Continuous Autonomous Improvement for Mobile Manipulation

Hrish Leen

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2024-73
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2024/EECS-2024-73.html

May 9, 2024

Acknowledgement

The author would like to thank Laura Smith, You Liang Tan, Oieer Smith, Jennifer Zhao for helping guide the project and building the full-stack system.

Continuous Autonomous Improvement for Mobile Manipulation

Hrish Leen
Department of Electrical Engineering and Computer Sciences
University of California Berkeley United States
hrish@berkeley.edu

Abstract: Lab-trained robot policies for manipulators often suffer performance drops when deployed in real-world unstructured environments. This happens from encountering data that's out of distribution from their training data that are usually gathered in structured lab environments. To overcome this challenge and help the robot continually cope in such scenarios, we introduce **C**ontinuous **A**utonomous Improvement for **Mo**bile Manipulation, or CAMo. CAMo is a robot learning system that builds on top of existing foundation models for navigation and manipulation by collecting data directly from these real-world environments and asynchronously compiling them to a server for further fine-tuning. Through its mobile base, CAMo is able to incorporate a lot of diverse scenes and real-world perturbations in its ever-increasing data set, better enabling itself to adapt to the difficulty of being in an unstructured environment. By leveraging the multi-modal capacity and stochastic nature of the diffusion head of its manipulation policy, CAMo can reinforce good manipulation behaviors through autonomously collected rollouts for similar but unseen tasks. Along with a LIDAR sensor onboard to enact fail-safe mechanisms and human intervention data for further navigation improvement, CAMo is able to continuously improve in the real world with decreasing human involvement as time goes on.

Keywords: Autonomous Improvement, Lifelong Learning, Mobile Manipulator

1 Introduction

Lab-trained robot policies for manipulation often suffer performance issues when deployed in real-world unstructured environments. The policies being trained are usually on stationary robot setups where the arm is tethered to a table with a carefully structured environment in front of it to train in. Because of having been trained on datasets that involve images in such static settings, these manipulation policies struggle to cope with different tables or settings in the wild that are too out of distribution. In this paper, we propose **CAMo**, a full-stack distributed robot learning system aimed to tackle this issue by allowing the robot to learn from data recorded in the wild.

CAMo leverages two foundational models, Octo [1] and NoMaD [2], for manipulation and navigation respectively, and finetunes these models to perform better in real-world scenarios. CAMo's distributed architecture and hardware allow it to keep collecting data continuously for several hours from autonomous rollouts at different in-the-wild locations in a safe manner with as minimal human intervention as possible. Through this paper, we demonstrate the stages of how we prepared CAMo to start collecting useful autonomous rollouts on simple manipulation tasks, decreasing the human effort required along the way. From teleoperation in the wild to asynchronous training, CAMo's full-stack system allows the user to easily train manipulation policies in diverse unstructured real-world environments with decreasing human labor. We show that CAMo, by collecting lots of autonomous rollouts, can show increased performance to unseen tasks or similar tasks demonstrated in expert human trajectories.

6th Conference on Robot Learning (CoRL 2022), Auckland, New Zealand.

2 Related Work

2.1 Foundational Models Used

The CAMo is run using separate policies for manipulation and navigation and finetunes these models to work better in the wild. For manipulation, we use Octo [1], an open-source generalist manipulation policy trained on datasets collected by various robot labs and companies [3][4][5]. In this paper, we leverage the stochastic nature of Octo's diffusion head and multi-modal capabilities to collect diverse but successful autonomously collected trajectories to be used for further fine-tuning. For navigation, we use NoMaD [2], a diffusion policy that allows for goal-conditioned navigation. CAMo uses NoMaD to traverse different tables using pre-collected topographic maps to continue collecting manipulation data in diverse environments. Both polices are pre-trained on large diverse datasets and serves as the backbone from which CAMo finetunes off.

2.2 Lifelong learning

Like previous works in lifelong learning for robotics [6][7], the policies CAMo uses are initialized/pre-trained on a large dataset collected in diverse environments for both manipulation and navigation. Similar to human-in-the-loop training papers [8][9], a key training stage involves shared control where the human can intervene and correct the robot mid-trajectory. However, after the manipulation policy achieves a certain success threshold, this stage is superseded by learning from completely autonomous rollouts. We leverage VLMs to filter these trajectories and add to the ever-growing data buffer accordingly.

