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]

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

robotutilitymodels.com

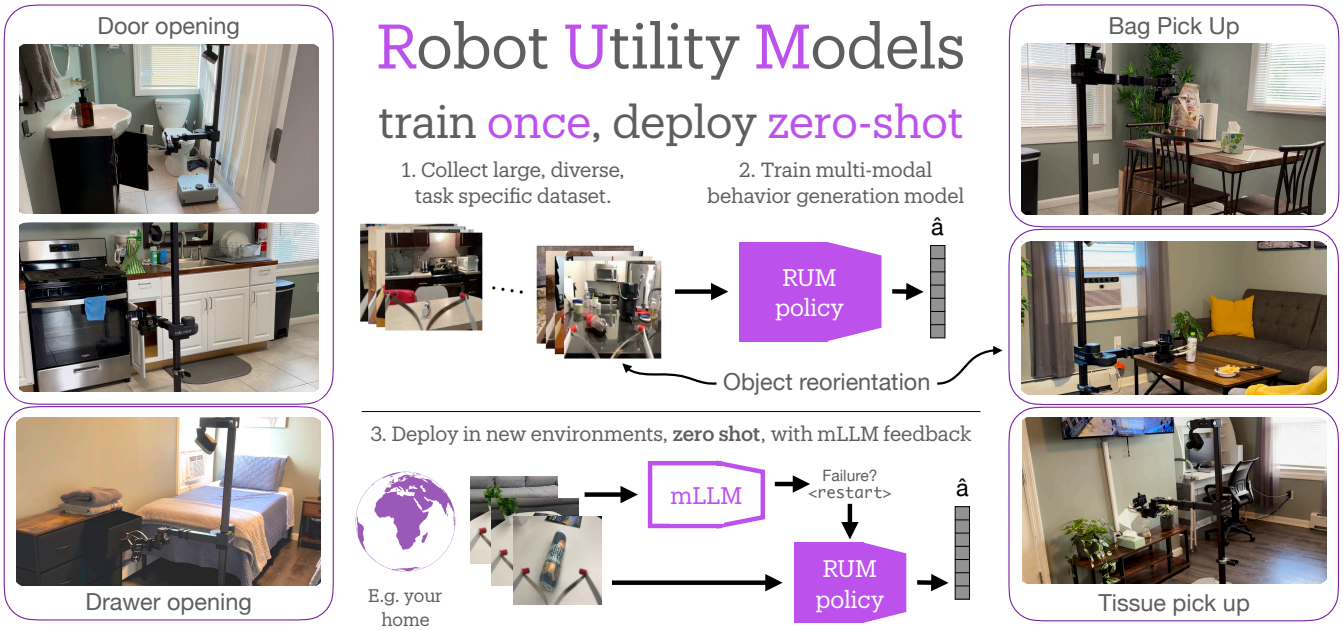

Fig. 1: Robot Utility Models are trained on a diverse set of environments and objects, and then can be deployed in novel environments with novel objects without any further data or training.

***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 the 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. 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.**

* Denotes Equal Contribution

I. 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 environ-

ment 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 datasets 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 policies 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.

Creating and deploying RUMs led us to several key lessons. First, we find that data quantity and quality are crucial for training a utility model, while model architecture is less critical. Second, we see that data diversity is essential for generalization to new environments, outweighing raw data quantity. Third, we find that single-environment performance improves by using an independent model for self-critique and retrying when appropriate.

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 II). These RUMs achieve a 90% average success rate on zero-shot deployment in 25 novel environments (Section III-A).
- The success of RUMs relies primarily on two key techniques. First, the use of multi-modal policies (Section II-C) provides a zero-shot success rate of 74.4% (Section III-B). Second, the mLLM-based self-critique and retrying system (Section II-D) further improves the success rate by 15.6% (Section III-F).
- 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 III-D and III-E).

To encourage the development of RUMs for a wider variety of tasks, our code, data, models, hardware designs, as well as experiment and deployment videos are open-sourced and can be found on our website: robotutilitymodels.com.

