains from unexpected sources, e.g. data diversity vs. data quantity (Section B.4 and B.5).

To encourage the development of RUMs for a wider variety of tasks, we open source our code, data, models, and hardware designs. Experiment and deployment videos can be found on our website: robotutilitymodels.com.

2 Robot Utility Models

We take a full-stack approach to create Robot Utility Models. At its core, our system follows the imitation learning framework. However, to effectively scale imitation learning to the point where our trained policies are deployable zero-shot, we create new tools and techniques to improve data collection, model training, inference, and deployment.

2.1 Data collection tool

One of the primary requirements of our system is to be able to scale up diverse yet accurate demonstration data for cheap. To this end, we continue on the evolutionary path of hand-held, portable data collection tools [26, 27, 28, 16, 22] that let us quickly collect precise demonstrations. Our hand-held data collection tool is built out of an iPhone Pro and a bill of materials that adds up to $25. For a detailed build instruction and the bill of materials, we refer the reader to the supplementary materials (Appendix D.5).

We design our data collection tool around the ARKit API from the widely available and used iPhone Pro (Figure 2). Given its technical capabilities, the only digital component in our tool is this iPhone, which makes our tool particularly robust to shipping and handling. The iPhone, and therefore our tool, can collect RGB video and depth data at up to 60 Hz and high precision 6D pose and position information from the ARKit API at up to 100Hz. To capture the gripper opening information, we trained an RGB-based model that predicts the gripper aperture from images.

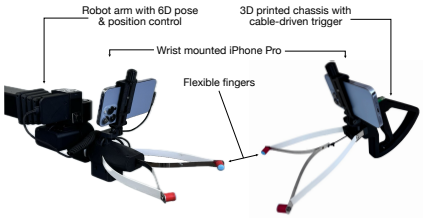


Figure 2: Our data collection tool (right) is matcehd with the end-effector on the Hello Stretch (left) for seamless transfer of policies trained on the tool's data.

2.2 Collected datasets

We collect data for each of our five tasks — drawer opening, drawer opening, reorientation, tissue pickup, and bag pickup — which we define clearly in A.2:

For each of our five RUMs, we focused on gathering approximately 1,000 demonstrations on approximately 40 environments, with about 25 demonstrations per environment on average. The only exceptions are door opening with 1,200 and drawer opening with 525 demonstrations. A small collection of such environments are shown in Figure 3. We also created an interactive data diversity visualization tool for insight on the task data we use for training: robotutilitymodels.com/data_diversity/.

2.3 Model training

Given that our data is collected by a large set of demonstration collectors, conceptually it is important for the model to handle any resultant multi-modality in the dataset. In this work, we train a large set of policy classes on our datasets for each task. Among the policy classes, the best performing ones are VQ-BeT [23] and Diffusion Policy (DP) [24]. We also train ACT [1] and MLP-BC policies on a limited set of tasks. Each policy class shares some features, such as a ResNet34-based vision encoder initialized to the HPR encoder from [16], and a transformer-based policy trunk.

2.4 Retrying with GPT-4o feedback

While a pre-trained model can solve the task in a new environment, to achieve the best possible performance, it is helpful to have additional runtime support for the model. For our deployment, we use an multimodal LLM (`gpt-4o-2024-05-13`) as an introspection module for our policies for a success detection and retrying mechanism. We define a single verification prompt for each task, and ask the mLLM to verify the success of the task given a summary of robot observations. As for the run summary, we give the mLLM every other frame from the robot camera, which is either from

the head or the wrist camera depending on the task. If the mLLM detects a failure (Figure 4), RUM automatically resets the robot to a home position and retries the task with a new initial robot state.

2.5 Deployment Details

Our primary hardware for Robot Utility Models deployment is the Hello Robot: Stretch robots with an iPhone on the wrist, but we support deploying our models on any robot arm with relative 6D pose and position control (Figure 5).

3 Capabilities of Robot Utility Models

To understand the capabilities of RUMs, we evaluate each of our models on a diverse set of environments. At the same time, we try to examine our recipe for training utility models and answer a set of questions about the trained models by running a set of ablation experiments. The primary questions that we try to answer are the following:

- How well do Robot Utility Models solve a task in an unseen environment while operating on unseen objects?
- What is the relative importance of different components of Robot Utility Models, such as training data, training algorithm, and self-verification?
  - What scale of data is needed to train capable RUMs?
  - What properties of data are most important for training RUMs?
  - How does mLLM-based self-critique affect RUMs, and where does it succeed or fail?
- How well can we deploy RUMs on new robot embodiments?

Evaluation details: We set up 25 novel environments – five for each task – with objects and props not seen in the training dataset. We take the robot to previously unseen kitchens, purchase new furniture online (door and drawer opening), and source new objects manually verified to not be in the training set (reorientation, bag and tissue pick up) — some samples can be seen in our Appendix D.2. We evaluate each system and policy for 10 trials in each of these environments, starting from the same grid of starting positions facing the task space used by [16] as we show in Appendix Figure 14. In the end, we found that across the five tasks, RUMs achieved an overall success rate of 90%.

We find that the quantity and quality of data is crucial for training a utility model, with the choice of model architecture being less critical. Second, we see that the diversity of the data collected is crucial for the model to generalize to new environments, and more important than the raw quantity of data. Third, we find that the model can be made more capable in single environments by performing self-critique on the model performance with an independent model and retrying when appropriate. Additional details of our experiments can be found in Appendix B.

4 Limitations and Conclusion

While in this work we create Robot Utility Models that can perform particular tasks zero-shot in novel environments, there are certain limitations that future versions can improve upon. The primary limitation that we see are of hardware: for example, two-fingered grippers like our data collection tool are unable to open doors with round doorknobs. Similarly, while flexible fingertips can be more lenient for the policy, it makes it hard to manipulate heavy objects. We encourage more research on better gripper and fingertip design to address these issues. Secondly, we assume navigation to be a separate component, and in this work assume that the robot is in the task space facing the task objective. Combining with modular navigation work such as [29] should address this issue. Finally, for mLLM introspection and retrying, we assume that the errors made by our model (a) leaves the task-space somewhat in-distribution, and (b) allows for an easy reset of the robot to the initial state. Increasing training data with failure recovery behavior in our dataset should let our robots recover more naturally from such failure cases.

Acknowledgments

We thank Shenglong Wang and the NYU HPC team for helping us with compute, Blaine Matulevic and Binit Shah for supporting our hardware needs, and Siddhant Haldar and Jeff Cui for providing feedback on the paper. NYU authors are supported by grants from Honda, Hyundai, NSF award 2339096 and ONR awards N00014-21-1-2758 and N00014-22-1-2773. MS is supported by the Apple Fellowship. LP is supported by the Packard Fellowship. SL is supported by the Daishin Songchon Foundation. Hello Robot authors are supported by NIH NIA R43AG072982.

References

[1] 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.

[2] Z. Fu, T. Z. Zhao, and C. Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*, 2024.

