WoMAP: World Models For Embodied Open-Vocabulary Object Localization

Author Names Omitted for Anonymous Review.

***Abstract*—Language-instructed active object localization is a critical challenge for robots, requiring efficient exploration of partially observable environments. However, state-of-the-art approaches either struggle to generalize beyond demonstration datasets (e.g., imitation learning methods) or fail to generate physically grounded actions (e.g., VLMs). To address these limitations, we introduce WoMAP (World Models for Active Perception): a recipe for training open-vocabulary object localization policies that: (i) uses a Gaussian Splatting-based real-to-sim-to-real pipeline for scalable data generation without the need for expert demonstrations, (ii) distills dense rewards signals from open-vocabulary object detectors, and (iii) leverages a latent world model for dynamics and rewards prediction to ground high-level action proposals at inference time. Rigorous simulation and hardware experiments demonstrate WoMAP's superior performance in a wide range of zero-shot object localization tasks, with more than 7x and 2.5x higher success rates compared to VLM and diffusion policy baselines, respectively. Further, we show that WoMAP achieves strong sim-to-real transfer in experiments on a TidyBot robot.**

***Index Terms*—Active Perception, World Models, Object Localization.**

I. Introduction

Perceptual activity in biological agents is inherently active and exploratory [11, 2]. As an example, consider the task of *open-vocabulary object localization*, where an agent needs to approach a target object specified by natural language in a previously unseen environment. In such settings, humans will actively seek information guided by prior expectations to search efficiently. For example, when looking for keys, we preferentially inspect locations where they are most likely to be found, e.g., near the door or on the couch.

However, reproducing intelligent search behavior for robots remains challenging, as it requires interpreting open-vocabulary object descriptions and commonsense reasoning from partial observations in unfamiliar environments. While vision-language models (VLMs) provide useful heuristics for exploration [5, 6, 40, 32], effectively grounding these high-level action proposals to physical execution is a non-trivial problem. This grounding can be achieved via imitation learning methods [30, 45], which require large-scale expert demonstrations and can struggle to generalize beyond demonstration datasets. Alternatively, reinforcement learning (RL) [44, 9, 28] offers another route to grounding, but is challenging to employ without an accurate simulation environment.

To address these challenges, we present **WoMAP** (**Wo**rld **M**odels for **A**ctive **P**erception): a novel recipe for efficient

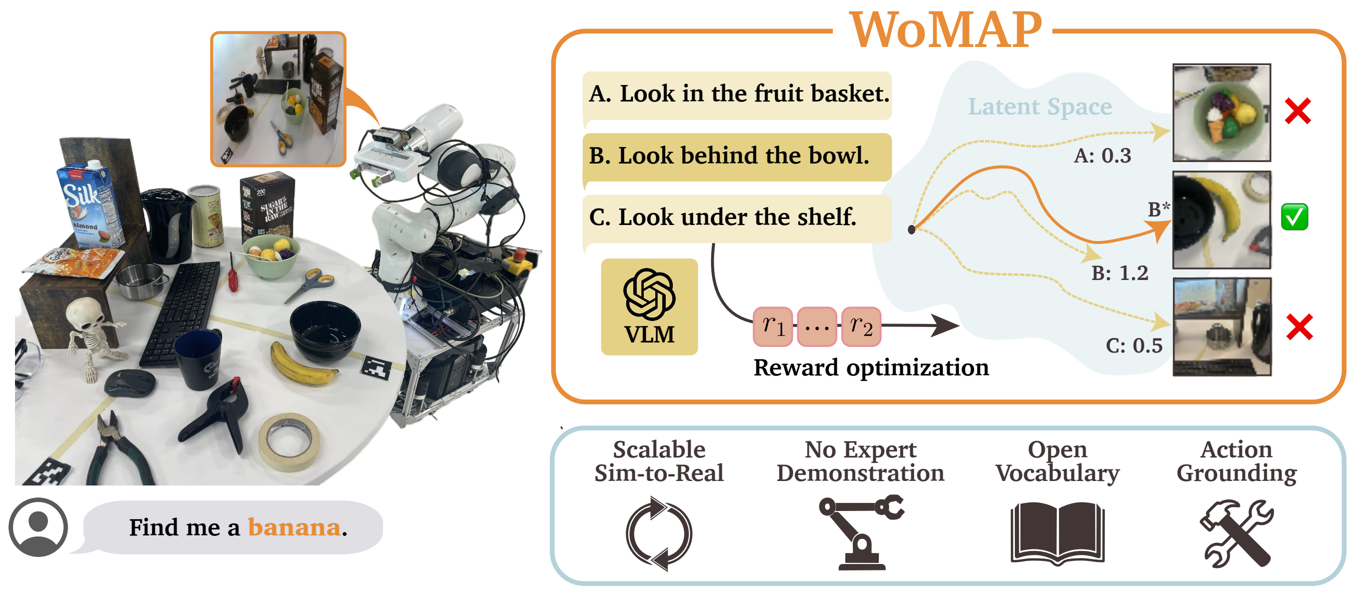

Fig. 1: **WoMAP** leverages a learned world model to evaluate and optimize actions given high-level proposals and chooses the candidate with the highest predicted reward. In this example, the VLM suggested three exploration directions; after analyzing the outcome of each action roll-out in latent space, WoMAP selects "looking behind the bowl" as the optimal choice.

active object localization that can be trained without expert demonstrations or online interactions with the environment (**??**). In order to achieve this, we propose an approach that learns a *latent world model* [13] using three key ingredients (Figure 2, left). First, we introduce a scalable data generation pipeline based on Gaussian Splatting [19] that allows us to generate photorealistic data with broad coverage using real-world videos. Second, we propose a training framework that is *reconstruction-free*; instead of using image reconstruction as a supervisory training signal (which can lead to poor generalization, training stability, and sample efficiency [46]), we construct dense rewards from the confidence outputs of open-vocabulary object detectors and *distill* these into the latent space of the world model. Finally, we present an inference-time planning scheme that optimizes high-level action proposals from VLMs using the trained world model.

Taken together, we contribute a novel approach to open-vocabulary object localization that can be trained in a data-efficient manner, generalize to novel scenes and object descriptions, and exploit commonsense reasoning abilities of VLMs. We demonstrate our approach on a suite of simulated and real-world object localization tasks and demonstrate significant improvements (2.5x – 7x higher success rates) over baselines that only utilize imitation learning or VLMs.