2.3 Mobile Manipulation Learning Systems

There exist many mobile manipulation papers in the literature [10][11]. On the hardware side, our model falls on the low-cost end and is optimized to last a long time without human supervision (i.e. a large battery, etc.) compared to other platforms out there. Like many of these systems, and due to the mobility of the robot, we build our own full-stack platform and distributed system to communicate observations and actions between the robot and an off-site computer. Another aspect that differentiates our paper from mobile manipulation learning papers [12] [13], is that our manipulation policy learns to utilize the full 6DOF our robot arm has to offer. While it's understandable to restrict the action space to high-level actions for the manipulator to accelerate learning as in previous papers, CAMo is built to learn complex manipulation tasks required at the places it navigates to.

3 CAMo's System Architecture

3.1 Hardware Setup

CAMo boasts being on the lower end of the range of prices compared to other mobile manipulator platforms (refer to Table 1) while maintaining competitive functionality and extremely easy to change and prototype. Equipped with an AgileX Tracer for the mobile base and a ViperX 300s with 6DOF for the manipulator platform, CAMo can lift payloads as large as 750g and can easily support hardware extensions with the Tracer's 100kg payload. CAMo has a portable ECOFlow Delta 1.2kWh battery that allows it to run for several hours. The base has a separate in-built battery that also allows it to run for several hours. For onboard compute, CAMo contains a NVIDIA Jetson AGX Orin, which is more than enough to support real-time data collection while communicating with off-site computers for other purposes. CAMo's has a shoulder camera propped with a stand and gripper camera to make data resemble the setups of systems on which Octo was trained on. CAMo contains a navigational camera at its midsection whose input is fed to NoMaD for navigation and a RPLIDAR S2 for safety purposes while being on the move. Finally, these parts are encased with commonly found aluminum extrusions that allow for rapid prototyping and durability to shield the components.

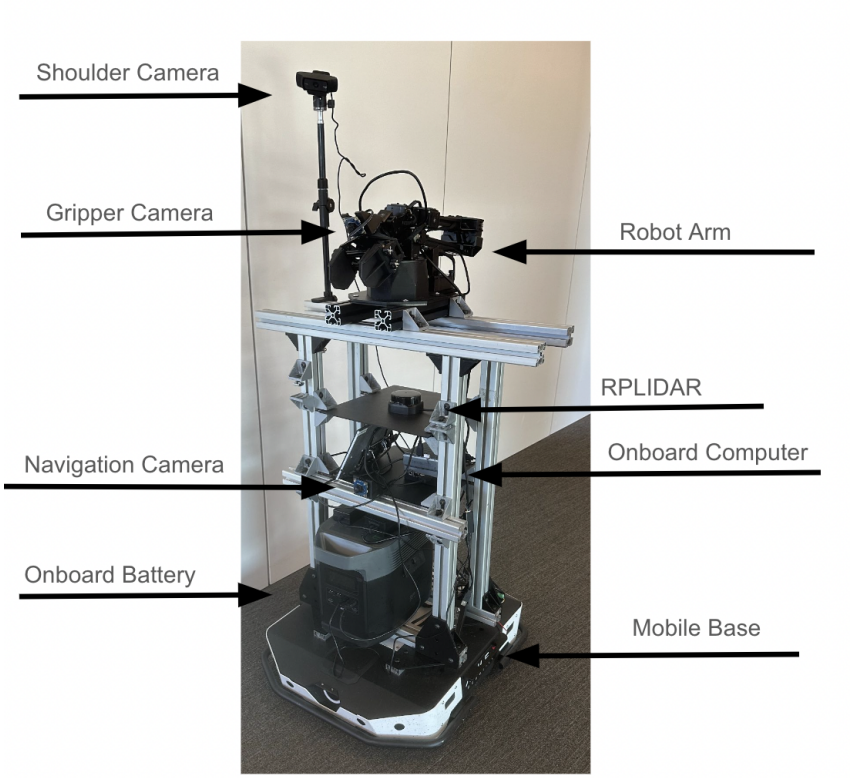

Figure 1: **Anatomy of CAMo:** Showcasing different components of the mobile manipulator system