[3] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[4] S. Haldar, Z. Peng, and L. Pinto. Baku: An efficient transformer for multi-task policy learning. *arXiv preprint arXiv:2406.07539*, 2024.

[5] Z. Fu, Q. Zhao, Q. Wu, G. Wetzstein, and C. Finn. Humanplus: Humanoid shadowing and imitation from humans. *arXiv preprint arXiv:2406.10454*, 2024.

[6] T. Lin, Y. Zhang, Q. Li, H. Qi, B. Yi, S. Levine, and J. Malik. Learning visuotactile skills with two multifingered hands. *arXiv preprint arXiv:2404.16823*, 2024.

[7] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[8] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, et al. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, pages 879–893. PMLR, 2018.

[9] A. Iyer, Z. Peng, Y. Dai, I. Guzey, S. Haldar, S. Chintala, and L. Pinto. Open teach: A versatile teleoperation system for robotic manipulation. *arXiv preprint arXiv:2403.07870*, 2024.

[10] S. P. Arunachalam, S. Silwal, B. Evans, and L. Pinto. Dexterous imitation made easy: A learning-based framework for efficient dexterous manipulation. In *2023 ieee international conference on robotics and automation (icra)*, pages 5954–5961. IEEE, 2023.

[11] X. Cheng, J. Li, S. Yang, G. Yang, and X. Wang. Open-television: Teleoperation with immersive active visual feedback, 2024. URL https://arxiv.org/abs/2407.01512.

[12] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[14] G. Gkioxari, B. Hariharan, R. B. Girshick, and J. Malik. R-cnns for pose estimation and action detection. *CoRR*, abs/1406.5212, 2014. URL http://arxiv.org/abs/1406.5212.

[15] A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, A. Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

[16] N. M. M. Shafiullah, A. Rai, H. Etukuru, Y. Liu, I. Misra, S. Chintala, and L. Pinto. On bringing robots home. *arXiv preprint arXiv:2311.16098*, 2023.

[17] H. Walke, K. Black, A. Lee, M. J. Kim, M. Du, C. Zheng, T. Zhao, P. Hansen-Estruch, Q. Vuong, A. He, V. Myers, K. Fang, C. Finn, and S. Levine. Bridgedata v2: A dataset for robot learning at scale, 2023.

[18] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[19] H.-S. Fang, C. Wang, H. Fang, M. Gou, J. Liu, H. Yan, W. Liu, Y. Xie, and C. Lu. Anygrasp: Robust and efficient grasp perception in spatial and temporal domains. *IEEE Transactions on Robotics*, 2023.

[20] M. Sundermeyer, A. Mousavian, R. Triebel, and D. Fox. Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13438–13444. IEEE, 2021.

[21] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg. Dex-Net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. In *Robotics: Science and Systems (RSS)*, 2017.

[22] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *arXiv preprint arXiv:2402.10329*, 2024.

[23] S. Lee, Y. Wang, H. Etukuru, H. J. Kim, N. M. M. Shafiullah, and L. Pinto. Behavior generation with latent actions. *arXiv preprint arXiv:2403.03181*, 2024.

[24] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

[25] Y. Guo, Y.-J. Wang, L. Zha, Z. Jiang, and J. Chen. Doremi: Grounding language model by detecting and recovering from plan-execution misalignment. *arXiv preprint arXiv:2307.00329*, 2023.

[26] S. Song, A. Zeng, J. Lee, and T. Funkhouser. Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations. *IEEE Robotics and Automation Letters*, 5(3): 4978–4985, 2020.

[27] S. Young, D. Gandhi, S. Tulsiani, A. Gupta, P. Abbeel, and L. Pinto. Visual imitation made easy. *arXiv e-prints*, pages arXiv–2008, 2020.

[28] J. Pari, N. M. Shafiullah, S. P. Arunachalam, and L. Pinto. The surprising effectiveness of representation learning for visual imitation, 2021.

[29] P. Liu, Y. Orru, C. Paxton, N. M. M. Shafiullah, and L. Pinto. Ok-robot: What really matters in integrating open-knowledge models for robotics. *arXiv preprint arXiv:2401.12202*, 2024.

[30] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-maron, M. Giménez, Y. Sulsky, J. Kay, J. T. Springenberg, T. Eccles, J. Bruce, A. Razavi, A. Edwards, N. Heess, Y. Chen, R. Hadsell, O. Vinyals, M. Bordbar, and N. de Freitas. A generalist agent. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856.

[31] L. Zhao, T. Yang, Y. Yang, and P. Yu. A wearable upper limb exoskeleton for intuitive teleoperation of anthropomorphic manipulators. *Machines*, 11(4):441, 2023.

[32] Y. Ishiguro, T. Makabe, Y. Nagamatsu, Y. Kojio, K. Kojima, F. Sugai, Y. Kakiuchi, K. Okada, and M. Inaba. Bilateral humanoid teleoperation system using whole-body exoskeleton cockpit tablis. *IEEE Robotics and Automation Letters*, 5(4):6419–6426, 2020.

[33] H. Fang, H.-S. Fang, Y. Wang, J. Ren, J. Chen, R. Zhang, W. Wang, and C. Lu. Low-cost exoskeletons for learning whole-arm manipulation in the wild. *arXiv preprint arXiv:2309.14975*, 2023.

[34] P. Wu, Y. Shentu, Z. Yi, X. Lin, and P. Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators. *arXiv preprint arXiv:2309.13037*, 2023.

[35] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[36] N. E. Sian, K. Yokoi, S. Kajita, F. Kanehiro, and K. Tanie. Whole body teleoperation of a humanoid robot development of a simple master device using joysticks. *Journal of the Robotics Society of Japan*, 22(4):519–527, 2004.

[37] Z. J. Cui, Y. Wang, N. M. M. Shafiullah, and L. Pinto. From play to policy: Conditional behavior generation from uncurated robot data. *arXiv preprint arXiv:2210.10047*, 2022.

[38] S. Yang, M. Liu, Y. Qin, R. Ding, J. Li, X. Cheng, R. Yang, S. Yi, and X. Wang. Ace: A cross-platform visual-exoskeletons system for low-cost dexterous teleoperation, 2024. URL https://arxiv.org/abs/2408.11805.

[39] Y. Park and P. Agrawal. Using apple vision pro to train and control robots, 2024.

[40] S. P. Arunachalam, S. Silwal, B. Evans, and L. Pinto. Dexterous imitation made easy: A learning-based framework for efficient dexterous manipulation. *arXiv preprint arXiv:2203.13251*, 2022.

[41] S. P. Arunachalam, I. Güzey, S. Chintala, and L. Pinto. Holo-dex: Teaching dexterity with immersive mixed reality. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5962–5969. IEEE, 2023.

[42] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, pages 4171–4186, 2018. doi:10.18653/v1/N19-1423.

[43] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, volume 139, pages 8748–8763, 2021.

[44] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[45] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[46] D. Kappler, J. Bohg, and S. Schaal. Leveraging big data for grasp planning. In *ICRA*, 2015.