II. Related Work

Active Object Localization. Broadly, active object localization has been explored with both end-to-end approaches [29, 10], such as imitation learning (IL) [30, 45] and reinforcement learning (RL) [44, 9, 28], and modular approaches incorporating foundation models [5, 6, 40]. End-to-end methods map visual observations directly to actions but typically require large amounts of expert demonstrations or interactions to learn effective exploration behaviors, [29, 30] and they generalize poorly to new environments or tasks [47]. In contrast, WoMAP does not require on-policy demonstrations or interactions with the environment, and leverages a learned world model to plan sequences of actions at inference time in order to achieve strong generalization across tasks and sim-to-real gaps.

Other modular approaches incorporate foundation models such as pre-trained object detectors or VLMs [6, 40, 18] to reason over observations and plan exploration. However, they rely heavily on the accuracy of each modular component, and usually require engineering effort to ground executions using other modalities or scene representations. WoMAP instead directly optimizes VLM outputs within a learned environment model, offering a light-weight solution for grounded actions with minimal reliance on external representations. Finally, in terms of task setup, most prior work focuses on simulated indoor navigation, utilizing rich contextual cues (e.g., the sofa is more likely to be in the living room) and restricting the action space in 2D for tractability. In contrast, our framework makes no such assumptions and addresses more general settings requiring full 6D camera control to locate objects in cluttered scenes.

World Models for Robotics. World models have become increasingly prominent in robotics, providing predictive foresight in learning action-conditioned dynamics and planning over longer horizons [22, 43, 46, 23]. To capture environment dynamics that generalize to test-time distributions, existing works typically rely on large quantities of uncurated data [43, 46], expert demonstrations [23, 3], or on-policy interactions [16], all of which are costly and labor-intensive to collect in real-world settings. In contrast, WoMAP learns a policy-agnostic environment model from synthetically generated offline data via Gaussian Splatting [19]. Many existing works also employ an image reconstruction loss to provide dense learning signals [43, 25], but often result in greater model complexity and unstable training. WoMAP instead leverages a pretrained latent representation with dense reward signals to encode rich visual and spatial information, improving both scalability and robustness for downstream tasks.

III. Methodology

A. Problem Formulation

We consider an open-vocabulary object localization task with a robot equipped with an onboard RGB camera with a six degree-of-freedom (6-DoF) action space, operating in an environment $E \in \mathcal{E}$. We model the problem as a partially observable Markov decision process (POMDP), where given the current state of the environment and the robot, the camera returns a partial observation o_t under sensing uncertainty, occlusions, and limited field of view. At each time step, the robot selects a continuous action $a_t \in \mathbb{R}^6$, corresponding to camera translation and rotation in 3D space, to obtain a new observation. Given a language description l of the target object $\mathcal{T}_g$, the robot seeks to efficiently obtain a best view: $\max_{a_{0:t}} \mathcal{R}(o_t, \mathcal{T}_g)$, where $\mathcal{R} \in [0, 1]$ is the object localization reward describing how well the target object $\mathcal{T}_g$ is identified in o_t.

Our proposed framework, WoMAP, uses a world model to capture latent space dynamics and reward prediction that can generalize to any $E \in \mathcal{E}$. However, as discussed in Section I, learning a world model that operate across such diverse task settings is non-trivial, requiring training data with sufficient coverage, strong supervisory reward signals, and the ability to incorporate high-level commonsense reasoning during planning. In the following sections, we describe the three core components of WoMAP as illustrated in Figure 2, addressing each of these fundamental challenges.

B. Scalable Data Generation

Unlike imitation learning methods, world models do not require expert trajectories for training, which are generally expensive to collect. However, they do require sufficient data coverage to effectively capture the dynamics of the environment, which necessitates strategic data collection design to maximize sample efficiency. Gathering diverse observation data from the real-world also poses significant challenges, and becomes hard to scale as the number of training environments increases.

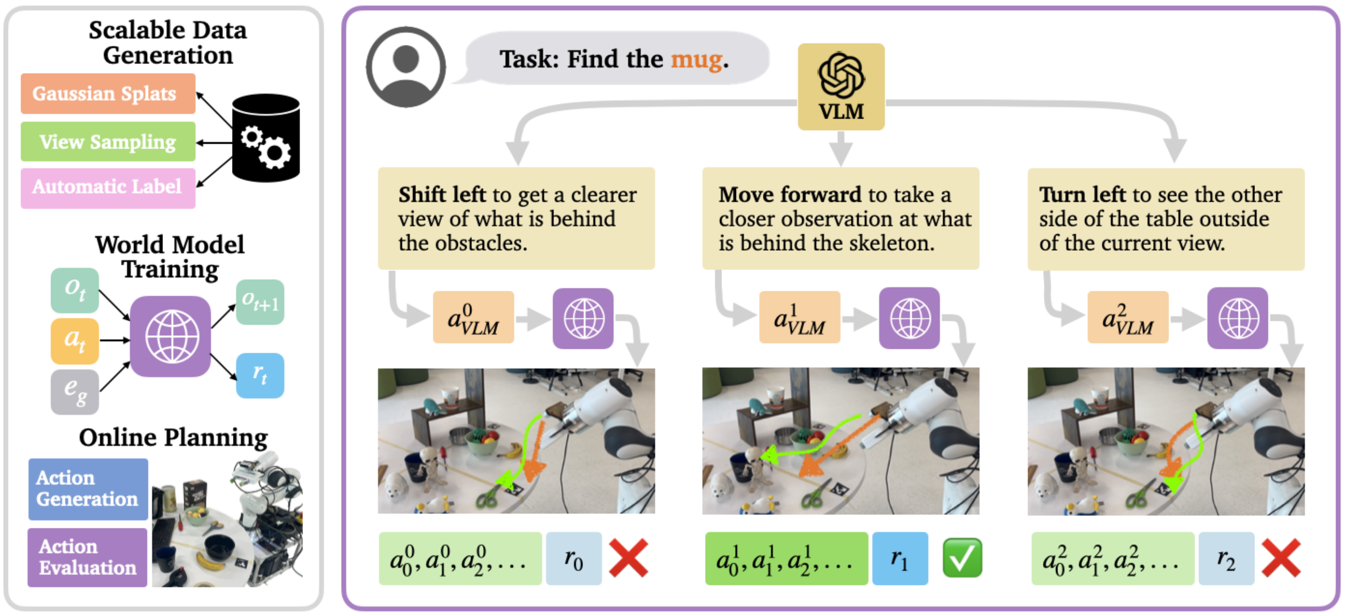
Scalable Data Generation
Gaussian Splats
View Sampling
Automatic Label
World Model Training
Online Planning
Action Generation
Action Evaluation
Task: Find the mug.
VLM
Shift left to get a clearer view of what is behind the obstacles.
Move forward to take a closer observation at what is behind the skeleton.
Turn left to see the other side of the table outside of the current view.