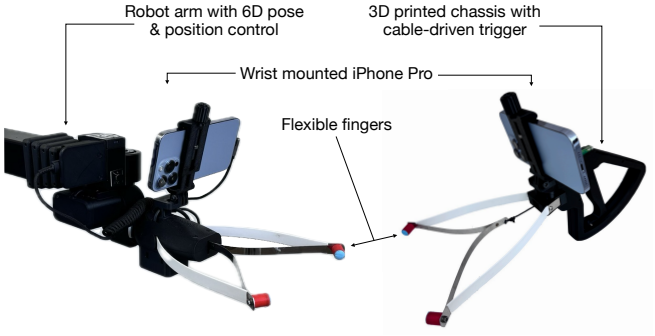

Fig. 2: Stick-v2, our data collection tool (right), is built out of an iPhone Pro and a bill of materials that adds up to $25. The tool is portable, robust, and makes it easy to start collecting data in a new environment in seconds. We match the end-effector on the Stretch robot (left) for seamless transfer of policies trained on Stick data.

II. 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.

A. 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. Following our previous work [16], we call this tool *Stick-v2*, which is a hand-held data collection tool built out of an iPhone Pro and has a bill of materials that adds up to $25. We combine inspirations from the quick deployability of Stick-v1, and the compact, handheld form factor of UMI gripper [22].

Our design prioritizes portability, convenience, and quick setup, which we found essential for scaling robot datasets and training RUMs. As shown in Section III-C, data diversity—collecting data across many environments—is crucial. Thus, a portable, mass-producible tool enables fast deployment. Additionally, it is important to minimize "per-environment setup time," whether it be for data collection, camera calibration, or the tool's SLAM system.

For the above reason, we design our data collection tool, Stick-v2, around the ARKit API from the widely available and used iPhone Pro (Figure 2). The iPhone can collect RGB video and depth data at up to 60 Hz and high precision 6D pose data from the ARKit API at up to 100Hz. This pose data gives us the absolute translate and rotation of the iPhone in space, with respect to some world frame. To capture the gripper opening information, we trained a simple RGB-based model that predicts the gripper aperture (in the range $[0, 1]$) from images. The model predicts x, y positions of the finger tips, then computes the normalized distance. We trained this model on a hand-labeled $\sim$200 image dataset of our gripper grasping various objects in front of a green screen, with the background randomized during training. The RGB, depth, and pose data is automatically synchronized and timestamped by the iPhone without the need for any calibration, further minimizing set-up time. This is in contrast to other data collection tools based on visual SLAM systems which have 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 unseen environments, especially in the real world where the environment is not controlled.

B. Collected datasets

We collect data for each of our five tasks, which are as defined below:

- **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 set it upright.
- **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.

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 sample of demonstrations from these environments can be found on our website: robotutilitymodels.com/#dataset. 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 Stick-v2 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. To gain quick insight on different task data we use for training, we created an interactive data diversity visualization tool: robotutilitymodels.com/data_diversity/.

C. 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. We also 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 use a sequence of 6 frames 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 III-B. Training all of our models took between 24 and 48 hours on 2 Nvidia A100 GPUs, with proportional speed-ups by using more GPUs or using more recent GPUs like H100s.

D. Retrying with GPT-4o feedback

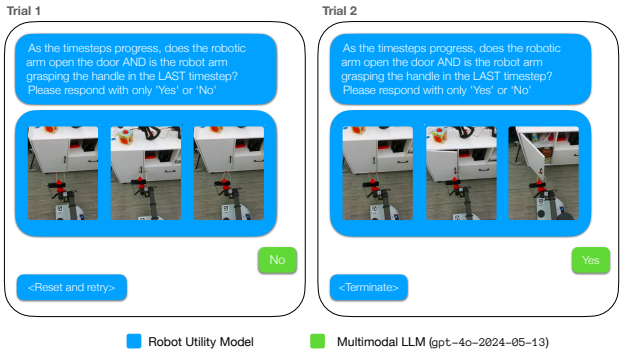

Fig. 3: Automated retrying with feedback from a multimodal LLM critic. We use a multimodal LLM (`gpt-4o-2024-05-13`) to verify the success of a task given a summary of robot observations. If the mLLM detects a failure, we automatically reset the robot and retry the task with a new initial robot state until success or timeout.

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 a 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

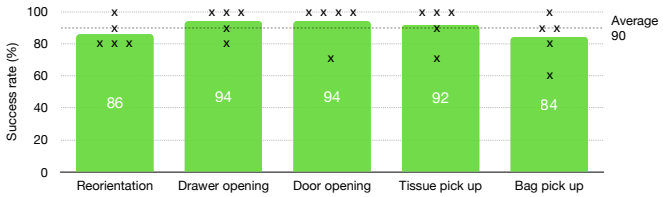

Fig. 4: Success rate of Robot Utility Models on average over five novel scenes in five different tasks, with automated retrying in each trial. The X's on the figure denote success rates from individual environments.