[47] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *JMLR*, 17(1):1334–1373, 2016.

[48] A. Depierre, E. Dellandréa, and L. Chen. Jacquard: A large scale dataset for robotic grasp detection. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3511–3516. IEEE, 2018.

[49] X. Zhu, R. Tian, C. Xu, M. Huo, W. Zhan, M. Tomizuka, and M. Ding. Fanuc manipulation: A dataset for learning-based manipulation with fanuc mate 200iD robot. https://sites.google.com/berkeley.edu/fanuc-manipulation, 2023.

[50] S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn. RoboNet: Large-scale multi-robot learning. In *Conference on Robot Learning (CoRL)*, volume 100, pages 885–897. PMLR, 2019.

[51] D. Kalashnikov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman. MT-Opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021.

[52] R. Zheng, C.-A. Cheng, H. Daumé III, F. Huang, and A. Kolobov. Prise: Llm-style sequence compression for learning temporal action abstractions in control. In *Forty-first International Conference on Machine Learning*, 2024.

[53] X. Li, M. Liu, H. Zhang, C. Yu, J. Xu, H. Wu, C. Cheang, Y. Jing, W. Zhang, H. Liu, et al. Vision-language foundation models as effective robot imitators. *arXiv preprint arXiv:2311.01378*, 2023.

[54] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta. R3m: A universal visual representation for robot manipulation. In *CoRL*, 2022.

[55] S. Karamcheti, S. Nair, A. S. Chen, T. Kollar, C. Finn, D. Sadigh, and P. Liang. Language-driven representation learning for robotics. *Robotics: Science and Systems (RSS)*, 2023.

[56] N. M. M. Shafiullah, C. Paxton, L. Pinto, S. Chintala, and A. Szlam. Clip-fields: Weakly supervised semantic fields for robotic memory. *arXiv preprint arXiv:2210.05663*, 2022.

[57] A. Gupta, M. Zhang, R. Sathua, and S. Gupta. Opening cabinets and drawers in the real world using a commodity mobile manipulator. *arXiv preprint arXiv:2402.17767*, 2024.

[58] F. Ebert, Y. Yang, K. Schmeckpeper, B. Bucher, G. Georgakis, K. Daniilidis, C. Finn, and S. Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. In *Robotics: Science and Systems (RSS) XVIII*, 2022.

[59] 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.

[60] R. Doshi, H. Walke, O. Mees, S. Dasari, and S. Levine. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. *arXiv preprint arXiv:2408.11812*, 2024.

[61] D. Fried, R. Hu, V. Cirik, A. Rohrbach, J. Andreas, L. Morency, T. Berg-Kirkpatrick, K. Saenko, D. Klein, and T. Darrell. Speaker-follower models for vision-and-language navigation. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 3318–3329, 2018.

[62] M. Shridhar, L. Manuelli, and D. Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on Robot Learning*, pages 894–906. PMLR, 2022.

[63] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. BC-Z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning (CoRL)*, pages 991–1002, 2021.

[64] Y. Du, O. Watkins, Z. Wang, C. Colas, T. Darrell, P. Abbeel, A. Gupta, and J. Andreas. Guiding pretraining in reinforcement learning with large language models. In *International Conference on Machine Learning*, pages 8657–8677. PMLR, 2023.

[65] P. Goyal, S. Niekum, and R. Mooney. Pixl2r: Guiding reinforcement learning using natural language by mapping pixels to rewards. In *Conference on Robot Learning*, pages 485–497. PMLR, 2021.

[66] A. Z. Ren, A. Dixit, A. Bodrova, S. Singh, S. Tu, N. Brown, P. Xu, L. Takayama, F. Xia, J. Varley, et al. Robots that ask for help: Uncertainty alignment for large language model planners. *arXiv preprint arXiv:2307.01928*, 2023.

[67] J. F. Mullen Jr and D. Manocha. Towards robots that know when they need help: Affordance-based uncertainty for large language model planners. *arXiv preprint arXiv:2403.13198*, 2024.

[68] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

[69] Z. Liu, A. Bahety, and S. Song. Reflect: Summarizing robot experiences for failure explanation and correction. *arXiv preprint arXiv:2306.15724*, 2023.

[70] J. Park, S. Lim, J. Lee, S. Park, M. Chang, Y. Yu, and S. Choi. Clara: classifying and disambiguating user commands for reliable interactive robotic agents. *IEEE Robotics and Automation Letters*, 2023.

[71] J. Gao, B. Sarkar, F. Xia, T. Xiao, J. Wu, B. Ichter, A. Majumdar, and D. Sadigh. Physically grounded vision-language models for robotic manipulation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12462–12469. IEEE, 2024.

[72] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[73] C. Zhang, X. Meng, D. Qi, and G. S. Chirikjian. Rail: Robot affordance imagination with large language models. *arXiv preprint arXiv:2403.19369*, 2024.

[74] C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, and Y. Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2998–3009, 2023.

[75] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, pages 9118–9147. PMLR, 2022.

[76] A. Zeng, M. Attarian, B. Ichter, K. Choromanski, A. Wong, S. Welker, F. Tombari, A. Purohit, M. Ryoo, V. Sindhwani, et al. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv preprint arXiv:2204.00598*, 2022.

[77] P. Sharma, A. Torralba, and J. Andreas. Skill induction and planning with latent language. *arXiv preprint arXiv:2110.01517*, 2021.

[78] S. Nair, E. Mitchell, K. Chen, S. Savarese, C. Finn, et al. Learning language-conditioned robot behavior from offline data and crowd-sourced annotation. In *Conference on Robot Learning*, pages 1303–1315. PMLR, 2022.

[79] Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023.

[80] M. Dalal, T. Chiruvolu, D. Chaplot, and R. Salakhutdinov. Plan-seq-learn: Language model guided rl for solving long horizon robotics tasks. *arXiv preprint arXiv:2405.01534*, 2024.

[81] H. Zhou, M. Ding, W. Peng, M. Tomizuka, L. Shao, and C. Gan. Generalizable long-horizon manipulations with large language models. *arXiv preprint arXiv:2310.02264*, 2023.

[82] V. Blukis, C. Paxton, D. Fox, A. Garg, and Y. Artzi. A persistent spatial semantic representation for high-level natural language instruction execution. In *Conference on Robot Learning*, pages 706–717. PMLR, 2022.

[83] 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. *arXiv preprint arXiv:2406.01967*, 2024.