Hardware Component	Price
AgileX Tracer	7k USD
ViperX 300s	6k USD
RPLIDAR S2	400 USD
Cameras (Fish eye USB x2, Logitech C920 x1)	100 USD
NVIDIA Jetson AGX Orin	2k USD
ECOFlow Delta Battery	900 USD
Total:	16.4k USD

Table 1: **Price Breakdown of CAMo**

3.2 Distributed Software Pipeline

Being a mobile system, it would be impractical for the robot to contain all of its compute resources onboard. Not only does the system need to store large amounts of autonomously collected data and train large foundation models, but it also needs to be capable of allowing humans to easily intervene for navigation and manipulation while being on the move. For this purpose, we've developed a distributed communication system with AgentLace that allows the compute onboard to communicate with an offsite computer to offload data, run model inference, and asynchronously train models. The software also allows for an additional connection of a laptop connected with a VR headset and controller so that a human can intervene in the robot's manipulation and navigation systems while it's running.

As illustrated in Figure 2, the compute onboard CAMo acts purely as a server to execute commands received by an off-site computer on the manipulation and navigation hardware. This off-site computer regularly stores data collected and sent by the onboard compute or the human intervener and consolidates it to create a bigger dataset that will be used for further fine-tuning. The off-site PC is

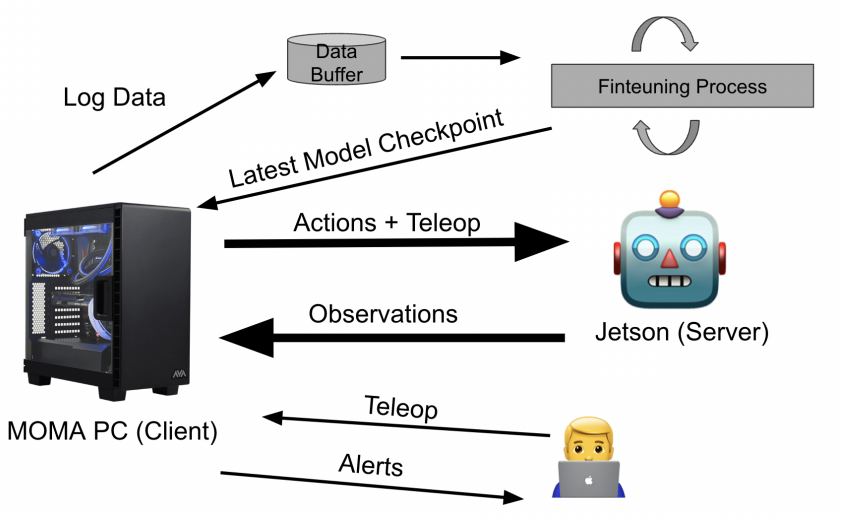

Figure 2: **Diagram of Distributed Communication**

asynchronously kick-starting training scripts for these large foundation models on more powerful TPUs. While the off-site PC is capable of also running training scripts, TPU training reduces wait times by 4 or 5 hours allowing improvements to be seen in the system relatively quickly. The off-site PC, also responsible for running model inference, switches to the newer models as they become available.

The separation of responsibilities provided by this distributed system drastically reduces the maintenance of the mobile robot itself, instead allowing the user to debug any system bugs from a full developer setup on the off-site PC. For ease of development, we've learned it's best to keep the responsibilities of the robot's compute to be minimal.