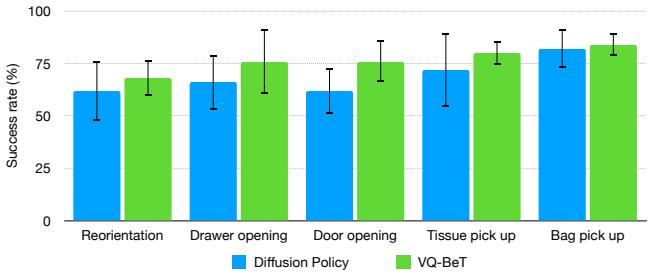

Fig. 5: Relative comparison of the success rate (with standard error) of different policy architectures on our dataset on all five tasks with one-try evaluation (without automated retrying). The performance of VQ-BeT and Diffusion Policy is generally close, with VQ-BeT narrowly outperforming Diffusion Policy.

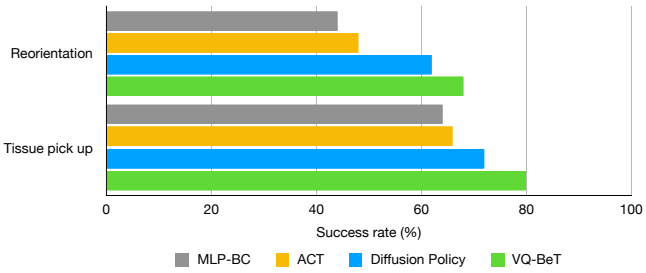

Fig. 6: Relative comparison of different policy architectures on our dataset on two tasks with one-try evaluation (without automated retrying). The performances of VQ-BeT and Diffusion Policy are close, while the performance of other algorithms is not far behind.

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. At the end of a fixed time horizon, if the mLLM detects a failure (Figure 3), RUM automatically resets the robot to a random home position within a 5cm radius of the original, and retries the task with the new initial robot state.

E. Deployment Details

Our hardware for Robot Utility Models deployment is the Hello Robot: Stretch robot with an iPhone on the wrist (Figure 2). We use the iPhone Pro as the deployment camera. We run lightweight policies (VQ-BeT, ACT, and MLP-BC) directly on the robot's Intel NUC, and Diffusion Policy through a GPU workstation. One step of our VQ-BeT policy runs with ∼15ms inference time on an A4000 GPU and ∼115ms inference time on the NUC CPU, while we find Diffusion Policy takes ∼540ms on GPU and ∼4300ms on CPU, hence the decision to run it on an external workstation.

III. 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?

a) Evaluation details: For each task, we set up 25 novel environments – five for each task – with objects and props not seen in the training dataset. To create these evaluation environments, 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). 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]. For the retrying-based experiments, while RUMs take 1.31 tries in average to succeed (Section III-F), we set a 10-try timeout to avoid getting stuck in infinite retry loops.

A. 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 test environments and objects not seen during training.

On Figure 4, we see that on unseen and novel environments, RUMs perform well, as, with automated retrying, it achieves a 90% success rate overall, and ranges between 84% to 94% on individual tasks. We see that in every environment we evaluate on, RUMs is able to achieve some success. 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 III-F). 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. Effect of policy architecture and training on RUMs

Once we have verified that RUMs can solve tasks in novel environments, we analyze the importance of different training components. Specifically, we compare the performance of

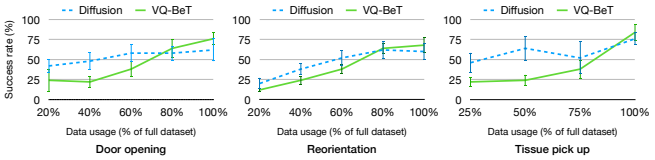

Fig. 7: Performance change of RUMs as we scale the datasets up on three tasks, evaluated 'one-try' (without automated retrying), with standard error shown on error bars. We observe better performance from Diffusion Policy (DP) on smaller datasets, but as we scale up, VQ-BeT outperforms DP in the 900–1,200 demonstrations limit.