A Method

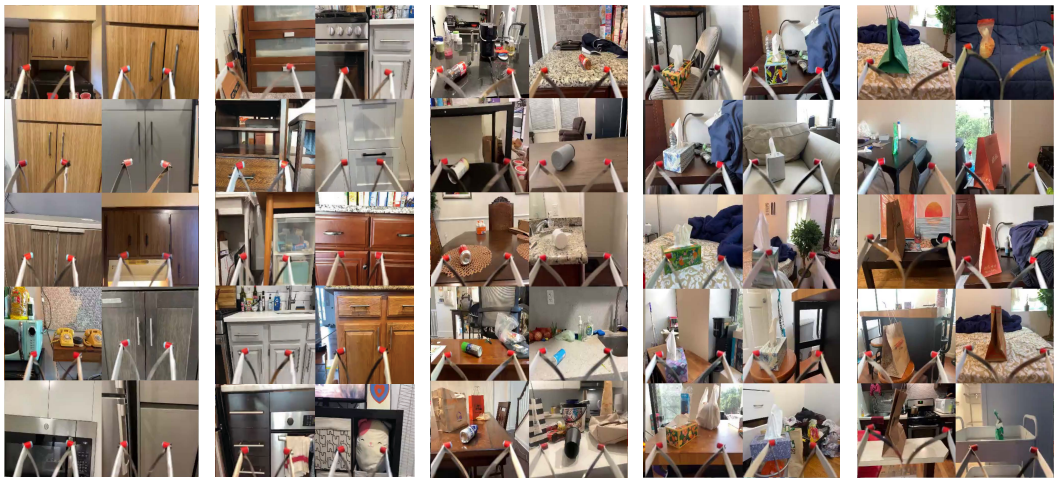
A.1 Data collection tool

Our design decisions are predicated on a few factors: portability, convenience, and set-up speed. We experimentally found these factors to be important to quickly scale up robot datasets and training RUMs. As we show with experiments in Section B.3, one of the most crucial aspect of data collection for RUMs is data diversity, i.e. collecting data from a large number of diverse environments. Thus, it is crucial to have a portable tool that is easy to mass-print, carry, and deploy in a new environment. Secondly, it is important for the collected data to be accurate across many environments with many variations. Finally, it is important to minimize the "per-environment set-up time", whether that time is spent setting up the data collection system, calibrating the camera, or the tool's SLAM system.

Our data is automatically synchronized and timestamped by the iPhone without the need for any calibration. This allows us to collect data from a wide variety of environments with no set-up time. This is in contrast to other data collection tools based on visual SLAM systems which has limited precision and are non-robust around "textureless" scenes such as close to flat walls, ceilings, or corners [22, 27]. Finally, not needing camera calibration makes our system deployable out-of-the-box in any environment, especially in the real world where the environment is not controlled. This enables us to, for example, collect data from retail home goods stores with minimal interruptions to enrich our datasets, which would be hindered if we had to calibrate the camera and odometry system for each new environment.

A.2 Collected datasets

- **Door opening:** Open doors with a long handle, on e.g. cabinets and microwaves. Due to hardware limitations, our robot cannot open doors with round knobs, so we exclude them from our dataset.
- **Drawer opening:** Open a drawer with a handle. We exclude drawers with knobs from our dataset for similar reasons as above.
- **Reorientation:** Pick up a cylindrical object (e.g. bottle) lying on a flat surface and place it upright on the same surface.
- **Tissue pickup:** Pick up a soft, flexible tissue paper from any tissue paper box.
- **Bag pickup:** Pick up a kraft paper bag or similar other bags from a flat surface.



Door opening | Drawer opening | Object reorientation | Tissue pick up | Bag pick up

Figure 3: A small sample of environment and objects from our collected dataset. We collect data for each of our five tasks on a diverse set of environments and objects using our data collection tool.

For the door opening task, we seeded this dataset with the Homes of New York dataset [16] as well as demonstrations collected during the Dobb·E experiments. For the other tasks, our dataset consists of new demonstrations collected using the data collection tool on a novel set of environments and objects. For demonstrations collected from the previous dataset by inexperienced data collectors, we do a manual quality check and exclude any environment that has a high number of low-quality demonstrations, such as failed demonstrations. Note that, to keep our experiments unbiased, we hold out test environments and objects and never collect any data on them.

A.3 Model training

We train each model for the same 500 epochs. Beyond that, we sweep to find the best hyperparameters for learning rate, history length, and chunk size, and use the recommended hyperparameters from the original papers for each model. Our final VQ-BeT models are trained on data subsampled at 3.75Hz, and uses 6 most recent frames of history to predict the next action. All of our models predict the action in relative 6D space for the robot end-effector, and absolute value in the range $[0, 1]$ for the gripper opening. We discuss the impact of choosing different training algorithms in Section B.2. Training all of our models took between 24 and 48 hours on 2 Nvidia A100 GPUs on our cluster, with proportional speed-ups by using more GPUs or using more recent GPUs like H100s.

A.4 Retrying with GPT-4o feedback

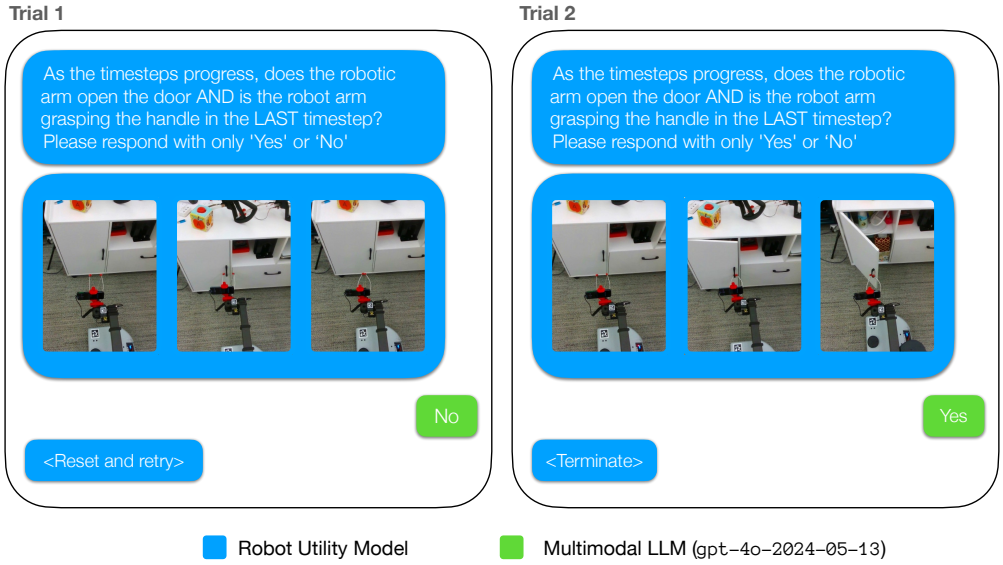


Figure 4: Automated retrying with feedback from multimodal LLM critic. We use a multimodal LLM (`gpt-4o-2024-05-13` in our experiments) to verify the success of a task given a summary of robot observations. If the mLLM detects a failure, we automatically reset the robot and retry the task with a new initial robot state until success or timeout.