3.3 Training Methodology

CAMo uses two foundation models for manipulation and navigation. For manipulation, we use Octo, an open-source generalist robot policy pre-trained on 800k trajectories collected from different labs. For navigation, we use NoMaD, a diffusion policy pre-trained on many navigation datasets available online. Out of the box, taking only a fish eye camera in front of the robot as input, NoMaD is robot controller agnostic and can navigate the Agile X tracer without additional fine-tuning. However, Octo, being able to take in different camera viewpoints and requiring a dataset collected on the robot arm to unnormalize actions, requires an initial dataset of expertly collected trajectories to be finetuned for controlling the ViperX arm with the camera setup CAMo has. When fine-tuning these policies, NoMaD is given just the images of the front-facing camera and odometry of the base whereas Octo is given images from the gripper camera, shoulder camera and language text of the task used for conditioning. Although Octo can take proprioception of the arm, we find that providing this information during training causes causal confusion and performance drops.

After initially collecting a dataset of expert trajectories, we move on to the second stage where a human intervener can further fine-tune the policies while the robot is in the wild. While the base is navigating between goal nodes, a human can assume control of the base at any time with a controller. The onboard compute is notified of the switch in command and immediately starts logging the intervention data. Similarly, CAMo supports shared control for manipulation. While the robot is performing rollouts, a human can intervene using the VR headset and complete the manipulation task. Unlike navigation, all manipulation rollouts are recorded, including the ones where a human has intervened. During this stage, the navigation policy is able to improve avoiding obstacles and humans and making sharp turns even though the base being controlled is much wider than those present in its

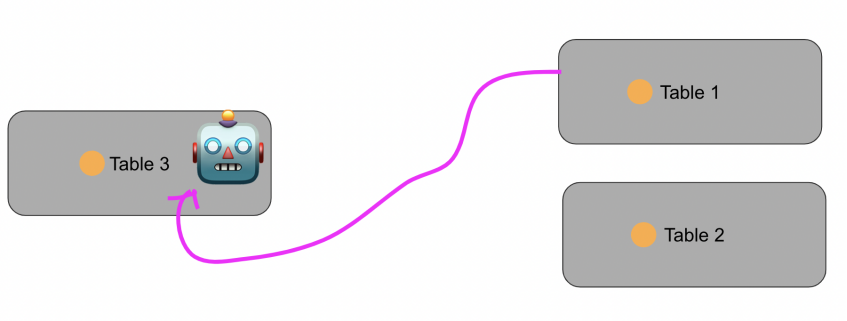

Figure 3: **Random Task Sampling Experiment Setup**

pre-trained datasets. The manipulation policy also improves in its success rates on the tasks its tasked with.

After the first two stages of training, when the manipulation policy shows reasonable success rates at the given tasks, constant human intervention is no longer needed and CAMo is able to continue to improve its manipulation policy with autonomous rollouts. Navigation is also seen to require fewer interventions and resorts to failsafe mechanisms and alerting supervising humans if completely stuck. To prevent the collection of suboptimal data, we use GPT4 to query if a trajectory has been successful or not given the task. Wv3e query GPT4 whether an object has been successfully picked up or not after a grasp.

Note: Although we use this autonomous improvement approach initially, we aim to use reinforcement learning or other methods as a way of improving autonomously for the official release

3.4 Safety

Safety is a crucial aspect of any mobile manipulation system to prevent catastrophic failure and harm to humans. This is especially true for the environments we deploy CAMo in where humans are going about their daily tasks and CAMo has to navigate amidst them. To this end, we employ guardrails and fail-safe mechanisms to avoid complications and allow the system to run continuously with minimal human supervision and intervention.

Although NoMaD does a good job of avoiding obstacles and humans while its navigating, the second stage of training with human interventions aims to improve these abilities by adding human intervention data to the dataset buffer. For manipulation, we enforce bounding boxes to where the end effector of the arm can be at any given moment during manipulation rollouts at a destination and retract the arm while it's navigating between destinations. An RPLIDAR is onboard CAMo to avoid collisions while navigating and in the event the robot is stuck or not moving for an extended period of time, an alert is sent to supervising humans.

4 Preliminary Experiments

4.1 Experiment Setup

To have minimal human intervention while ensuring diverse data collection, we run CAMo in the wild while it performs randomly sampled tasks. We deploy CAMo in the common room on the floor that our lab is in, where CAMo will encounter many people while running. Initially, we collected topographic maps of paths between 3 tables in the common room (${}^3P_2 = 6$ paths). As seen in Figure 3, from whichever table CAMo is currently at, we randomly sample a destination table and a manipulation task. We feed the corresponding topographic path to CAMo's goal image-conditioned navigation policy and then deploy CAMo's manipulation policy at the table conditioned on the randomly sampled task text.