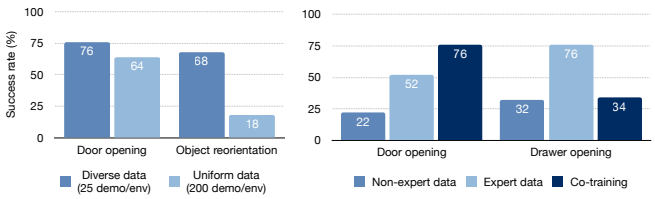

Fig. 8: Understanding the importance of different qualities of data in training RUMs. On the left, we see that diverse datasets are more valuable than more uniform datasets, with strong effects on the reorientation task with many unseen environments and objects. On the right, we see that expert data is more valuable than non-expert data while learning behavior on an identically sized dataset. Moreover, we see that co-training with both types of data may sometimes reduce performance, contrary to common knowledge, especially when the task demands multiple degrees of motion.

various policy architectures on our dataset without introspection. We train multiple policy classes per task, including VQ-BeT [23], Diffusion Policy (DP)[24], and baselines ACT[1] and MLP-BC on two tasks. Figure 5 and 6 show the base success rates of different architectures without retrying.

As we see in Figure 5, 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 6 that while ACT and MLP-BC are not exactly on par, they are not too far behind. As we see the training algorithm is not necessarily a make-or-break decision, we recommend more energy 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.

C. Effect of scaling datasets on RUMs

As our experiments show the importance of training data in creating RUMs, we investigate the dataset properties that successful RUMs rely on. In particular, the dataset scale 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 7, 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 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.

D. 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 8 (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 much more diverse environments and objects at test-time, only the 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.

E. Impact of 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 8 (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 depending on task complexity, and in extreme cases may even hurt performance in co-training, which goes against a common practice in some earlier works [1, 12].

F. Effects of introspection and retrying with self-critique

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 this introspection method can fail.

In Figure 9 (left), we can see the improvement rate of using self-critique over simply using the RUM policies

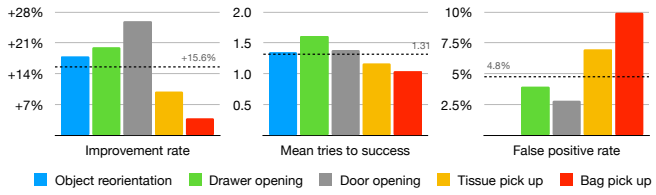

Fig. 9: The role of introspection and retrying in RUMs: on the left, we show that retrying improves the performance of RUMs significantly, with an average improvement of 15.6% over 'single-try'. In the middle, we show that with retrying, most tasks get solved with 1.31 tries on average. On the right, we see that while the mLLM verifier helps, it may also have false positives (4.8% average over five tasks) which may let some errors slip past.

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 perpetually retrying is much less useful. Thankfully, when RUMs succeed, they do so within 1.31 tries on average, as we see from Figure 9 (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 9 (right). However, we posit that this source of error will reduce with stronger verifiers in the form of more powerful language and vision models.

IV. Related Works

a) Large Scale Data Collection: The data acquisition pipeline is key in data-driven robot learning frameworks. Previous work has used a variety of techniques, combining open-source datasets from diverse simulation and real-world environments from many robot embodiments [29, 15, 3, 12].

Common approaches to robot demonstration collection involve pairing robots with remote controllers or similar devices. These range from full robotic exoskeletons [30, 31, 32] to simpler tools [1, 33, 2], and methods that don't require physical robot movement [16, 26, 28, 27, 22]. Other control methods include gaming controllers [34, 35], VR devices [9, 36, 11, 37, 38, 39, 40, 5], and mobile phones [8].

Physically moving a robot is intuitive but hard to scale, while controller-based methods require mental mapping of inputs to robot behavior. Non-physical approaches, though efficient, lack force feedback. Studies such as [16, 22] compare these methods. In this work, we improve upon the device proposed in [16, 22] for our data collection pipeline.

b) 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 [41, 42, 43, 44]. However, compared to these vision and language pre-trained models, learning a robotics foundation model has been more challenging due to the limited size of available datasets [45, 46, 47, 48], the significant discrepancy across the domains [49, 50, 15], and the inherent challenges of representing diverse robot actions [23, 3, 51].

Recent research addresses these challenges by adopting modular and hierarchical systems, incorporating pre-trained language and visual models [52, 53, 54, 55, 56, 57], and using efficient large-scale data collection methods [12, 3, 17, 58, 59]. These techniques enhance generalization in pre-trained robot models, enabling them to operate across different robot embodiments and environments [18, 29, 7, 60]. Whereas these approaches rely on fine-tuning with task-specific data, our project achieves generalizable performance without fine-tuning for each new robot or environment.

c) Large Models Feedback and Improvement: Due to their capacity to comprehend intricate semantics and relations, LLMs have been used for run-time monitoring [61, 62, 63], and have been applied to robotic agents powered by imitation learning [64, 7, 65, 66] and reinforcement learning [67, 68]. 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 [69, 70, 71, 72, 25, 73, 74], suggesting affordances of what is possible in the environment through Value functions [75], and imagination of outcomes [76] or planning and decompose complex tasks into mid-level plans [77, 78, 79, 80]. Language models could also be used to improve the overall performance of autonomous agent systems by improving reward signal [81, 68, 82], leveraging their long-horizon reasoning [83, 84, 85], or designing environments [86]. In this project, we employ an mLLM to provide feedback in the form of a reset signal, a manner analogous to that of the studies above.