A.5 Deployment Details

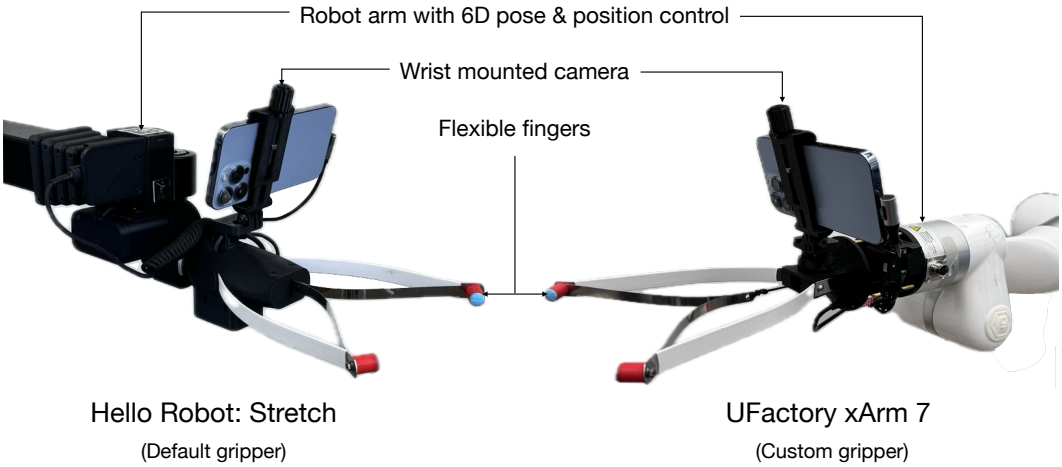


Figure 5: Picture of the some robot setups where our Robot Utility Models can be deployed. We show the Hello Robot: Stretch, and the xArm 7 robot with iPhone Pros on the wrist. Beyond these, we also deploy on Stretch robots with default D405 wrist cameras.

We design and release an associated robot end-effector that can be mounted on standard robot arms, such as the xArm or Franka Panda. Similarly, while we primarily use the iPhone Pro as the deployment camera, we also show deployment on other wrist cameras, such as the Intel Realsense D405, which is the default wrist camera on Hello Stretch Edition 3 onwards. Overall, our deployment hardware system really relies on three things: our end-effector with a flexible two-fingered gripper and gripper tips, a wrist camera with a sufficient field of view, and an arm with six degrees of freedom to mount our wrist. We release default integration code for Hello Stretch 3 and an xArm wrist mount that we created, which should serve as illustrative examples for other arms.

B Experiments

B.1 Zero-shot evaluation of RUMs on unseen environments

The most important test of capability for a Robot Utility Model is whether such a model is capable of solving the target task in a new environment operating on new objects. We test for this capability by running our RUMs on our set of 25 eval environments and objects not seen during training.

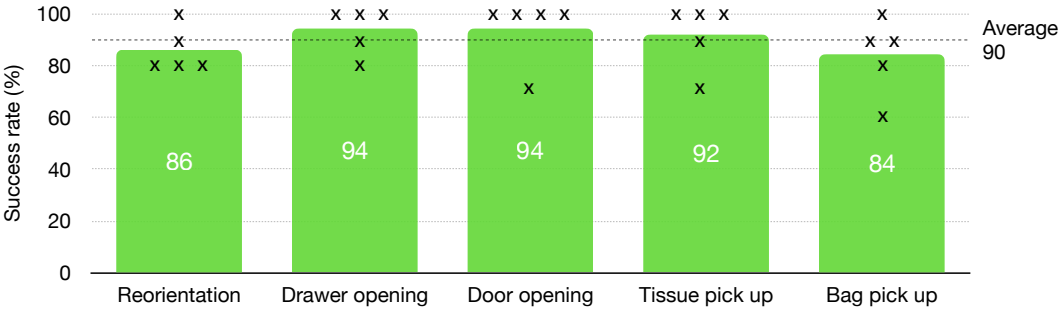


Figure 6: Success rate of Robot Utility Models on average over five novel scenes in five different tasks. The X's on the figure denote success rates from individual environments.

On Figure 6, we see that on unseen and novel environments, RUMs perform well, achieving a 90% success rate overall, and ranging between 84% to 94% on individual tasks. We discuss some of the

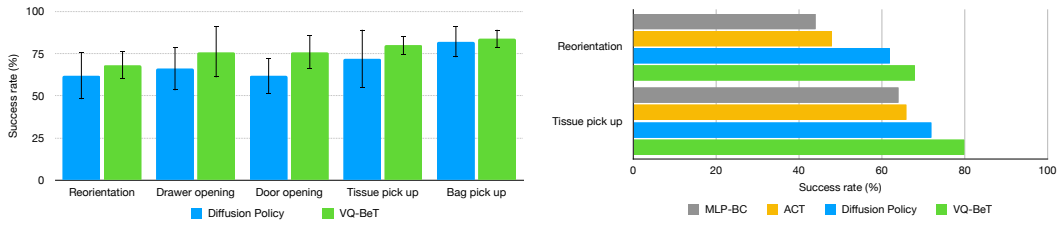


Figure 7: Left: Relative comparison of the success rate (with standard error) of different policy architectures on our dataset on all five tasks (without automated retrying). Right: Relative comparison of different policy classes on our dataset on two tasks (without automated retrying). The performance of VQ-BeT and Diffusion Policy is generally neck-to-neck, while the performance of other algorithms is not far behind, implying that the training data is much more important than training algorithm.

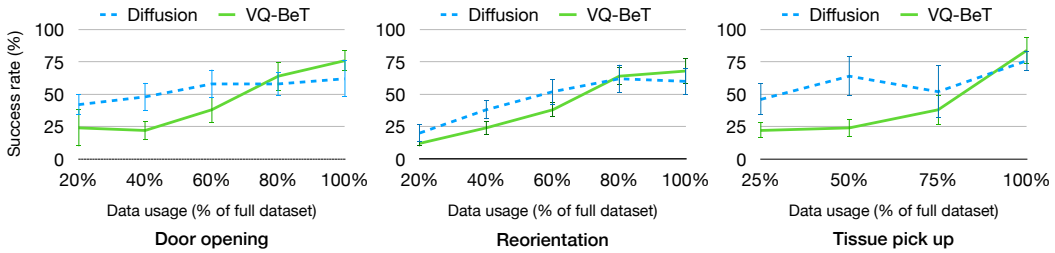


Figure 8: Understanding the performance change of RUMs as the dataset scales up on three of our tasks, with standard error on error bars. We see better performance from Diffusion Policy (DP) on smaller datasets, but as we scale up, VQ-BeT outperforms DP in 900–1,200 demonstrations limit.

failure cases we observe in the Appendix Section D.7. Additionally, we show the performance of RUMs on each test environment on Table 1, showing that across all of our evaluation experiments, RUMs achieves some success in every environment. This success implies that our policies have a general idea of solving the target task; then such policies are further boosted with post-training methods (Section B.6). On all of our following experiments, we try to understand these two factors separately: the raw performance of the underlying RUM policies, and the effect of introspection and retrying on the performance of RUMs.

B.2 Effect of policy architecture and training method on RUMs

Once we have verified that RUMs can actually solve tasks in novel environments, we investigate the relative importance of different components within the training recipe. In particular, we compare the raw performance of different policy architectures on our dataset without the introspection component. We train a set of policy classes on our datasets for each task, including VQ-BeT [23], Diffusion Policy (DP) [24], and as baselines, ACT [1] and MLP-BC on two of the tasks. We show the relative comparison of the base success rates of different policy architectures, without retrying, in Figure 7.

