GRAPPA: Generalizing and Adapting Robot Policies via Online Agentic Guidance

Author Names Omitted for Anonymous Review.

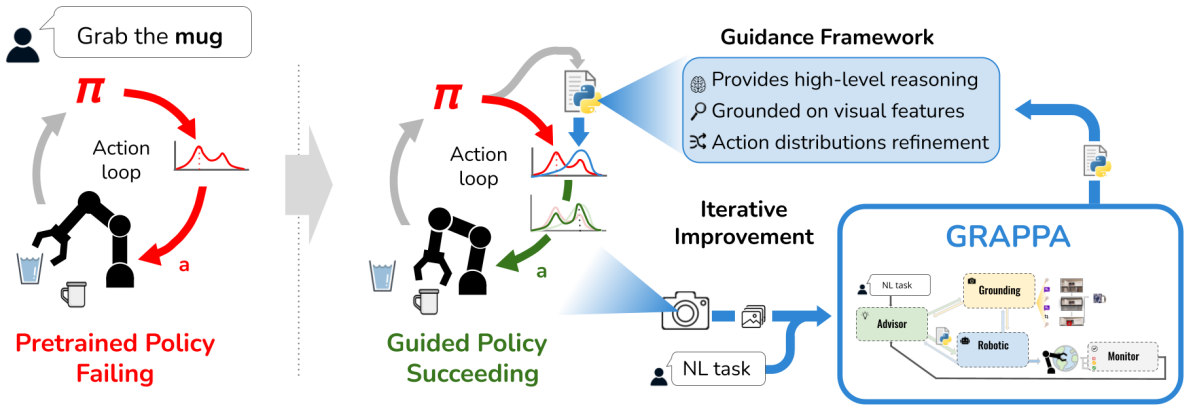

Fig. 1: An illustration of how GRAPPA intervenes in the action loop of pre-trained robotic policies in failure cases to provide visuomotor guidance generated with an agentic framework of agents to shift the action distribution for correct execution.

***Abstract*— Robot learning approaches such as behavior cloning and reinforcement learning have shown great promise in synthesizing robot skills from human demonstrations in specific environments. However, these approaches often require task-specific demonstrations or designing complex simulation environments, which limits the development of generalizable and robust policies for unseen real-world settings. Recent advances in the use of foundation models for robotics (e.g., LLMs, VLMs) have shown great potential in enabling systems to understand the semantics in the world from large-scale internet data. However, it remains an open challenge to use this knowledge to enable robotic systems to understand the underlying dynamics of the world, to generalize policies across different tasks, and to adapt policies to new environments. To alleviate these limitations, we propose an agentic framework for robot self-guidance and self-improvement, which consists of a set of role-specialized conversational agents, such as a high-level advisor, a grounding agent, a monitoring agent, and a robotic agent. Our framework iteratively grounds a base robot policy to relevant objects in the environment and uses visuomotor cues to shift the action distribution of the policy to more desirable states, online, while remaining agnostic to the subjective configuration of a given robot hardware platform. We demonstrate that our approach can effectively guide manipulation policies to achieve significantly higher success rates, both in simulation and in real-world experiments, without the need for additional human demonstrations or extensive exploration. Code and videos are available at: `https://project-grappa.github.io/`**

I. Introduction

In recent years, the emergence of foundation models, including pre-trained large language models (LLMs) and vision language models (VLMs), has enabled impressive capabilities in understanding context, scenes, and dynamics in the world. Furthermore, emergent capabilities such as in-context learning have shown great potential in the transfer of knowledge between domains, e.g., via few-shot demonstrations or zero-shot inference. However, the application of these models to robotics is still limited, given the intrinsic complexity and scarcity of human-robot interactions and the lack of large-scale datasets of human-annotated data or demonstrations [1], [2].

Approaches that leverage LLMs and VLMs alongside robotics systems often fall into one of two categories. The first category is that of using foundation models as zero-shot planners, code-generators, and task-descriptors—all of which attempt to use the foundation model to provide some high-level instruction to a low-level policy or to describe the policy's actions in natural language [3], [4], [5], [6] or generated plans as code [7], [8]. While these works have illustrated impressive reasoning capabilities, they still require either learning additional mappings to interface the generated natural language instructions with the low-level policy or, in the case of the code generation examples, they rely on pre-existing handcrafted primitives to compose. The secondary category fine-tunes foundation models for robot learning, using large corpora of robotic demonstration datasets to supervise part or all of the policy to learn to perform tasks in an end-to-end manner, [9], [10], [11]. While effective in-domain, methods often struggle to generalize to novel object categories, tasks, and environments unseen during training, and may suffer from negative transfer when trained on very diverse collections of robot data [12], [13].

In this paper, we extend the deployability of robot policies by using foundation models to generate low-level visuomotor guidance to handle out-of-distribution scenarios, such as sim-to-real differences or new tasks and robot platform variations (Figure 4). We design an agent-based framework,

where a team of conversational agents works together to refine the action distribution of a robot's base policy, via grounded visuomotor guidance (Figure 1). As illustrated in Figure 3, and upon a request from the advisor agent, the grounding agent can look for objects through a combination of detection, tracking, and high-level reasoning, to broaden or restrict the search as the context demands. Once the target object has been visually located, the Monitor and Advisor agents generate a guidance function that outputs a biasing guidance distribution, which, when combined with the action distribution of the Robotic agent (policy), ensures that the policy can complete task-relevant behaviors until it succeeds. In this way, the agentic framework bridges high-level reasoning with low-level control, enabling systems to reason about failure and self-guide.