Fig. 2: **Left: Three Core Components of WoMAP:** scalable data generation with Gaussian Splats (III-B), world model with object detection reward supervision (III-C), latent space action planning (III-D). **Right: The action optimization/selection process.** Given the task and current observation, VLM generates high-level proposals which we translate to coarse actions; we further optimize each action within WoMAP's reward gradient field, and execute the sequence with the highest predicted reward.

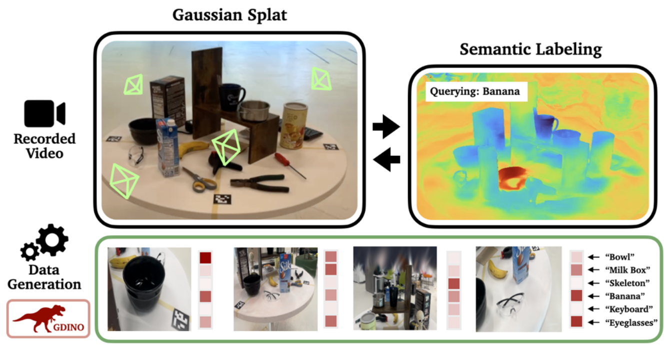

Fig. 3: **Data Generation with Gaussian Splats.** We train Gaussian Splats for each scene and obtain ground truth object locations through semantic labeling [36] for informative view sampling. Each observation is labeled with GroundingDINO [21] to get confidence scores for all training targets.

In WoMAP, we introduce a scalable real-to-sim-to-real data generation pipeline that utilizes only a few real-world videos to efficiently generate diverse training data. Our pipeline leverages Gaussian Splatting [19] to generate a photorealistic simulation environment from video input and can render arbitrary views given the view matrix. Shown in Figure 3, we automatically annotate the location and dimension of each target in the training scene by distilling language semantics from CLIP [27] from semantic Gaussian Splats [36]. Within the trained splat, we collect a training dataset $\mathcal{D}$ consisting of M observation-reward-pose tuples, i.e., $\mathcal{D} = \{(o_i, r_i, P_i),\ \forall i \in [M]\}$, where $P_i \in \mathbb{R}^6$ denotes the spatial camera pose, and o_i is generated by rendering the scene with the computed view matrix at P_i. At training time, given two samples we compute the action a_{ij} required to transition from P_i to P_j, since training the world model does not require sequentially-ordered data.

Further, to improve sample efficiency, we design our data distribution to concentrate on trajectories starting from random initial positions and leading towards sampled target objects, with added linear and angular perturbations for data augmentation. These trajectories can be generated with any planning algorithm and require no human demonstration. We show more implementation details in Appendix A-A3. In Section IV-D. We demonstrate that despite only training on synthetically rendered images in GSplat, WoMAP still achieve strong zero-shot sim-to-real performance.

C. World Models for Active Perception

Given data generated from Sec. III-B, we outline key design choices that enable the world model to accurately capture evolving dynamics with environment interactions. A central innovation of our pipeline is the use of dense reward distillation from open-vocabulary object detectors, which allows for data-efficient training without relying on image reconstruction objectives.

1) World Model Architecture: As shown in Figure 4, the world model consists of three standard core components [13, 14]:

$$\begin{aligned} &\text{Observation Encoder:} && z_t = h_\theta(o_t), \\ &\text{Dynamics Predictor:} && z_{t+1} \sim q_\psi(z_{t+1} \mid z_t, a_t), \qquad (1) \\ &\text{Rewards Predictor:} && r_t \sim v_\phi(r_t \mid z_t, e_g), \end{aligned}$$

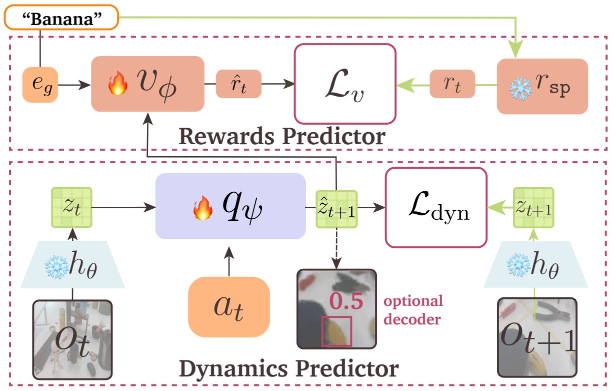

Fig. 4: **World Model Architecture** for simultaneous dynamics and rewards prediction.

where $z_t \in \mathbb{R}^d$ denotes the latent state, e_g represents the language embedding computed from a description l of the target object $\mathcal{T}_g$, $r_t \in \mathbb{R}$ denotes the associated reward with z_t when querying for $\mathcal{T}_g$, and θ, ψ, ϕ denote network parameters for each component of the world model.

The observation encoder h_θ maps a high-dimensional camera observation o_t to a compact latent space z_t. Inspired by [46], we leverage a pre-trained vision encoder model DINOv2 [24] to directly compute flattened patch embeddings of the given image as z_t to retain rich visual and spatial features resulting from large-scale pre-training. The dynamics predictor q_ψ models the transition distribution $p(z_{t+1} \mid z_t, a_t)$ using a standard ViT architecture [8]. With variational inference, we parametrize q_ψ as a Gaussian distribution $q_\psi(z_{t+1} \mid z_t, a_t) \sim \mathcal{N}(\mu_t, \sigma_t^2)$ and minimize the Kullback-Leibler (KL) divergence between the true state transition, $p(z_{t+1} \mid z_t, a_t)$ and q_ψ, with the loss function: $\mathcal{L}_{\text{dyn}} = \text{KL}(p(z_{t+1} \mid z_t, a_t) \| q_\psi(z_{t+1} \mid z_t, a_t))$. During training, we supervise q_ψ recurrently on a sequence of H observation-action pairs $\{(o_i, a_i)\}_{i=1}^{H}$ to enforce dynamics consistency. The rewards predictor v_ϕ estimates the reward for each latent state, conditioned on the language embedding of the task $\mathcal{T}_g$. Additional implementation details can be found in Appendix A-B.