As we see in Figure 7, VQ-BeT and DP are the top two algorithms in terms of performance, with comparable performance in most tasks and overlapping error bars. Moreover, we see from Figure 7 that while ACT and MLP-BC are not exactly on par, they are not far behind either. This observation implies that with training data of sufficient quality, the choice of algorithm may not be a make-or-break decision, and more energy should be spent on collecting diverse and accurate data. While we have similar performances on the test environment, we use VQ-BeT over DP for our final models due the higher performance and a lower latency on the robot CPU itself during deployment.

B.3 Effect of scaling datasets on RUMs

As our experiments show the importance of training data in creating RUMs, we investigate the properties of the dataset that a successful RUMs relies on. In particular, we dig into the scale of

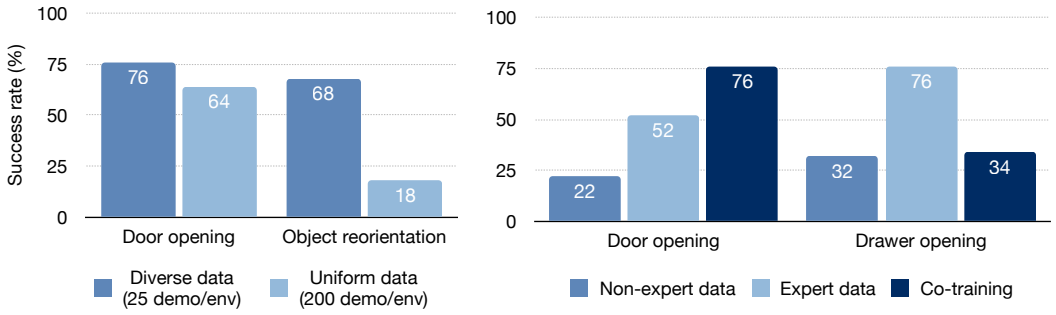


Figure 9: Understanding the importance of different qualities of data in training RUMs. On the left, we see that diverse datasets are more valuable than more uniform datasets, with strong effects on the reorientation task with many unseen environments and object. On the right, we see that usually expert data is more valuable than non-expert or play data while learning behavior on a same sized dataset. Moreover, we see that co-training with expert data and play data may sometimes reduce the policy performance, contrary to common knowledge.

dataset at which reliable generalization emerges, and how RUMs' performance vary with dataset size. We train our policies on a random subset of environments from the task-specific datasets, and evaluate them on our evaluation environments.

In Figure 8, we show the performance of VQ-BeT and Diffusion Policy without retrying trained on such data subsets on our evaluation environments as we scale up the dataset. We see that while Diffusion Policy performs better on smaller datasets, it saturates on larger datasets where VQ-BeT outperforms it. This observation implies that while a smaller dataset may be sufficient for training a capable RUMs, a larger dataset is crucial for achieving the best performance. Even on our largest datasets, we see that the performance of VQ-BeT continues to improve as the dataset scales up, implying that more data may improve RUMs even further.

B.4 Importance of data diversity in training RUMs

Beyond the scale of the dataset, we also investigate how the diversity of the training data impacts the performance of RUMs in Figure 9 (left). We create two alternate datasets of equal size for the door opening and the object reorientation tasks. The first datasets are composed of a large number of diverse environments with roughly 25 demonstrations in each environment. The second dataset is composed of fewer, between 5 and 6, distinct environments with roughly 200 demonstrations on each environment. We see that on the door opening task, where the scene diversity is narrower, both diverse and uniform environment trained policies performed well. However, in the reorientation task, with many different unseen environments and objects, only diverse-environment trained RUM policy performs well – the policy trained on more uniform environments experiences a 50% performance drop. This result implies that to train an effective RUM, collecting a diverse dataset is important.

B.5 Impact of using expert demonstrations on training policies

While scaling up the dataset size and diversity is important for training RUMs, an important question to consider is the quality of the training dataset. Namely, while it may be easy to collect a large number of demonstrations by a large number of demonstrators, the quality of the demonstrations may vary. In this section, we investigate the value of using expert demonstrations in training RUMs.

In Figure 9 (right) we compare the performance of RUMs trained on roughly 500 demonstrations, where the data is either sampled from expert or non-expert demonstration collectors. Here, "expertise" is defined as experience deploying Dobb·E policies on the robot. We see that in general, expert data is more valuable than non-expert data, with expert data outperforming non-expert data in all tasks. Moreover, we see that co-training with expert and non-expert data can sometimes, but not always, improve the performance of the policy. This observation implies depending on the task, data

quality can have different levels of suboptimality, and in extreme cases may even hurt performance in co-training, which goes against a common practice in some earlier works [1, 12].

B.6 Effects of introspection and retrying with self-critique in RUMs

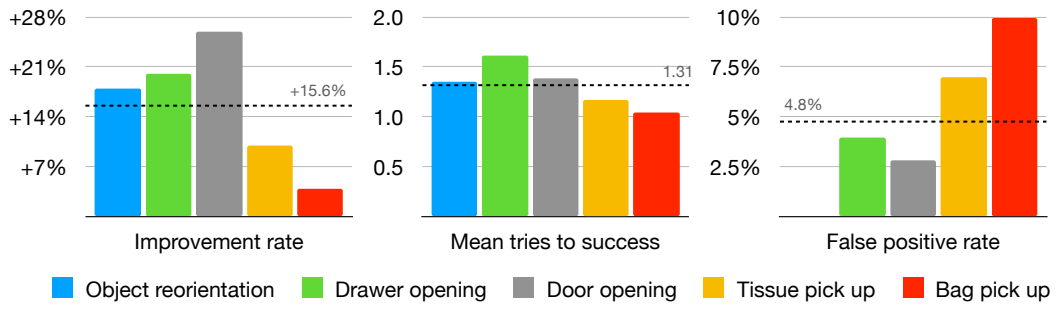


Figure 10: Understanding the details of introspection and retrying in RUMs. On the left, we see that retrying improves the performance of RUMs significantly, with an average 15.6% improvement. In the middle, we see that with retrying, most tasks get solved quite fast, on average with 1.31 tries. On the right, we see that while the mLLM is able to help, it can also have false positives (4.8% average over five tasks) which may let some errors slip past.

In RUMs, we are using a multimodal large language model (mLLM) as a self-critique method to identify failures. However, a pretrained mLLM in practice is just another layer of fail-safe for our robot deployment, and not a guarantee of success in itself. Thus, in this section we try to understand how it helps, and how such introspection method can fail.