In simulated and real-world experiments, we comprehensively and empirically demonstrate that our framework for **G**ene**r**alizing and **A**da**p**ting Robot **P**olicies via Online **A**gentic Guidance (GRAPPA) provides robustness for a variety of representative base policy classes to sim-to-real transfer paradigms and out-of-distribution settings, such as unseen objects.

In summary, our paper provides the following contributions:

- We propose GRAPPA, an agentic robot policy framework that self-improves by generating a guidance function to update base policies' action distributions, online. Our framework is capable of learning skills from scratch after deployment and generalizes across various base policy classes, tasks, and environments.
- For robustness against cluttered and unseen environments, we propose a grounding agent that performs multi-granular object search, which enables flexible visuomotor grounding.
- We provide experimental results showcasing GRAPPA as a helper tool for aiding in the sim-to-real transfer of policies without the need for extensive data collection.

II. Related Work

Vision-Language Models for Robot Learning: Several works explore the notion of leveraging pre-trained or fine-tuned Large Language Models (LLMs) and/or Vision-Language Models (VLMs) for high-level reasoning and planning in robotics tasks [1], [5], [7], [14], [8], [15], [16], [17], [18], [19], [20], [21]—typically decomposing high-level task specification into a series of smaller steps or action primitives, using system prompts or in-context examples to enable powerful chain-of-thought reasoning techniques. This strategy of encouraging models to reason in a step-wise manner before outputting a final answer has led to significant performance improvements across several tasks and benchmarks [1]. Despite these promising achievements, these approaches rely on handcrafted primitives [5], [14], [7], [18], struggle with low-level control, or require large datasets for retraining. Furthermore, various approaches that leverage VLMs for robot learning suffer from a granularity problem when using off-the-shelf models in a single-step/zero-shot manner [22] or are unable to perform failure correction without costly human intervention [14], [18], [23], [22]. In contrast, our framework bridges high-level reasoning with low-level control, by leveraging an agentic framework for online modification of a base policy's action distribution at test-time, without requiring human feedback or datasets for fine-tuning. Moreover, we mitigate the granularity problem by proposing a flexible and recursive grounding mechanism that uses VLMs to query open-vocabulary perception models.

Agent-based VLM frameworks in Robotics: Rather than using single VLMs in an end-to-end fashion, which might incur issues in generalization and robustness, various works have sought to orchestrate multiple VLM-based agents to work together in an interconnection multi-agent framework. Here, multiple agents can converse and collaborate to perform tasks, yielding improvements for the overall framework in online adaptability, cross-task generalization, and self-supervision [24], [25], [26]. These *agentic* frameworks have provided possibilities for enabling the identification of issues in task execution, providing feedback about possible improvements. Challenges remain, however, in that this feedback is often not sufficiently grounded on the spatial, visual, and dynamical properties of embodied interaction to be useful for policy adaptation; instead, the generated feedback is often too high-level or provides merely binary signals of success or failure.

Self-guided Robot Failure Recovery: [27] offer an analysis of frameworks for leveraging VLMs as behavior critics. Some approaches have explored integrating such pre-trained models to improve the performance of reinforcement learning (RL) algorithms. For instance, [19] use LLMs in a zero-shot fashion to design and improve reward functions, however this approach relies on human feedback to generate progressively human-aligned reward functions and further requires simulated retraining via RL, with high sample-complexity. On the other hand, [28] avoid the need for explicit human feedback by directly using a VLM (CLIP) to compute the rewards to measure the proximity of a state (image) to a goal (text description), enabling gains in sample-efficiency for guiding a humanoid robot to perform various maneuvers in the MuJoCo simulator. A limitation of this approach, however, lies in the difficulty of generating rewards for long-horizon or multi-step tasks, which are characteristic of tasks involving complex agent-object interactions. [6] present a framework for detecting and analyzing failed executions automatically. However, their system focuses on explaining failure causes and proposing suggestions for remediation, as opposed to also performing policy correction. In this paper, we propose a framework that directly adapts a base policy's action distribution, during deployment, without requiring additional human feedback.

III. Grounding Robot Policies with Guidance

A. Problem Formulation

We consider a pre-trained stochastic policy $\pi : O \times S \rightarrow A$ that maps observations o_t and robotic states s_t to action distributions $a_{\pi,t}$, at each time step t. Our objective is to