2) Reward Distillation: Despite providing dense reward signals, image reconstruction objectives often lead to training instability [15, 4], which we further demonstrate in Appendix A-B1. To tackle this challenge, WoMAP introduces a novel reward distillation procedure that generates dense rewards signals without reconstruction objectives for data-efficient training. As shown in Figure 3, during data generation, WoMAP computes a per-frame reward $r_{\texttt{det}}(o_t, \mathcal{T}) \in [0, 1]$ for each object $\mathcal{T}$ in the observation o_t using the detection confidence provided by a pretrained object detector, e.g., GroundingDINO [21], scaled by the associated detection bounding-box size. This annotation procedure yields a rich, task-relevant training signal for each object in the scene, and scales efficiently with environment complexity by enabling parallel processing of detections. At training time, we distill the reward signal $r_{\text{sp}}(o_t, e_g) \in [0, 1]$ from $r_{\texttt{det}}(o_t, \mathcal{T})$ by using it as a supervisory signal for the rewards predictor v_ϕ conditioned on the language embedding e of each relevant object. This distillation pipeline enables an effective planning framework using world models, which we discuss in the following subsection.

D. Planning with WoMAP

While world models can directly plan actions via sampling or gradient-based optimization, such methods are often inefficient in continuous action spaces without informed guidance. Especially for complex problem spaces, gradient-based methods struggle to localize target objects within a finite optimization budget and frequently converge to suboptimal solutions. WoMAP addresses this limitation by leveraging VLMs' commonsense reasoning to generate informed action proposals, which are then refined via model predictive control using a world model for spatial grounding.

For a given task instruction, we prompt a VLM using chain-of-thought prompting [39] to provide high-level guidance for promising locations for the robot to explore. In preliminary experiments, we observed that VLMs struggle with spatial understanding when prompted for numerical relative actions given an input image. Consequently, we query the VLM using a multiple-choice prompt with the options given by a fixed set of textual description of the possible actions, e.g., "turn left," or "move forward." We provide additional implementation details in Appendix A-C. Subsequently, WoMAP optimizes a set of candidate actions provided by the VLM to maximize the expected rewards:

$$\begin{aligned} \max_{a_{t:t+T}} \quad & \sum_{\tau=1}^{T} (\mathbb{E}_{v_\phi}[r_{t+\tau} \mid z_{t+\tau}, e_g)] + \gamma \, \|a_{t+\tau-1} - a_{t+\tau-2}\|_1), \\ \text{subject to} \quad & z_{t+\tau} \sim q_\psi(z_{t+\tau} \mid z_{t+\tau-1}, a_{t+\tau-1}) \;\; \forall \tau \in [T], \end{aligned} \tag{2}$$

at each timestep t with latent state z_t, target-object language embedding e_g, previous control action a_{t-1}, MPC planning horizon T, and weight $\gamma \in \mathbb{R}_+$. While the first objective term seeks to maximize the expected rewards, the second objective term incentivizes smoothness of the robot's trajectories.

Figure 2 illustrates the trajectory planning process on the right panel. The VLM provides three action proposals. WoMAP optimizes each of the three action proposals and estimates their reward. It then execute the optimized actions with the highest reward.

IV. Experiments

We evaluate WoMAP on open-vocabulary active object localization tasks both in simulation and on a TidyBot [42] to answer the following questions: (1) How does WoMAP compare to prior work across a range of environments with different task difficulty, defined by scene complexity, occlusion degree, and initial conditions? (2) Can WoMAP achieve strong sim-to-real transfer when trained only on photorealistic simulation data? (3) Can WoMAP achieve zero-shot generalization to visual (unseen lighting, backgrounds) and semantic (unseen instructions, target objects) conditions? In Appendix A-B1, we ablate training with image reconstruction objectives, freezing

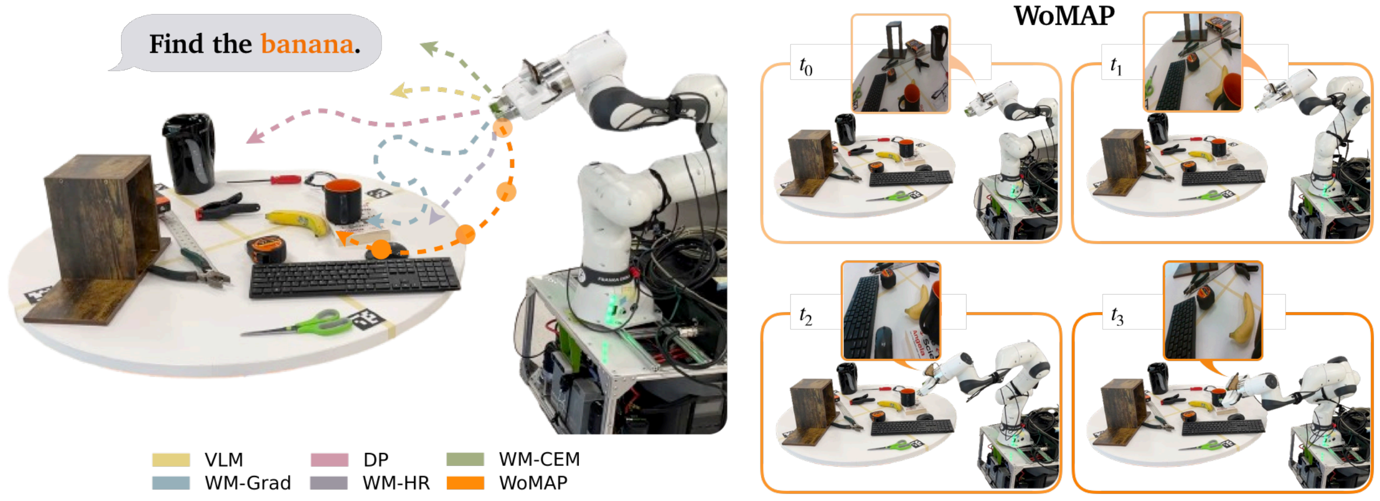

Fig. 5: Visualization of the TidyBot's trajectories for all planners, when asked the find a banana occluded by a mug. While other planners fail to find the banana, WoMAP finds the banana efficiently, unlike WM-Grad, which computes inefficient, circuitous paths. Note note that the VLM fails to closely approach the banana, the DP does not look behind occlusions, WM-CEM is sample-inefficient, and WM-HR is less robust to hallucinations by the world model. Further, (right) we show images from the scene and wrist cameras at different timesteps when planning with WoMAP.

vs. finetuning pretrained image encoders, and training image encoders from scratch.