In Figure 10 (left), we can see the improvement rate of using self-critique over simply using the RUM policies without any retrying mechanism. On average over our 5 tasks, we see a 15.6% improvement over simply using RUM policies. While retrying is crucial to a higher success rate, a system that is stuck retrying for a long time is much less useful. Thankfully, on average, when RUMs succeeds, it does so within 1.31 tries on average, as we see from Figure 10 (middle). Finally, we analyze the primary failure mode of mLLMs, which is predicting false positives: classifying a trajectory as a success when it's actually a failure. On average, 4.8% of our trajectories exhibit such behavior, constituting of half of the total errors, as seen on Figure 10 (right).

B.7 Transferring RUMs to different embodiments

Finally, we investigate the ability of RUMs to be transferred to different embodiments and cameras. We test the performance of two RUMs on the other robot setup shown in Figure 5: UFactory xArm 7, which is different from the Hello Robot Stretch setup we run other experiments on. We see that RUMs can be transferred to different embodiments and cameras with minimal loss in performance: roughly 10% drop in performance in both cases without corrective mLLM feedback, as shown in Figure 11. We expect combining RUMs with the mLLM self-critique would result in similar increase in performance in other embodiments as well; in fact, with an external third person camera, we expect to see a higher portion of the errors being caught and corrected. This experiment implies that RUMs can be easily deployed on different robots and cameras with minimal effort, making it a versatile tool for a wide range of robotic applications.

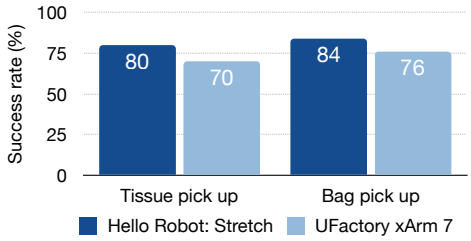


Figure 11: Performance of RUMs without corrections on different embodiments as shown in Figure 5: RUMs can transfer to different embodiments with minimal loss in performance.

C Related works

Large Scale Data Collection: The data acquisition pipeline represents one of the most critical element of a data-driven robot learning framework. Previous works has employed a diverse array of data acquisition techniques, combining many open-sourced datasets across diverse simulation or real-world data including diverse robot embodiment from many institutions across the globe [30, 15, 3, 12].

The most common approaches to robot demonstration collection involves pairing the robot or end-effector with remote controller devices or kinematically isomorphic equipment. The devices utilized have a range of complexity and forms: they encompass full robotic exoskeletons [31, 32, 33], as well as simpler data collection tools [1, 34, 2], and also methods that don't require physically moving a robot [16, 26, 28, 27, 22]. Additionally, various control methods have also been employed, including the use of video game controllers [35, 36], Virtual Reality (VR) devices [9, 37, 11, 38, 39, 40, 41, 5], and mobile phones [8].

While the most intuitive method is to physically move a real robot, it is both difficult to do and hard to scale to a diverse set of environments. The hardware controller approach can be inefficient because it requires the demonstrator to mentally map robot behavior to controller inputs. The opposite, using a device without moving the robot is efficient in that the demonstrator's movements can be mapped directly to the robot, but it is challenging to apply force feedback. Studies that provides perspective on the relative merits of these two direction are [16, 22], which combines the versatility of simple controller with the intuitiveness of moving a physical end-effector. In this work, we employ a device that inherits and improves the device proposed from [16, 22] for our data collection pipeline.

Pretrained Robot Models: Pre-trained foundation models have demonstrated a wide range of generalization performance across various domains, with the capability to learn from internet-scale pre-training data [42, 43, 44, 45]. However, in comparison to these vision and language pre-trained models, learning a foundation model for robotics has been considered a relatively challenging area, due to the limited quantity of available datasets [46, 47, 48, 49], the significant discrepancy across the domains [50, 51, 15], and the inherently challenging nature of the action datasets in terms of tokenization [23, 3, 52].

To address these issues, recent research is increasingly adopting techniques that introduce modular and hierarchical systems, incorporate pre-trained language and visual models [53, 54, 55, 56, 29, 57], and collect large scale data with efficient data collection schemes [12, 3, 17, 58, 59]. Consequently, they have enabled the pre-trained foundation robot models to exhibit enhanced generalization performance, thereby showcasing that the robotic agents are capable of operating in more than one robot embodiment and operating environment [18, 30, 7, 60]. In contrast with the aforementioned approaches, which follow a method of training on internet-scale data and fine-tuning on task-specific data, our approach does not expect that the model will have access to a dataset in the environments where the robot is expected to operate. Rather, this project demonstrates the capacity of generalizable performance without a necessity to fine-tune the model for each novel robot embodiment and environment.

Large Models Feedback and Improvement: Due to their capacity to comprehend intricate semantics and relations, Natural language and Large language models (LLM), have recently been applied to robotic agents powered by imitation learning [61, 7, 62, 63] and reinforcement learning [64, 65].

Among the wide capabilities afforded by language models, those commonly employed in the context of decision-making include providing feedback in the resolution of uncertain information [66, 67, 68, 69, 25, 70, 71], suggesting affordance of what is possible in the environments by combining with Value functions [72], and imagination of outcomes [73] or planning and decompose complex tasks into mid-level plans [74, 75, 76, 77]. Language models could also be used to improve the overall performance of autonomous agent systems by improving reward signal [78, 65, 79], leveraging their

long-horizon reasoning [80, 81, 82], or designing environments [83]. In this project, we employ the mLLM to provide feedback in the form of a reset signal in open-ended environments, a manner analogous to that of the studies above.

D Additional Details

D.1 Detailed Results from Experiments with Self-critique and Retrying

Task	Environment/Object	Success $\cdot/10$
Door Opening	Kitchen Trash Door	7
	Kitchen Cabinet Door	10
	Brown Cabinet Door	10
	Metal Cabinet Door	10
	White File Cabinet Door	10
Drawer Opening	Kitchen Drawer	10
	Cloth Drawer	9
	White File Cabinet Drawer	10
	Small File Cabinet Drawer	10
	Dresser Drawer	8
Bag Pick Up	Hollister Bag	9
	American Eagle Bag	10
	Qdoba Bag	8
	Journey's Bag	9
	Yellow Bag	6
Tissue Pick Up	White Tall Box	10
	White Short Box	10
	Black Square Box	9
	Red Square Box	10
	Kleenex Box	7
Object Reorientation	Pink Bottle	9
	White Board Cleaner	8
	Spices Container	8
	Coke Can	8
	Compressed Air	10

Table 1: Detailed success statistics of RUMs on our evaluation environments.

D.2 Evaluation Environments

Figure 12: Picture of evaluation environments for the tasks Reorientation, Drawer opening, and Door opening.

Figure 13: Pictures of the evaluation environments for the task Tissue pick up and Bag pick up.

D.3 Multimodal Large Language Model Prompts for Success Verification

Here, we present the prompt that we use to verify RUMs success with mLLMs.

Door Opening

As the timesteps progress, does the robotic arm open the door AND is the robot arm grasping the handle in the LAST timestep?
Please respond with only 'Yes' or 'No'.

Drawer Opening

As the timesteps progress, does the robotic arm grasp the drawer handle and open it AND is the drawer open in the last timestep?
Please respond with only 'Yes' or 'No'.