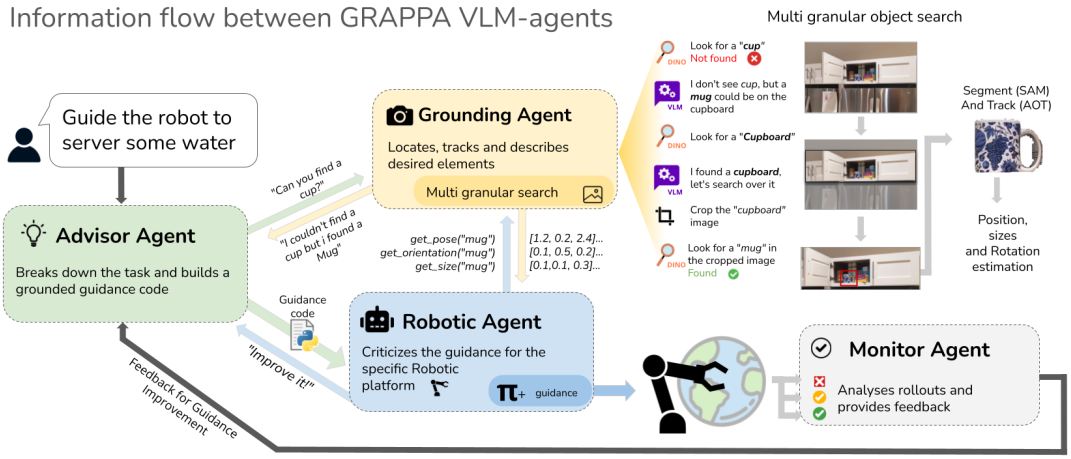

Fig. 3: Information flow between the agents to produce a guidance code. **a)** The advisor agent orchestrates guidance code generation by collaborating with other agents and using their feedback to refine the generated code. **b)** The grounding agent uses segmentation and classification models to locate objects of interest provided by the advisor, reporting findings back to the advisor. **c)** The robotic agent uses a Python interpreter to test the code for the specific robotic platform and judge the adequacy of the code. **d)** The monitor agent analyses the sequence of frames corresponding to the rollout of the guidance and give feedback on potential improvements.

generate a guidance distribution g_t that, when combined with this base policy, enhances overall performance during inference without requiring additional human demonstrations or extensive exploration procedures. Specifically, we aim to develop a modified policy $\pi_{\text{guided}} : O \times S \to A$ that achieves better performance on tasks where the original policy π struggles. We define this new policy as follows:

$$\pi_{guided}(a_{g,t}|o_t, s_t) = \pi(a_{\pi,t}|o_t, s_t) * G(a_{\pi,t}|o_t, s'_{t+1}), \quad (1)$$

where $G : A \times O \times S \to [0, 1]$ is a guidance function that maps observation o_t, action a_t, possible future state s'_{t+1} into a guidance score g_t. The '$*$' operator here denotes the operation of combining both distributions conceptually, which we explore in detail in Section III-D. For the scope of this project, we assume that a dynamics model $\mathcal{D} : S \times A \to S$ is available, which can forecast possible future states of the robot $s'_{t+1} = \mathcal{D}(s_t, a_{\pi,t})$ given the current state s_t and action $a_{\pi,t}$.

Focusing on leveraging the world knowledge of Vision Language Models, while avoiding adding latency to the action loop, we choose to express these guidance functions as Python code. By integrating these code snippets into action loop of the base policies, we eliminate the need of time-consuming queries to large reasoning models.

B. A Multi-agent Guidance framework for Self Improvement

In order to generate the guidance function G, we leverage a group of conversational agents empowered with visual grounding capabilities and tool usage. Illustrated in Figure 3, the framework is composed of four main agents: an Advisor Agent, the Grounding Agent, the Monitor Agent, and the Robotic Agent.

Advisor Agent. A Vision Language Model is responsible for breaking down the task and communicating with the other agents to generate a sound guidance function for a given task.

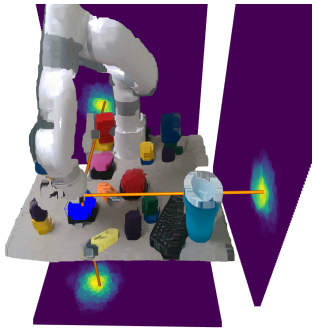

Fig. 4: Heatmap visualization of the guidance distribution, generated online by our proposed agentic framework. The distribution expresses the relevance of each possible future state for completing the task (e.g. "press the blue button"). The guidance is then used to bias the robot policy's action distribution towards the desirable behavior.

Grounding Agent. A Vision Language Model that iteratively queries the free-form text segmentation models to locate [29], [30], track [31], and describe elements relevant to the task execution.

Monitor Agent. Responsible for identifying the causes of the failures in the unsuccessful rollouts, the Monitor Agent consists of a Vision Language model equipped with a key frame extractor.

Robotic Agent. Language Model equipped with descriptions of the robot platform, a robot's dynamics model and wrapper functions for integration with the base policy. It criticizes the provided guidance functions to reinforce its relevance to the task and alignment with the robot's capabilities.

C. Guidance Procedure

The conversational agents interact with each other through natural language and query their underlying tools to iteratively produce a guidance code tailored to the task at hand, the environment, and the robot's capabilities. The information flow between these agents is depicted by Figure3.

For a given task expressed in natural language and an image of the initial state of the environment, the *Advisor*

Agent uses Chain-of-Tought [32] strategy to generate a high-level plan of the steps necessary to accomplish the task. Being able to query a *Grounding Agent* and the *Robotic agent*, the Advisor is able to collect relevant information about trackable objects and elements in the environment, as well as the capabilities and limitations of the robotic platform.

Given a plan and list of relevant objects required for the task completion, the Grounding Agent uses Grounding Dino [30] and the Segment Anything Model (SAM) [29] to locate the elements across multiple granularities and levels of abstraction. If an object (e.g., "cup") is not found, it searches for semantically similar items (e.g., "mug") or parent objects (e.g., "shelf") to refine detection. Found objects are tracked via DeAOT [33], enabling a flexible segment-and-track pipeline [29]. Object statuses inform the Advisor Agent for plan revision or guidance generation.

The Robotic agent acts as a critic to improve the guidance function generated. Equipped with a Python interpreter and details of the base policy's action space, the agent can evaluate the guidance function in terms of feasibility and relevance to the robot's capabilities. Once a function suffices the system's requirements, it is saved to be used in the action loop, in combination with the dynamics model, to provide a guidance score for possible actions sampled from the base policy.