A. Environments and Tasks

We evaluate all methods in four PyBullet (PB) simulation environments [7] and three real-world environments both within Gaussian Splat (GS) and on a TidyBot, designing the evaluation environments with a particular focus on practical scene configurations and task difficulty. Within each environment, we create distinct scenes by fixing a set of target objects (listed in Appendix B-A) but randomly arranging them, while also randomly adding other objects, creating distractions and occlusions. We curate four distinct environments in PB and three in GS, themed on office, kitchen, and random everyday objects, with varying difficulties. For a comprehensive evaluation of all methods, we vary the task difficulty along two axes: (i) *scene difficulty*, determined by the number, diversity, and layout of objects in the scene, and (ii) *initial-pose difficulty*, representing the task difficulty due to occlusions, viewability, and distance to the target object which depends on the initial pose of the robot. In Appendix B-A, we provide a detailed discussion of the evaluation setup, as well as illustrations of the tasks and environments.

B. Baselines, Ablations, and Evaluation Metrics

We benchmark WoMAP against a VLM-based planner using GPT-4o [17], similar to prior work [12, 32]. We prompt GPT-4o for the best action given the observation o_t using the same prompt template as WoMAP's, described in Appendix A-C. In addition, we compare against a multi-task diffusion policy (DP), trained on each environment using expert trajectories for multiple objects. We also benchmark against world model-only planners, using (i) derivative-free, cross-entropy optimization for action proposals (*WM-CEM*), inspired by [46]; (ii) gradient-based action optimization (*WM-Grad*); and (iii) heuristic-guided action proposals (*WM-HR*) without a VLM, consisting of a fixed set of atomic actions that was also used to prompt the VLM, which are further optimized via gradient descent. We use success rate and efficiency (given by the success rate weighted by the path length [1]) as evaluation metrics, where success is defined by a threshold on the detection confidence of the target object and the proportion of the associated bounding-box in the robot's camera image. See Appendix B-C for additional implementation details.

C. Evaluation across Varying Task Difficulty

While varying the scene difficulty and the initial-pose difficulty, we examine the performance of each method in novel (unseen) scenes in PyBullet and Gaussian Splat environments. Figures 6 and 7 summarize our main results, showing that on average, WoMAP significantly outperforms state-of-the-art VLM and DP baselines by more than $9\times$ and $2\times$, respectively. While the VLM planner fails to account for physical grounding and the DP policy struggles to generalize beyond the training distribution, WoMAP generates grounded actions across diverse scenes. Moreover, we observe a progressive increase in the performance of the world-model-based planners with more informed search methods, in the order of (i) sampling-based WM-CEM, which is sample-inefficient even with $4\times$ as many action proposals, (ii) gradient-based WM-Grad, which relies on myopic local gradients and generates inefficient, circuitous actions, evidenced by its much lower efficiency compared to success scores, (iii) heuristics-based WM-HR, which does not leverage intelligent guidance from the VLM, limiting

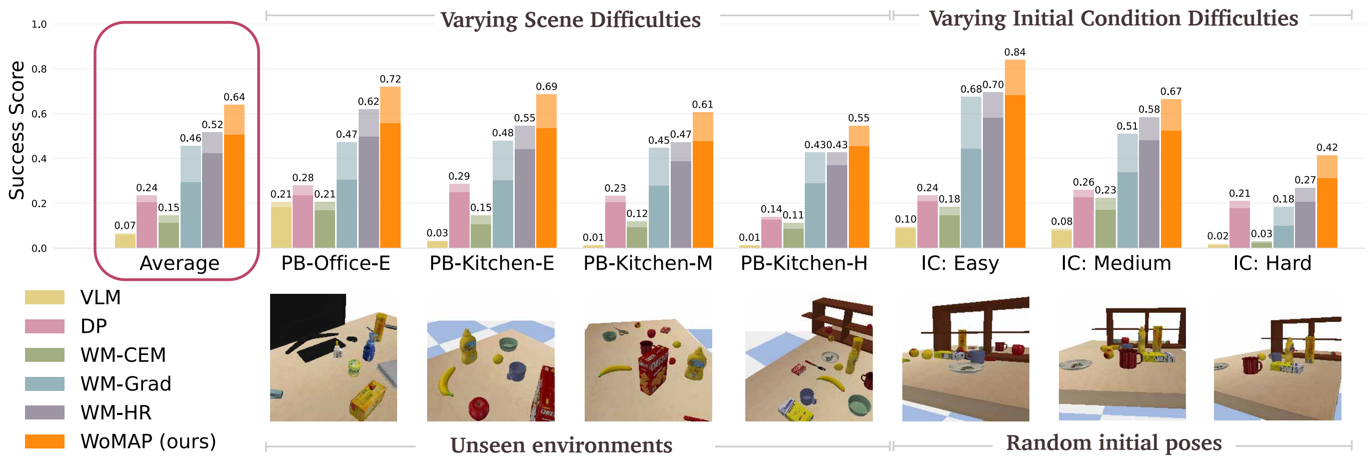

Fig. 6: **PyBullet evaluation tasks and results.** Success rates (translucent bars) and efficiency scores (solid bars) in active object localization across PyBullet scenes (presented in the order of increasing difficulty) and initial-pose conditions: easy (E), medium (M), and hard (H). WoMAP outperforms all baseline methods in all scenes and initial-pose conditions.

its performance in challenging problems, and ultimately (iv) WoMAP, which evaluates and optimizes the VLM action proposals with the world model.

In Figure 5, we visualize the trajectories computed by all planners on a TidyBot robot tasked with finding a banana occluded behind a mug in the real-world. We observe the same findings as that of the simulation environments. Notably, the VLM fails to closely approach the banana, and the DP does not look behind occlusions. Further, the scene and wrist camera images demonstrate the efficiency of WoMAP in localizing the banana. Next, we discuss the performance of the planners with respect to the difficulty of the scene and initial pose and direct readers to Appendix B for ablations on the correlation between data quantity/scene diversity and performance.

Varying Scene Difficulty. As expected, the performance of all methods decreases with increasing scene difficulty, with a drop in success rates over 50% for VLM and DP baselines. In comparison, WoMAP's performance only drops by 23.6% in PyBullet and 40.2% in Gaussian Splat. Notably, all world-model-based planners exhibit lower performance drops, suggesting the importance of planning with physical priors provided by world models.

Varying Initial Conditions. WoMAP achieves the second-smallest performance drop in the PyBullet environments (after the DP) and the smallest performance drop in the Gaussian Splat environments, even with its highest absolute scores. By leveraging the high-level reasoning capabilities of the VLM for action proposals, WoMAP effectively mitigates hallucinations in world models that arise in difficult-to-predict scenarios, e.g., occlusions.