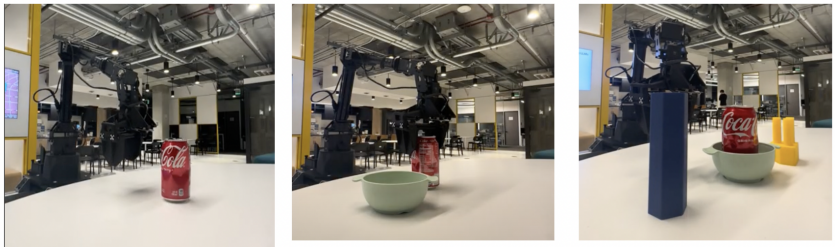

Figure 4: **Examples of the three different scenarios for soda can pickup**

Note: Experiments shown here are preliminary results and will be replaced with other experiments involving the random sampling experiment above for official release. The experiments below are used to ascertain good model parameters for finetuning octo on the task of soda can pickup. We then perform experiments to check if its possible to improve task percentage on autonomous rollouts

The tables below showcase success rates of soda can pickup for three different scenarios: no distractions, light distractions (1 or 2 objects other than the soda can), and heavy distractions (3 objects other than the soda can). Refer to Figure 4 for examples of these scenarios. Distractions are distributed at random between each trial.

4.2 Behavior Cloning for Soda Can Pickup

Table 2 compares different BC fine-tuning methods for the Octo model. As we can see, the best performance comes with expert trajectories and fine-tuning the entire model. Note that the expert trajectories are co-trained along with the datasets Octo is pre-trained with.

Table 2: Octo BC experiments

Policy	No Distraction	Light Distraction	Heavy Distraction
Octo No Expert Demos - Diffusion Head	0/10	0/10	0/10
Octo Expert Demos - MSE Head	**10/10**	7/10	3/10
Octo Expert Demos - Diffusion Head	**10/10**	**10/10**	**5/10**
Octo Expert Demos - Finetuning Diffusion Head Only	5/10	3/10	0/10

4.3 Learning from Autonomous Rollouts

The below policies are trained with autonomous rollouts collected by deploying the manipulation policy and using GPT4 to classify whether a soda can has been successfully grasped or not. Since the autorollouts involved data that changed the rotation, pitch and yaw of the end effector while the expert trajectories didn't, the policy trained with only 33 autorollouts demonstrated extreme behaviors on those axes. By collecting more autorollouts we were able to homogenize the datasets and a policy with 77 rollouts performed the best, even on the heavy distraction task which was unseen in the expert demos.

Table 3: BC with autonomous rollouts

Policy	No Distraction	Light Distraction	Heavy Distraction
Octo No Expert Demos	0/10	0/10	0/10
Octo Expert Demos and 33 Auto Rollouts - Diffusion Head	5/10	0/10	0/10
Octo Expert Demos and 77 Auto Rollouts - Diffusion Head	**10/10**	**10/10**	**8/10**
Octo Expert Demos and 77 Auto Rollouts - MSE Head	**10/10**	**10/10**	2/10

5 Conclusion, Limitations and Future Works

We present CAMo, a low-cost mobile manipulator system that continuously improves its manipulation policy autonomously. By deploying and letting it run directly in the wild, we show that CAMo can learn to not only navigate amongst humans but also improve upon tasks it hasn't seen before in expertly collected demos from autonomously collected rollouts.

Many limitations of CAMo currently provide avenues for future research. CAMo separates the navigation and manipulation policies in order to bootstrap from models that exist, however, this limits its capabilities to accomplish tasks that require coordination from both navigation and manipulation. By offloading inference to an off-site computer due to onboard computational limitations, the frequency of actions that CAMo can perform is limited. Moreover, CAMo requires a lot of human-teleoperated expert demos initially to achieve reasonable success from a novel robotic platform.