After the execution of a rollout and the identification of failure in the task completion, the Monitor Agent is triggered to analyze the causes of the failure. By extracting key frames from the rollout video using PCA [34] and K-means clustering, the agent can feed a relevant and diverse set of images to the Visual Language Model prompted to access the failure causes. In the iterative applications of our framework, the Monitor Agent provides this feedback to the Advisor Agent, which can use this information to refine the guidance functions generated in the previous iterations.

Temporal-aware Guidance Functions. Inspired by recurrent architectures, we instruct the agents to generate guidance function conditioned on a customizable hidden state (h_t) expressed as an optional dictionary parameter as shown in the following example:

```
# Guidance function example in the context of
    grabbing a mug
def guidance_code(state,
    hidden_state={"mug_reached": False,"mug_grabbed
        ":False}):
    #available grounding functions
    #x,y,z = get_pose("mug")
    #h,w,d = get_size("mug")
    #rx,ry,rz = get_orientation("mug")
    ...
    return score, new_hidden_state
```

The idea of using abstraction in a hidden state has proven to significantly improve the guidance performance, allowing the guidance functions to keep track of the task progress and adapt the guidance to longer horizon tasks. The guided policy can thus be written as:

$$\pi_{guided}(a_{g,t}|o_t, s_t, h_t) = \pi(a_{\pi,t}|o_t, s_t) * G(a_{\pi,t}|o_t, s'_{t+1}, h_t). \quad (2)$$

The complete guidance procedure is summarized in Algorithm 1. Note that we refer to the self-orchestrated conversation between the agents which yields the guidance code as the function `Generate_Guidance_Function`.

D. Guidance and policy integration

Aiming to guide a wide range of policies, our framework is designed to work both with continuous and discrete action spaces. In this section, we discuss the operation of combining the guidance function with the base policy's action distributions. Furthermore, we discuss how deterministic regression models can be adapted to work with our framework.

Action-space Adaptation. We assume the availability of a dynamics model $\mathcal{D}$ that can forecast possible future states of the robot given a possible action $a'_{\pi,t}$. In the manipulation domain, a dynamics model is often available in the form of a forward kinematics model, a learned dynamics model, or a simulator. Oftentimes, the action space A of policies them-self is the same as the robot's state S either being or joint angles of the robot or the gripper's end-effector pose. For the last cases, where both the action and state space are expressed in $SE(3)$ integrating the guidance function with a base policy would only require a multiplication of the guidance scores with the action probabilities of the base policy. In other scenarios, adapting the robot's action and state space to match the representation of the visual cues (position, orientation, and size) would be required.

Algorithm 1 Guidance procedure of `GRAPPA`

Input
π: Base policy
$\mathcal{D}$: Dynamics Model
env: Environment

for each episode **do**
  $o_t,\ s_t \leftarrow$ env.init ▷ *observation and initial state*
  $G \leftarrow$ Generate_Guidance_Function($o_t,\ s_t$)
  $h_t \leftarrow$ Get_Hidden_State($G(o_t, s_t)$)
  for each time step t **do**
    $\mathcal{A}_{\pi,t} \leftarrow \{\pi(o_t, s_t)^i\}_{i=0}^{n}$ ▷ *Sample n actions*
    $\pi_t \leftarrow \pi(\mathcal{A}_{\pi,t}|o_t, s_t)$ ▷ *Get action probabilities*
    $\mathcal{S}_{\pi,t} \leftarrow \mathcal{D}(s_t, \mathcal{A}_{\pi,t})$ ▷ *Infer possible future states*
    $G_{\pi,t} \leftarrow G(o_t, \mathcal{S}_{\pi,t}, h_t)$ ▷ *Compute the guidance for the sampled possible future states*
    Normalize $G_{\pi,t}$
    $\pi_{\text{guided},t} \leftarrow \pi_t * G_{\pi,t}$ ▷ *Combine distributions*
    $a_t \leftarrow \mathcal{A}_{\pi,t}\,[\text{argmax}(\pi_{\text{guided},t})]$ ▷ *Select the best a*
    $o_t,\ s_t \leftarrow$ env.step(a_t) ▷ *Execute a_t, update state s_{t+1} and observation o_{t+1}*
    $h_t \leftarrow$ Get_Hidden_State($G(o_t, s_t, h_t)$) ▷ *Update*

Considering the visual grounding, the action space and the state space share the same representation ($SE(3)$), the operation to combine the guidance function with the base policy

TABLE I: Performance improvement on the RL-Bench [35] benchmark, by applying 5 iterations of guidance improvement over unsuccessful rollouts.