D. Sim-to-Real Transfer with Gaussian Splats

We evaluate WoMAP's sim-to-real transfer ability on 20 hardware trials for each of the 3 corresponding real-world tasks using the TidyBot [42] platform. For each trial, we randomize both the scene configuration and target object to include a diverse range of initial conditions and task difficulties.

TABLE I: Success rates (%) for zero-shot sim-to-real transfer for VLM and WoMAP.

Model	GS-Kitchen	GS-Office	GS-Random
VLM (sim)	6	13	3
VLM (real)	0	5	0
WoMAP (sim)	71	65	32
WoMAP (real)	55	65	63

TABLE II: Visual generalization results for various background and lighting conditions.

Axis	Success Rate %	Efficiency %
Nominal	63	60
Lighting	50	47
Backgrounds	30	28

As shown in Table I, despite being trained entirely in the Gaussian Splat simulation, WoMAP transfers effectively to the real world, achieving the same success rate in *GS-Office* and a higher success rate in *GS-Random* compared to the sim success rates, with only a moderate performance drop of 23% in *GS-Kitchen*. In contrast, the VLM baseline typically predicts unrealistic actions that violate joint limits, resulting in a substantial performance drop of 62% or more. This finding highlights WoMAP's strong generalization capabilities in producing reliable reward predictions under domain shifts, enabling efficient sim-to-real transfer.

E. Generalization to Novel Task Conditions

We examine the visual and semantic generalization capabilities of WoMAP trained only on nominal conditions in *GS-Random* on 10 scenes with 30 trajectories each. We evaluate WoMAP in out-of-distribution lighting and background conditions, illustrated in Figure 8. In Table II, we show that WoMAP achieves strong zero-shot generalization with a success rate and efficiency score of 50% and 47% compared to 63%

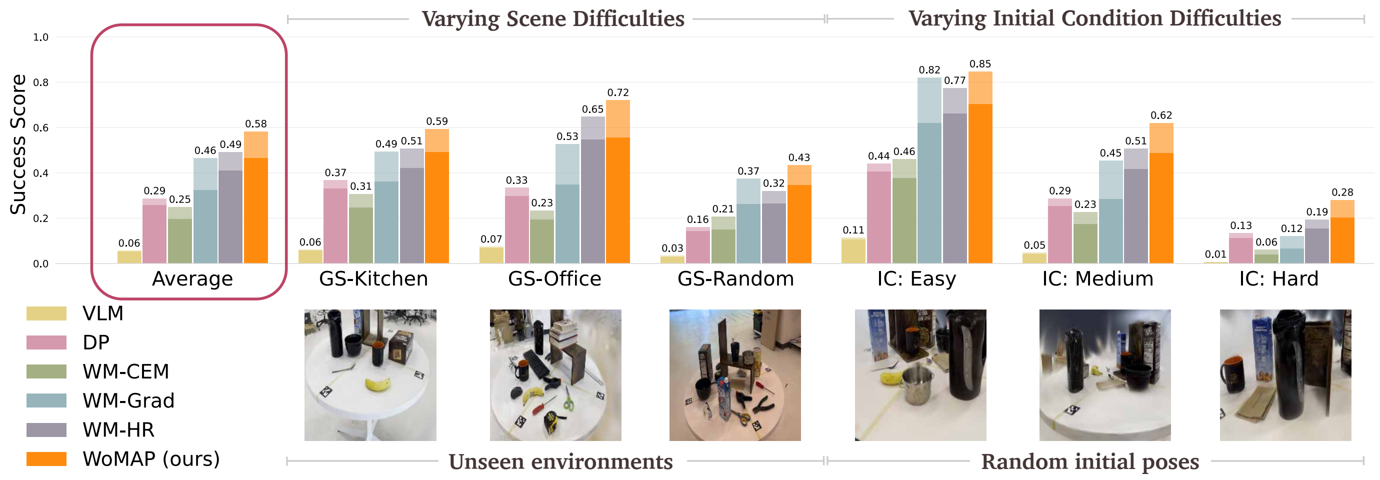

Fig. 7: **Gaussian Splat evaluation tasks and results.** Success rates (translucent bars) and efficiency scores (solid bars) in active object localization across Gaussian Splat scenes and initial-pose conditions: easy (E), medium (M), and hard (H). As in the PyBullet scenes, WoMAP outperforms all baseline methods via effective action grounding and optimization.

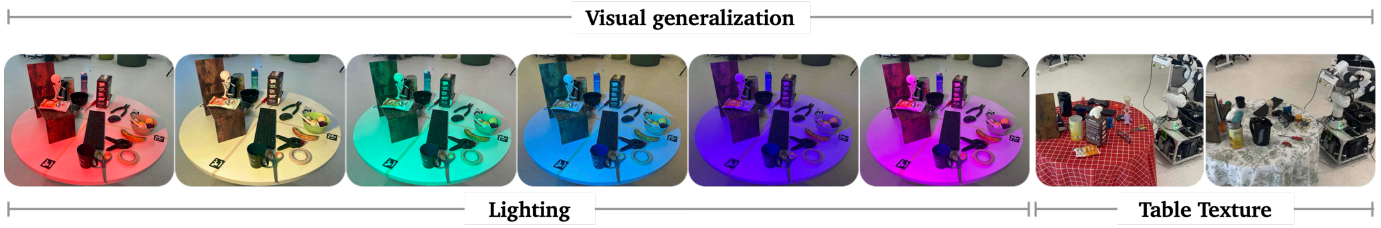

Fig. 8: Visual generalization: lighting and background conditions.