V. Limitations and Conclusion

While in this work we create Robot Utility Models that can perform particular tasks zero-shot in novel environments, future versions can improve upon certain limitations. Primary among those are that of hardware: for example, two-fingered grippers like our Stick-v2 cannot open doors with round doorknobs. Similarly, while flexible fingertips can be more lenient for the policy, they make manipulating heavy objects hard. 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 [56] should address this issue. Finally, for 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.

VI. Funding Acknowledgements

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*. PMLR, 2018, pp. 879–893.

[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)*. IEEE, 2023, pp. 5954–5961.

[11] X. Cheng, J. Li, S. Yang, G. Yang, and X. Wang, "Open-television: Teleoperation with immersive active visual feedback," 2024. [Online]. Available: 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*, 2014, pp. 580–587.

[14] G. Gkioxari, B. Hariharan, R. B. Girshick, and J. Malik, "R-cnns for pose estimation and action detection," *CoRR*, vol. abs/1406.5212, 2014. [Online]. Available: 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)*. IEEE, 2021, pp. 13 438–13 444.

[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*, vol. 5, no. 3, pp. 4978–4985, 2020.

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

[30] L. Zhao, T. Yang, Y. Yang, and P. Yu, "A wearable upper limb exoskeleton for intuitive teleoperation of anthropomorphic manipulators," *Machines*, vol. 11, no. 4, p. 441, 2023.

[31] 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*, vol. 5, no. 4, pp. 6419–6426, 2020.

[32] 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.

[33] 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.

[34] 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*, vol. 36, 2024.

[35] 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*, vol. 22, no. 4, pp. 519–527, 2004.

[36] 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.

[37] 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. [Online]. Available: https://arxiv.org/abs/2408.11805

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

[39] 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.

[40] 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)*. IEEE, 2023, pp. 5962–5969.

[41] 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*, pp. 4171–4186, 2018.

[42] 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)*, vol. 139, 2021, pp. 8748–8763.

[43] 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.

[44] 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*, 2023, pp. 4015–4026.

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

[46] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *JMLR*, vol. 17, no. 1, pp. 1334–1373, 2016.

[47] 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)*. IEEE, 2018, pp. 3511–3516.

[48] 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.

[49] 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)*, vol. 100. PMLR, 2019, pp. 885–897.

[50] 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.

[51] 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.

[52] 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.

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

[54] 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.

[55] 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.

[56] 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.

[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] Y. Du, K. Konyushkova, M. Denil, A. Raju, J. Landon, F. Hill, N. de Freitas, and S. Cabi, "Vision-language models as success detectors," in *Proceedings of The 2nd Conference on Lifelong Learning Agents*. PMLR, 2023, pp. 120–136.

[62] R. Sinha, A. Elhafsi, C. Agia, M. Foutter, E. Schmerling, and M. Pavone, "Real-time anomaly detection and reactive planning with large language models," in *Robotics: Science and Systems*, 2024.

[63] S. S. Raman, V. Cohen, I. Idrees, E. Rosen, R. Mooney, S. Tellex, and D. Paulius, "Cape: Corrective actions from precondition errors using large language models," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 14 070–14 077.

[64] 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*, 2018, pp. 3318–3329.

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

[66] 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)*, 2021, pp. 991–1002.

[67] 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*. PMLR, 2023, pp. 8657–8677.

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

[69] 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.

[70] 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.

[71] 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.

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

[73] 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.

[74] 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)*. IEEE, 2024, pp. 12 462–12 469.

[75] 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.

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

[77] 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*, 2023, pp. 2998–3009.

[78] 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*. PMLR, 2022, pp. 9118–9147.

[79] 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.

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

[81] 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*. PMLR, 2022, pp. 1303–1315.

[82] 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.

[83] 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.

[84] 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.

[85] 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*. PMLR, 2022, pp. 706–717.

[86] 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.