Model	turn tap	open drawer	sweep to dust-pan of size	meat off grill	slide block to color target	push buttons	reach and drag	close jar	put item in drawer	stack blocks	**Avg. Success**
Act3D 25 demos/task	76	76	96	64	92	84	96	48	60	0	69.2
+1% guidance	80 (+4)	96 (+20)	96	84 (+20)	92	84	100 (+4)	84 (+36)	80 (+20)	8 (+8)	**80.4 (+11.2)**
+10% guidance	88 (+12)	100 (+24)	96	88 (+24)	92	84	100 (+4)	60 (+12)	80 (+20)	0	78.8 (+9.6)
Act3D 10 demos/task	32	60	84	16	60	72	68	32	44	8	47.6
+1% guidance	44 (+12)	88 (+28)	88 (+4)	24 (+8)	68 (+8)	72	76 (+8)	52 (+20)	60 (+16)	8	**58 (+10.4)**
+10% guidance	44 (+12)	64 (+4)	84	20 (+4)	68 (+8)	76 (+4)	76 (+8)	40 (+8)	56 (+12)	8	53.6 (+6)
Act3D 5 demos/task	24	0	84	4	8	32	8	8	12	0	18
+1% guidance	48 (+24)	16 (+16)	84	8 (+4)	12 (+4)	40 (+8)	24 (+16)	20 (+12)	20 (+8)	0	**27.2 (+9.2)**
+10% guidance	24	0	84	8 (+4)	12 (+4)	44 (+12)	20 (+12)	8	20 (+8)	0	22 (+4)
Diffuser actor 5 demos/ task	24	64	40	28	44	68	40	24	44	0	37.6
+1% guidance	40 (+16)	92 (+28)	64 (+24)	40 (+12)	44	68	52 (+12)	24	84 (+40)	0	**50.8 (+13.2)**
+10% guidance	40 (+16)	84 (+20)	52 (+12)	28	52 (+8)	68	48 (+8)	32 (+8)	76 (+32)	0	48 (+10.4)

can be expressed as an element-wise weighted average:

$$\pi_{guided} = (1-\alpha)\pi \cdot \alpha G, \tag{3}$$

where $\alpha \in [0, 1]$ represents the percentage of guidance applied with respect to the base-policies distribution and is here denoted as *guidance factor*.

Adaptation of Regression Policies. To properly leverage the high-level guidance expressed in the guidance functions and the low-level capabilities of the base policy, it is desired that the policy's action space be expressed as a distribution. In the case of regression policies that do not provide uncertainty estimates, several strategies can be employed to infer the action distribution. One common approach is to assume a Gaussian distribution centered at the predicted value and compute the variance using ensembles of models trained with different initialization, different data samples, or different dropout seeds or different checkpoint stages [36]. Other strategies to infer the distributions of the model include using bootstrapping, Bayesian neural networks, or using Mixture of Gaussians [37].

IV. Experiments & Results

A. Experimental Setup

We demonstrate the efficacy of GRAPPA, in simulation on the RL-Bench benchmark [35] and on two challenging real-world tasks. For the real-world setup, we use the UFACTORY Lite 6 robot arm as the robotic agent and, as the end-effector, we use the included UFACTORY Gripper Lite, a parallel jaw gripper. The arm is mounted on a workbench. For perception, we use a calibrated RGB-D Camera, specifically the Intel RealSense Depth Camera D435i. All experiments were conducted on a desktop machine with a single NVIDIA RTX 4080 GPU, 64GB of RAM, and a AMD Ryzen 7 8700G CPU.

Base Policies: We evaluate the effectiveness of our guidance framework using different base policies, *Act3D* [38], *3D Diffuser Actor* [39], and a RandomPolicy. All policies plan in a continuous space of translations, rotations, and gripper state ($SE(3) \times \mathbb{R}$), however they utilize different inference strategies:

- ***Act3D*** samples waypoints in the Cartesian space ($\mathbb{R}^3$) and predicts the orientation and gripper state for the best scoring sampled waypoint, combining a classification and regression strategies into a single policy.
- ***3D Diffuser Actor***, on the other hand, uses a diffusion model to compute the target waypoints and infers the orientation and gripper state from the single forecast waypoint, thus tackling the problem as a single regression task.
- ***RandomPolicy*** denotes any of the former frameworks that has not been trained for a specific task, therefore the weights are randomly initialized.

The fundamentally different types of policies' outputs make them a great use case for our policy guidance framework. Furthermore, the common representation of the action and state spaces of both policies ($SE(3) \times \mathbb{R}$) provides a straightforward integration with our grounding models.

As described in Section III-D, the regression components of the policies require an adaptation to transform the single predictions of the model into a distribution over the action space. For the sake of simplicity, we assume a Gaussian distribution over the action space, with the mean centered on the predicted values and the standard deviation fixed on a constant value. The outputs of the classification component of Act3D (waypoint positions) were directly considered as samples of a distribution over the Cartesian space ($\mathbb{R}^3$).

The integration of base policies with the guidance distributions was performed by applying a weighted average parameterized by α as shown in Equation 3.

B. Experimental Evaluation

Our experimental evaluation aims to address the following questions: (1) Does GRAPPA enable policies to bridge the sim-to-real gap for deployment on real-world tasks and in out-of-distribution settings? (2) Does GRAPPA improve the performance of pre-trained base policies on specific robotics tasks and environments without additional human demonstrations? (3) Does GRAPPA enable policies to learn new skills from scratch? (4) What is the effect of guidance on expert versus untrained policies?

Does GRAPPA enable policies to bridge the sim-to-real gap for deployment on real-world tasks and in out-of-distribution settings? We first train a *3D Diffuser Actor* policy with 10 tasks with 100 demonstrations each with

a single RGB-D camera setup (*GNFactor* [40] setting), using exclusively simulation-based demonstrations. Note that this policy was chosen as it displayed stronger overall performance in the simulation benchmark. We roll out this policy in the real world in the button-pressing task in a cluttered environment, as depicted by Figure 4. To facilitate sim-to-real integration, we adapted the dimensions, scale, and alignment of the real-world workspace to match the simulation data, and positioned the RGB-D camera (Intel RealSense) in a similar placement as the camera used to train the base policy.