Future research can investigate improving end-to-end policies that control both navigation and manipulation. Incorporating reinforcement learning to learn mobile manipulation skills from scratch in a safe manner is also a big open problem that needs to be addressed.

References

[1] O. Mees, D. Ghosh, K. Pertsch, K. Black, H. R. Walke, S. Dasari, J. Hejna, T. Kreiman, C. Xu, J. Luo, et al. Octo: An open-source generalist robot policy. In *First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024*, 2024.

[2] A. Sridhar, D. Shah, C. Glossop, and S. Levine. Nomad: Goal masked diffusion policies for navigation and exploration, 2023.

[3] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. Ryoo, G. Salazar, P. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-1: Robotics transformer for real-world control at scale, 2023.

[4] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023.

[5] O. X.-E. Collaboration, A. O'Neill, A. Rehman, 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. 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. Stulp, G. Zhou, G. S. Sukhatme, G. Salhotra, G. Yan, G. Feng, G. Schiavi, G. Berseth, G. Kahn, G. Wang, H. Su, H.-S. Fang, H. Shi, H. Bao, H. B. Amor, H. I. Christensen, H. Furuta, 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. 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, 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. 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. Vanhoucke, W. Zhan, W. Zhou, W. Burgard, X. Chen, X. Wang, X. Zhu, X. Geng, X. Liu, X. Liangwei, X. Li, Y. Lu, Y. J. Ma, Y. Kim, Y. Chebotar, Y. Zhou, Y. Zhu, Y. Wu, Y. Xu, Y. Wang, Y. Bisk, 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, and Z. Lin. Open X-Embodiment: Robotic learning datasets and RT-X models. https://arxiv.org/abs/2310.08864, 2023.

[6] K. Bousmalis, G. Vezzani, D. Rao, C. Devin, A. X. Lee, M. Bauza, T. Davchev, Y. Zhou, A. Gupta, A. Raju, A. Laurens, C. Fantacci, V. Dalibard, M. Zambelli, M. Martins, R. Pevceviciute, M. Blokzijl, M. Denil, N. Batchelor, T. Lampe, E. Parisotto, K. Żołna, S. Reed, S. G. Colmenarejo, J. Scholz, A. Abdolmaleki, O. Groth, J.-B. Regli, O. Sushkov, T. Rothörl, J. E. Chen, Y. Aytar, D. Barker, J. Ortiz, M. Riedmiller, J. T. Springenberg, R. Hadsell, F. Nori, and N. Heess. Robocat: A self-improving generalist agent for robotic manipulation, 2023.

[7] A. Herzog, K. Rao, K. Hausman, Y. Lu, P. Wohlhart, M. Yan, J. Lin, M. G. Arenas, T. Xiao, D. Kappler, et al. Deep rl at scale: Sorting waste in office buildings with a fleet of mobile manipulators. *arXiv preprint arXiv:2305.03270*, 2023.

[8] H. Liu, S. Nasiriany, L. Zhang, Z. Bao, and Y. Zhu. Robot learning on the job: Human-in-the-loop autonomy and learning during deployment. In *Robotics: Science and Systems (RSS)*, 2023.

[9] H. Liu, S. Dass, R. Martín-Martín, and Y. Zhu. Model-based runtime monitoring with interactive imitation learning. *arXiv preprint arXiv:2310.17552*, 2023.

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

[11] J. Wong, A. Tung, A. Kurenkov, A. Mandlekar, L. Fei-Fei, S. Savarese, and R. Martín-Martín. Error-aware imitation learning from teleoperation data for mobile manipulation, 2021.

[12] C. Sun, J. Orbik, C. Devin, B. Yang, A. Gupta, G. Berseth, and S. Levine. Fully autonomous real-world reinforcement learning with applications to mobile manipulation, 2021.

[13] H. Xiong, R. Mendonca, K. Shaw, and D. Pathak. Adaptive mobile manipulation for articulated objects in the open world. *arXiv preprint arXiv:2401.14403*, 2024.

[14] F. Liu, K. Fang, P. Abbeel, and S. Levine. Moka: Open-vocabulary robotic manipulation through mark-based visual prompting, 2024.