Reorientation

As the timesteps progress, does the robotic arm/gripper reorient the object upright AND is the object upright in the LAST frame?
Please respond with only 'Yes' or 'No'.

Tissue Pick-Up

As the timesteps progress, does the robotic arm/gripper grasp the tissue AND is the gripper grasping the tissue in the LAST timestep?
Please respond with only 'Yes' or 'No'.

Bag Pick-Up

As the timesteps progress, does the robotic arm/gripper grasp the bag AND is the gripper grasping the bag in the LAST timestep?
Please respond with only 'Yes' or 'No'.

D.4 Evaluation Schedule

In Figure 14, we show the starting position of the robot for our 10-run evaluations to understand the positional generalization capabilities of Robot Utility Models.

Figure 14: 10-run evaluation schedule used to evaluate Robot Utility Models, with robot starting positions denoted by the pale blue dots in the image. We assume that the robot is at the task space facing the object, but it can be at different offsets with respect to the target object. On our object centric tasks (reorientation, bag and tissue pickup) we also randomize the position of the object itself.

D.5 Bill of Materials

Here, we present the bill of materials for our hardware components, assuming that the interested researcher or user owns an iPhone Pro already. The total cost comes out to be slightly below $25 for the entire setup.

Item	Price	Unit Price	Qty
Reacher Grabber Tool	26.99	13.50	1
Brass Tapered Heat-Set Inserts	21.82	0.22	3
Thread-Forming Screws	7.75	0.31	3
Button Head Screw - M4 x 0.70 - 8mm	12.91	0.13	1
Button Head Screw - M4 x 0.70 - 5mm	8.64	0.09	2
Button Head Screw - M4 x 0.70 - 35mm	16.77	0.34	2
Nylon-Insert Locknut	5.57	0.06	2
Dowel Pin	16.09	0.32	3
Nylon Unthreaded Spacer	18.41	0.18	2
Kevlar Cord	20.99	20.99	1/100
Heat Shrink Tubing	10.79	10.79	1/30
Black 3D Printer Filament	25.99	25.99	3/20
Total		**21.99**	

Table 2: Main Body

Item	Price	Unit Price	Qty
Socket Head Screw - M3 x 0.5mm - 8mm	12.52	0.13	2
Steel Hex Nut - M3 x 0.5mm	2.62	0.03	2
M3 Steel Washer	2.19	0.02	2
Red 3D Printer Filament	25.99	25.99	3/1000
Oomoo 25 Silicone Rubber	33.99	33.99	1/200
Total		**0.61**	

Table 3: Gripper Tips

D.6 Deploying on Stretch's Default D405 Camera

Deploying our Robot Utility Models on the standard Hello Robot Stretch SE3 requires normalizing the image coming out of the default Intel Realsense D405 wrist camera. We created an affine transformation that maps the D405 image to the same pixel coordinates as the iPhone camera.

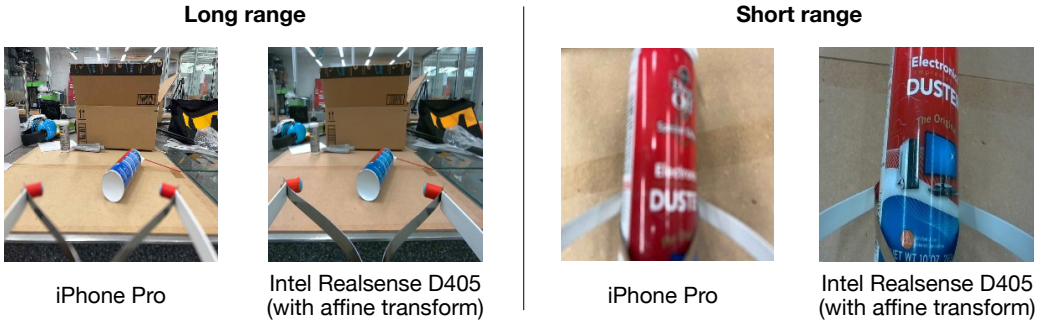


Figure 15: We can see the corresponding D405 camera image alongside the iPhone Pro image. While in the long range, the images look similar, in the short range iPhone images are out of focus because of the different focal lengths of the cameras.

As we can see from Figure 15, applying the affine transform to the D405 camera maps it to pretty similar viewpoint as the wrist mounted iPhone. While we can run RUMs directly with this camera transform, we see a performance drop which we hypothesize happens because of the especially apparent difference in close-range. This difference is caused by the different focal lengths of the two cameras, and may be solved in the future with image augmentations.

Item	Price	Unit Price	Qty
Socket Head Screw - M5 x 0.8mm - 20mm	17.10	0.17	1
Socket Head Screw - M5 x 0.8mm - 50mm	4.26	0.85	1
Steel Hex Nut - M5 x 0.8mm	5.24	0.05	2
Button Head Screw - M4 x 0.70 - 8mm	12.91	0.13	1
Black 3D Printer Filament	25.99	25.99	3/20
Total		**2.03**	

Table 4: Phone Holder

D.7 Failure Modes

Figure 16: Examples of some failures in real world rollouts. Since RUMs retries on failure with mLLM feedback, the failure modes tend to be peculiar, some examples of which are shown here.

As we mention in the main paper, with mLLM guided retries, our failures tend to be more peculiar than simply "robot failed to complete task". In this section, we try to shine some light on what kind of failures we experience in our system.

- **Reorientation:** Primary failure modes for this task are when retry becomes impossible because of environmental issues, such as the target bottle rolling away on the table, being dropped off the surface (an example of which is shown on the Figure 16), pushing it too far into the table (to a position too far for our robot arm), or being rotated sideways by the gripper before grasping. In out-of-distribution surfaces, it can be hard to estimate how large the surface is visually and thus placing the object after reorientation may miss the surface or the robot may run into the surface.
- **Drawer opening:** Beyond the most direct failure mode of missing the drawer handle, we experienced some failure modes where the model does not know when to stop pulling on cloth drawers and thus pulls out the entire drawer. Without force feedback, it can be hard to tell visually when the drawer starts sagging. Force feedback on the fingertips would help the robot correct for it.

- **Door opening:** Here, the primary failure mode we experience are on unusual doors, such as the trash cabinet door with a hole in it. There, GPT sometimes classifies the door as "open" even when it is closed. In some rare cases, when door handles are close together, the robot may grasp around both handles and fail to reset as it gets stuck when retracting.
- **Tissue pick up:** The tissue box itself being light and easy to move means that sometimes the box moves with the tissue as its being picked up. As a result, the box may get picked up with the tissue, or get pushed off from its table by the robot (Figure 16.)
- **Bag pick up:** The case of bag picking up is interesting because it has one of the highest success rates from the raw RUM policy but also sees the smallest improvement (4%) from GPT feedback. This failure from mLLM feedback happens usually because from the robot wrist or head camera, it can be hard to tell whether the bag has been picked up. As a result, GPT tends to have a high number of false positives for this task. Having a better third-person view of the workspace should help address this issue.