TABLE II: Real-world performance improvement in a sim-to-real setting. *Diffuser actor* policy trained on 10 tasks in simulation (*GNFactor* [40] setup: 100 demos/task) and evaluated on the tasks *"Push buttons"* with different guidance levels. **Red** cells refer to failures due to timeouts in task execution. **Orange** cells refer to errors in perception. **Green** cells refer to successes.

Policy	Task completion Success Rate [%]	Error Breakdown
Naive Sim2Real	0.0	
+15% guidance	66.7	
+50% guidance	50.0	
+75% guidance	100	
Finetuned Baseline *(10 real-world demos)*	33.0	
+15% guidance	50.0	
+50% guidance	16.7	
+75% guidance	100	

As depicted by Table II, the cluttered scenario in Figure 4 has shown to be out-of-distribution scenarios for the base policy that was solely trained in simulation (Naive Sim2Real). Fine-tuning the policy with 10 real-world demonstrations of pressing buttons in uncluttered environments has proven to slightly increase the performance of the policy. These base policies, combined with our guidance framework, achieve a higher success rate in 5 out of the 6 scenarios with guidance. In the only exception case (finetuned baseline + 50%), the base perception models failed to detect and keep track of the target button during the rollout—leading to a detrimental guidance of the policy.

A known limitation of diffusion-based policies is their struggle to generalize beyond their training set. To test the ability of GRAPPA to remedy this type of situation, we again take a policy trained in simulation and fine-tune on 15 demonstrations of a pick-and-place task in the real world, shown in Figure 5. We assess across two axes of generalization: 1) position generalization, where the objects are the same as used in the training demonstrations, but positions of the target are different; and 2) appearance generalization, where we vary the type of object used, while keeping the same semantic class of objects. Table III showcases the qualitative results for 10 rollouts: we see that in both types of generalization, the 3D Actor Diffuser is unable to complete the task in any trial, while GRAPPA using 50% of guidance succeeds in 3 out of 5 cases for position generalization and 4 out of 5 for appearance generalization.

TABLE III: GRAPPA guiding the base policy for out-of-distribution cases, illustrated in Figure 5.

Baseline	Position SR	Appearance 1	2	3	4	5	SR
3D Actor Diffuser	0/5	✗	✗	✗	✗	✗	0/5
+ GRAPPA (50%)	3/5	✓	✗	✓	✓	✓	4/5

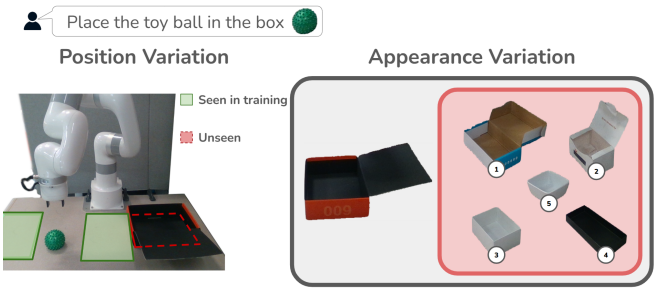

Fig. 5: GRAPPA guiding the base policy for out-of-distribution cases. The task involves grasping a deformable toy ball and placing it inside a box.

Across both real-world tasks, a higher percentage of guidance is necessary compared to the simulation results seen in Table I. This hints at two practical properties of GRAPPA's guidance effect. First, there must be a trade-off between high-level guidance (provided by GRAPPA) and low-level control (provided by the base policy); in more complex tasks, elevated guidance values disrupt the robot's low-level motions, leading to some failure cases. We compare trajectories of the failing base policy versus GRAPPA-guided in Figure 6. Second, the optimal guidance level is also dependent on the performance of the base policy itself. Notably, while the simulation-based policy has already achieved a good performance on the test set, it is unable to deal with out-of-distribution cases and requires much more correction in the real-world setting.

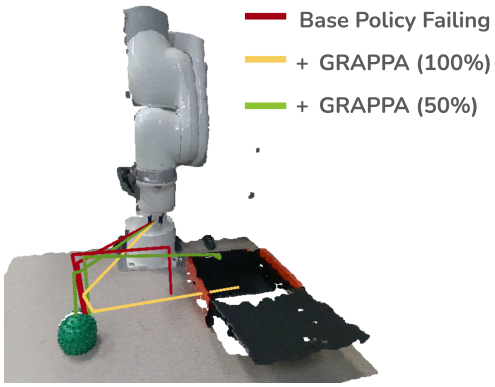

Fig. 6: Illustration of the effect of different guidance percentages on a failure case of the base policy. In red we show the base policy failing in an out-of-distribution scenario; with 100% of guidance (yellow), the end position is successfully above the box, but it has *lost low-level notions*. By balancing both with intermediate guidance (50%) shown in green, we can complete the task.

Does GRAPPA improve the performance of pre-trained

base policies on specific robotics tasks and environments without additional human demonstrations? We first assess the effect of the proposed guidance on the *Act3D* and *3D Diffuser Actor* baselines following the *GNFactor* [40] setup, which consists of a single RGB-D camera and table-top manipulator performing 10 challenging tasks with 25 variations each. Guidance is iteratively generated for the failure cases. For the failed rollouts, our policy improvement framework ran for 5 iterations. As displayed by Table I, the framework was able to improve the success rate of the base policy on most of the tasks, with the best results achieved by using 1% guidance. The low amount of guidance has shown to be enough to bend the action distribution to the desired direction, while still preserving the low-level nuances captured by the base policy. This suggests that GRAPPA is capable of improving base policies by adding abstract understanding and grounding of the desired task, while preserving the low-level movement profiles captured by the original policies.