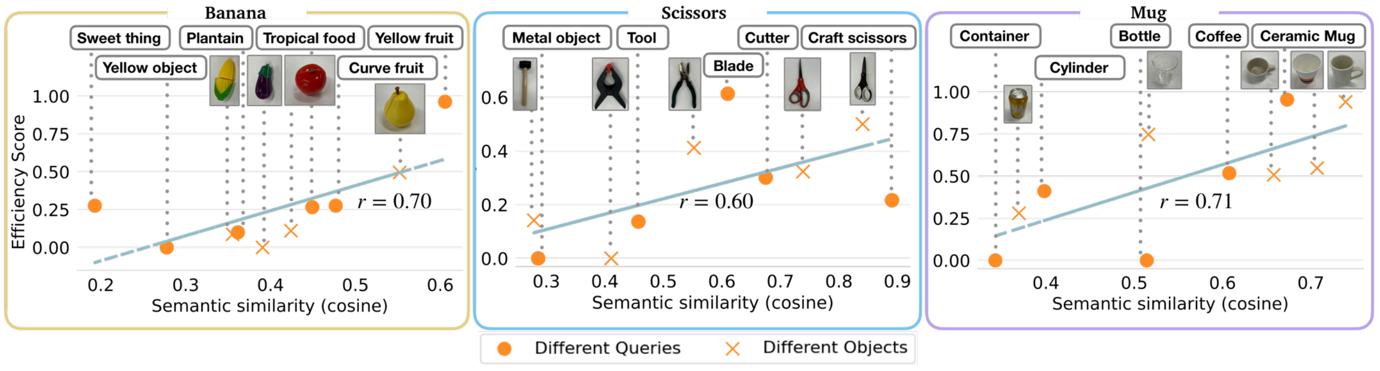

Fig. 9: Generalization plots for unseen queries and objects in the same category: (left) *banana*, (center) *scissors*, (right) *mug*. We see a positive correlation in semantic similarity (cosine distance) of the objects/queries with the most similar object present in our training objects, and the efficiency score suggesting the model's performance.

and 60% in nominal conditions, respectively (a decrease of less than 22%), even under extreme lighting conditions, with a further performance drop with out-of-distribution backgrounds. In general, these findings show that WoMAP learns robust latent-space features for generalization to out-of-distribution test-time conditions.

In addition, we evaluate semantic generalization of WoMAP to unseen target objects and task instructions across two axes: (i) to unseen language instructions to find target objects that were seen during training and (ii) to unseen target objects with unseen language instructions. As illustrated in Figure 9, we consider three representative object categories: "banana," "scissors," and "mug," where WoMAP is only trained on a single banana, scissors, and mug. We find that WoMAP achieves strong semantic generalization, with an expected decrease in performance with decreasing *semantic similarity* as measured by the the cosine similarity metric. For example, we ask WoMAP to find the following unseen objects: "pear," "pliers," and "beaker." Even though WoMAP has not seen these objects during training, WoMAP is able to find each of these objects at test time. We discuss these results further in Appendix B.

V. Conclusion

We present WoMAP, a recipe for open-vocabulary active object localization, that uses a scalable data generation pipeline to train a latent world model without expert demonstrations or online interaction data. WoMAP distills dense rewards signals into the world model with a reconstruction-free training architecture for strong generalization from a few training samples. At inference time, WoMAP utilizes the world model for dynamics and rewards prediction to ground high-level action proposals from VLMs, demonstrating more efficient object localization performance and strong generalization capabilities to novel task settings.

VI. Limitations and Future Work

Interactive Active Object Localization. Although we limit our problem to non-interactive object localization problems in this work, interaction between the robot and its environment is crucial to efficient exploration in many problem setting. Consequently, active object localization with interactive feedback from the environment is a promising direction for future research to enable more expressive and manipulation-intensive tasks.

Uncertainty Quantification in Rewards Distillation. WoMAP distills the confidence of pretrained object detectors into a world model as a rewards signal. However, learned object detectors sometimes produce uncalibrated confidence estimates, which could corrupt the training data, negatively impacting its effectiveness in grounding action proposals. Future work will explore calibration methods for pretrained object detectors to ensure data fidelity during training.

Hallucination Detection and Uncertainty Quantification in World Models. WoMAP's action optimization fails when the world model hallucinates the dynamics/rewards, usually in areas where the world model is not confident. Uncertainty quantification of world models remains critical to identifying when to trust predictions from world models, which has been relatively unexplored. In future work, we will derive calibrated uncertainty quantification methods to enable uncertainty-aware planning to ensure effective action grounding and optimization. In addition, we will explore incorporating calibrated uncertainty estimates from the VLM on the action proposals into WoMAP to enable risk-sensitive planning.

References

[1] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018.

[2] R. Bajcsy. Active perception. *Proceedings of the IEEE*, 76(8):966–1005, 1988. doi: 10.1109/5.5968.

[3] Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation world models. *arXiv preprint arXiv:2412.03572*, 2024.

[4] Maxime Burchi and Radu Timofte. Mudreamer: Learning predictive world models without reconstruction. *arXiv preprint arXiv:2405.15083*, 2024.

[5] Matthew Chang, Theophile Gervet, Mukul Khanna, Sriram Yenamandra, Dhruv Shah, So Yeon Min, Kavit Shah, Chris Paxton, Saurabh Gupta, Dhruv Batra, Roozbeh Mottaghi, Jitendra Malik, and Devendra Singh Chaplot. Goat: Go to any thing, 2023. URL https://arxiv.org/abs/2311.06430.

[6] Devendra Singh Chaplot, Dhiraj Prakashchand Gandhi, Abhinav Gupta, and Russ R Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. *Advances in Neural Information Processing Systems*, 33: 4247–4258, 2020.

[7] Erwin Coumans and Yunfei Bai. Pybullet, a Python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2022.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[9] Lei Fan, Mingfu Liang, Yunxuan Li, Gang Hua, and Ying Wu. Evidential active recognition: Intelligent and prudent open-world embodied perception, 2023. URL https://arxiv.org/abs/2311.13793.

[10] Theophile Gervet, Soumith Chintala, Dhruv Batra, Jitendra Malik, and Devendra Singh Chaplot. Navigating to objects in the real world. *Science Robotics*, 8(79): eadf6991, 2023.

[11] James J. Gibson. *The Ecological Approach to Visual Perception*. Houghton Mifflin, Boston, 1979.

[12] Dylan Goetting, Himanshu Gaurav Singh, and Antonio Loquercio. End-to-end navigation with vision language models: Transforming spatial reasoning into question-answering. *arXiv preprint arXiv:2411.05755*, 2024.

[13] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[14] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

[15] Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. *arXiv preprint arXiv:2203.04955*, 2022.

[16] Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. *arXiv preprint arXiv:2310.16828*, 2023.

[17] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[18] Hanxiao Jiang, Binghao Huang, Ruihai Wu, Zhuoran Li, Shubham Garg, Hooshang Nayyeri, Shenlong Wang, and Yunzhu Li. Roboexp: Action-conditioned scene graph via

interactive exploration for robotic manipulation. *arXiv preprint arXiv:2402.15487*, 2024.

[19] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4): 139–1, 2023.

[20] Steven M LaValle and James J Kuffner. Rapidly-exploring random trees: Progress and prospects: Steven m. lavalle, iowa state university, a james j. kuffner, jr., university of tokyo, tokyo, japan. *Algorithmic and computational robotics*, pages 303–307, 2001.

[21] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European Conference on Computer Vision*, pages 38–55. Springer, 2024.