Does GRAPPA enable policies to learn new skills from scratch? We evaluated the performance of the framework on learning new skills from scratch on 4 tasks of the RL-Bench benchmark: *turn tap*, *push buttons*, *slide block to color target*, *reach and drag*. In this setup, we initialized the Act3D policy with random weights and applied 100% of guidance ($\alpha = 1.0$) over the policy for the x, y, and z components. Only leveraging the waypoint sampling mechanism of Act3D and overwriting its distribution with the values queried from the guidance functions generated. The results show that the framework is capable of learning new skills from scratch, achieving a higher success rate than the base policy pre-trained on 5 demonstrations/tasks for the tasks "turn tap" and "push buttons" tasks. When utilizing only the untrained Act3D policy (without guidance) the policy achieved 0 success rate on the tasks. Figure 7 demonstrates the iterative improvement of our guidance framework, wherein the guidance code generated for each failed rollouts from the previous iteration is iteratively updated.

It is worth mentioning that a few variations of the simulated tasks "turn tap" and "reach and drag", which seemly would require a precise orientation control of the manipulator, could be solved by guiding only the Cartesian components of the police's output. For these variations, a qualitative analysis shows that successful roll-outs could be achieved by tapping the end-effector on the target objects.

What is the effect of guidance on expert versus untrained policies? As discussed previously, GRAPPA can learn tasks from scratch using a random base policy with 100% of guidance (α). Given that we are not affecting the policy, the product of the iterative learning from scratch is the generated guidance script which captures a high-level understanding of the task, e.g., spatial relationships, and task completion criteria, among others. We can qualitatively see the effect of the learning process by comparing the guidance scripts for different iterations of the same task.

On the other hand, applying the guidance to an existing expert policy aims to shift the action distribution to account for failure cases, like potentially out-of-distribution scenes. The goal here is to use the gained high-level understanding to aid the policy in task completion. Table I shows that adding too much weight to the guidance function can yield diminishing returns as it can overpower the nuanced low-level details from the expert policy: notice that the performance gain across the board is bigger using 1% of guidance.

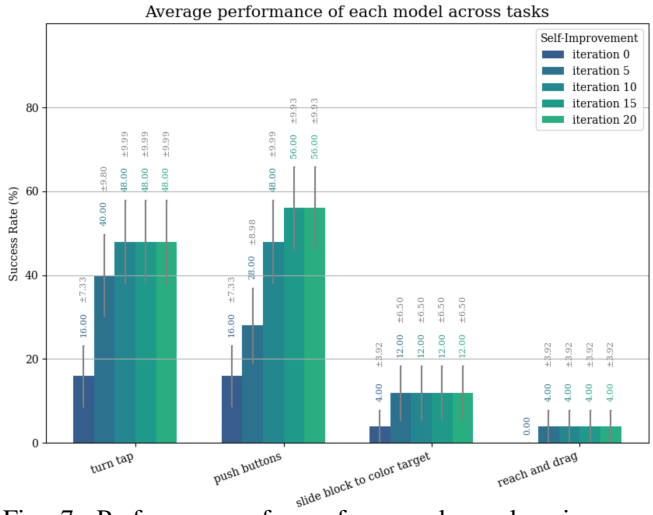

Fig. 7: Performance of our framework on learning new skills from scratch (guidance over untrained policies), and iteratively improving the guidance functions generated.

V. Conclusion

Summary: In this work, we proposed GRAPPA, a novel framework for the self-improvement of embodied policies. Our self-guidance approach leverages the world knowledge of a group of conversational agents and grounding models to guide policies during deployment. We demonstrated the effectiveness of our approach in autonomously improving manipulation policies and learning new skills from scratch, in simulated RL-bench benchmark tasks and in two challenging real-world tasks. Our results show that the proposed framework is especially effective in improving the following high-level task structures and key steps to solve the task. This capability can be well suited for improving pre-trained policies that struggle with long-horizon tasks or for learning new simple skills from scratch.

Limitations: From an analysis of the guided rollouts, a few of the task variations proved challenging for the perception models used by the grounding agent, leading to false positive detections or failure to locate specific objects. This was mainly observed in the simulation, given that simulated graphics are often not represented in the training set of detection models. A potential solution is to integrate more robust object detection models or verification procedures to ensure the correct detection of objects in the scene. Additionally, occasional scene understanding errors by the VLM led to inaccurate guidance and unexpected behaviors.

References

[1] Y. Hu, Q. Xie, V. Jain, J. Francis, J. Patrikar, N. Keetha, S. Kim, Y. Xie, T. Zhang, Z. Zhao, *et al.*, "Toward general-purpose robots

via foundation models: A survey and meta-analysis," *arXiv preprint arXiv:2312.08782*, 2023.

[2] S. Yenamandra, A. Ramachandran, M. Khanna, K. Yadav, J. Vakil, A. Melnik, M. Büttner, L. Harz, L. Brown, G. C. Nandi, *et al.*, "Towards open-world mobile manipulation in homes: Lessons from the neurips 2023 homerobot open vocabulary mobile manipulation challenge," *arXiv preprint arXiv:2407.06939*, 2024.

[3] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. Reid, and N. Suenderhauf, "Sayplan: Grounding large language models using 3d scene graphs for scalable robot task planning," in *7th Annual Conference on Robot Learning*, 2023.

[4] S. Saxena, B. Buchanan, C. Paxton, B. Chen, N. Vaskevicius, L. Palmieri, J. Francis, and O. Kroemer, "Grapheqa: Using 3d semantic scene graphs for real-time embodied question answering," *arXiv preprint arXiv:2412.14480*, 2024.

[5] 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.

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

[7] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 9493–9500.

[8] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, "Progprompt: Generating situated robot task plans using large language models," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 11 523–11 530.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] C. Huang, O. Mees, A. Zeng, and W. Burgard, "Visual language maps for robot navigation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 10 608–10 615.

[16] H. Ha, P. Florence, and S. Song, "Scaling up and distilling down: Language-guided robot skill acquisition," in *Conference on Robot Learning*. PMLR, 2023, pp. 3766–3777.

[17] Y. Ding, X. Zhang, C. Paxton, and S. Zhang, "Task and motion planning with large language models for object rearrangement," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 2086–2092.

[18] Z. Michał, C. William, P. Karl, M. Oier, F. Chelsea, and L. Sergey, "Robotic control via embodied chain-of-thought reasoning," *arXiv preprint arXiv:2407.08693*, 2024.

[19] 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.

[20] X. Li, M. Zhang, Y. Geng, H. Geng, Y. Long, Y. Shen, R. Zhang, J. Liu, and H. Dong, "Manipllm: Embodied multimodal large language model for object-centric robotic manipulation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 18 061–18 070.

[21] Y. Mu, Q. Zhang, M. Hu, W. Wang, M. Ding, J. Jin, B. Wang, J. Dai, Y. Qiao, and P. Luo, "Embodiedgpt: Vision-language pre-training via embodied chain of thought," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[22] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "Voxposer: Composable 3d value maps for robotic manipulation with language models," *arXiv preprint arXiv:2307.05973*, 2023.

[23] J. Liang, F. Xia, W. Yu, A. Zeng, M. G. Arenas, M. Attarian, M. Bauza, M. Bennice, A. Bewley, A. Dostmohamed, *et al.*, "Learning to learn faster from human feedback with language model predictive control," *arXiv preprint arXiv:2402.11450*, 2024.

[24] M. Xu, P. Huang, W. Yu, S. Liu, X. Zhang, Y. Niu, T. Zhang, F. Xia, J. Tan, and D. Zhao, "Creative robot tool use with large language models," *arXiv preprint arXiv:2310.13065*, 2023.

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

[26] M. Parakh, A. Fong, A. Simeonov, T. Chen, A. Gupta, and P. Agrawal, "Lifelong robot learning with human assisted language planners," *arXiv e-prints*, pp. arXiv–2309, 2023.

[27] L. Guan, Y. Zhou, D. Liu, Y. Zha, H. B. Amor, and S. Kambhampati, "" task success" is not enough: Investigating the use of video-language models as behavior critics for catching undesirable agent behaviors," *arXiv preprint arXiv:2402.04210*, 2024.

[28] J. Rocamonde, V. Montesinos, E. Nava, E. Perez, and D. Lindner, "Vision-language models are zero-shot reward models for reinforcement learning," *arXiv preprint arXiv:2310.12921*, 2023.

[29] Y. Cheng, L. Li, Y. Xu, X. Li, Z. Yang, W. Wang, and Y. Yang, "Segment and track anything," *arXiv preprint arXiv:2305.06558*, 2023.

[30] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu, *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," *arXiv preprint arXiv:2303.05499*, 2023.

[31] Y. Cheng, L. Li, Y. Xu, X. Li, Z. Yang, W. Wang, and Y. Yang, "Segment and track anything," 2023. [Online]. Available: https://arxiv.org/abs/2305.06558

[32] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," 2023. [Online]. Available: https://arxiv.org/abs/2201.11903

[33] Z. Yang and Y. Yang, "Decoupling features in hierarchical propagation for video object segmentation," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[34] A. Maćkiewicz and W. Ratajczak, "Principal components analysis (pca)," *Computers & Geosciences*, vol. 19, no. 3, pp. 303–342, 1993.

[35] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "Rlbench: The robot learning benchmark & learning environment," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3019–3026, 2020.

[36] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, *et al.*, "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," *Information fusion*, vol. 76, pp. 243–297, 2021.

[37] J. Mena, O. Pujol, and J. Vitrià, "A survey on uncertainty estimation in deep learning classification systems from a bayesian perspective," *ACM Computing Surveys (CSUR)*, vol. 54, no. 9, pp. 1–35, 2021.

[38] T. Gervet, Z. Xian, N. Gkanatsios, and K. Fragkiadaki, "Act3d: 3d feature field transformers for multi-task robotic manipulation," in *7th Annual Conference on Robot Learning*, 2023.

[39] T.-W. Ke, N. Gkanatsios, and K. Fragkiadaki, "3d diffuser actor: Policy diffusion with 3d scene representations," *arXiv preprint arXiv:2402.10885*, 2024.

[40] Y. Ze, G. Yan, Y.-H. Wu, A. Macaluso, Y. Ge, J. Ye, N. Hansen, L. E. Li, and X. Wang, "Gnfactor: Multi-task real robot learning with generalizable neural feature fields," 2024. [Online]. Available: https://arxiv.org/abs/2308.16891