[22] Russell Mendonca, Shikhar Bahl, and Deepak Pathak. Alan: Autonomously exploring robotic agents in the real world. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3044–3050. IEEE, 2023.

[23] Kensuke Nakamura, Lasse Peters, and Andrea Bajcsy. Generalizing safety beyond collision-avoidance via latent-space reachability analysis. *arXiv preprint arXiv:2502.00935*, 2025.

[24] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[25] Han Qi, Haocheng Yin, Yilun Du, and Heng Yang. Strengthening generative robot policies through predictive world modeling. *arXiv preprint arXiv:2502.00622*, 2025.

[26] Mohammad Nomaan Qureshi, Sparsh Garg, Francisco Yandun, David Held, George Kantor, and Abhisesh Silwal. Splatsim: Zero-shot sim2real transfer of rgb manipulation policies using gaussian splatting. *arXiv preprint arXiv:2409.10161*, 2024.

[27] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[28] Santhosh K. Ramakrishnan, Dinesh Jayaraman, and Kristen Grauman. Emergence of exploratory look-around behaviors through active observation completion. *Science Robotics*, 4(30), May 2019. ISSN 2470-9476. doi: 10.1126/scirobotics.aaw6326. URL http://dx.doi.org/10.1126/scirobotics.aaw6326.

[29] Ram Ramrakhya, Eric Undersander, Dhruv Batra, and Abhishek Das. Habitat-web: Learning embodied object-search strategies from human demonstrations at scale. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5173–5183, 2022.

[30] Ram Ramrakhya, Dhruv Batra, Erik Wijmans, and Abhishek Das. Pirlnav: Pretraining with imitation and rl finetuning for objectnav. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17896–17906, 2023.

[31] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL https://arxiv.org/abs/1908.10084.

[32] Allen Z Ren, Jaden Clark, Anushri Dixit, Masha Itkina, Anirudha Majumdar, and Dorsa Sadigh. Explore until confident: Efficient exploration for embodied question answering. *arXiv preprint arXiv:2403.15941*, 2024.

[33] Dhruv Shah, Michael Equi, Blazej Osinski, Fei Xia, Brian Ichter, and Sergey Levine. Navigation with large language models: Semantic guesswork as a heuristic for planning, 2023. URL https://arxiv.org/abs/2310.10103.

[34] Ola Shorinwa, Jiankai Sun, and Mac Schwager. Fast-splat: Fast, ambiguity-free semantics transfer in gaussian splatting. *arXiv preprint arXiv:2411.13753*, 2024.

[35] Ola Shorinwa, Johnathan Tucker, Aliyah Smith, Aiden Swann, Timothy Chen, Roya Firoozi, Monroe Kennedy III, and Mac Schwager. Splat-mover: Multi-stage, open-vocabulary robotic manipulation via editable gaussian splatting. *arXiv preprint arXiv:2405.04378*, 2024.

[36] Ola Shorinwa, Jiankai Sun, Mac Schwager, and Anirudha Majumdar. Siren: Semantic, initialization-free registration of multi-robot gaussian splatting maps. *arXiv preprint arXiv:2502.06519*, 2025.

[37] Ajay Sridhar, Dhruv Shah, Catherine Glossop, and Sergey Levine. Nomad: Goal masked diffusion policies for navigation and exploration. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 63–70. IEEE, 2024.

[38] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, et al. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 conference proceedings*, pages 1–12, 2023.

[39] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[40] Congcong Wen, Yisiyuan Huang, Hao Huang, Yanjia Huang, Shuaihang Yuan, Yu Hao, Hui Lin, Yu-Shen Liu, and Yi Fang. Zero-shot object navigation with vision-language models reasoning. In *International Conference on Pattern Recognition*, pages 389–404. Springer, 2025.

[41] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[42] Jimmy Wu, William Chong, Robert Holmberg, Aaditya

Prasad, Yihuai Gao, Oussama Khatib, Shuran Song, Szymon Rusinkiewicz, and Jeannette Bohg. Tidybot++: An open-source holonomic mobile manipulator for robot learning. *arXiv preprint arXiv:2412.10447*, 2024.

[43] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. Daydreamer: World models for physical robot learning. In *Conference on robot learning*, pages 2226–2240. PMLR, 2023.

[44] Joel Ye, Dhruv Batra, Abhishek Das, and Erik Wijmans. Auxiliary tasks and exploration enable objectgoal navigation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16117–16126, 2021.

[45] Naoki Yokoyama, Ram Ramrakhya, Abhishek Das, Dhruv Batra, and Sehoon Ha. Hm3d-ovon: A dataset and benchmark for open-vocabulary object goal navigation. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5543–5550. IEEE, 2024.

[46] Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

[47] Kaiwen Zhou, Kaizhi Zheng, Connor Pryor, Yilin Shen, Hongxia Jin, Lise Getoor, and Xin Eric Wang. Esc: Exploration with soft commonsense constraints for zero-shot object navigation. In *International Conference on Machine Learning*, pages 42829–42842. PMLR, 2023.

[48] Shijie Zhou, Haoran Chang, Sicheng Jiang, Zhiwen Fan, Zehao Zhu, Dejia Xu, Pradyumna Chari, Suya You, Zhangyang Wang, and Achuta Kadambi. Feature 3dgs: Supercharging 3d gaussian splatting to enable distilled feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21676–21685, 2024.

Appendix A
Implementation Details

A. Details on Data Generation

1) Preliminaries: Gaussian Splatting for Photorealistic Novel View Rendering: We provide a brief introduction to Gaussian Splatting and its applications to robotics. Gaussian Splatting [19] is a volumetric scene representation that uses explicit ellipsoidal primitives to represent non-empty space in any given environment. Trained entirely from RGB poses and associated camera poses, Gaussian Spatting enables real-time photorealistic novel-view synthesis without any structural priors, unlike many existing scene reconstruction methods. Given its high-fidelity reconstruction, novel-view synthesis, and amenability to open-vocabulary semantics, Gaussian Splatting has been widely applied in robotics, e.g., robot manipulation [35, 26]. Training a latent world model requires abundant data coverage of the environment, which is challenging to collect in real world. In this work, we leverage Gaussian Splatting for scalable data generation from only a few real-world videos. Specifically, we employ semantic Gaussian Splatting [48, 34] for automatic labeling of target objects, distilling language semantics from CLIP [27] into the Gaussian Splat. In the following subsections, we briefly describe the procedure used for generating, aligning, and rendering views for multiple scenes.

Collecting Videos and Training the Gaussian Splat. Trained Gaussian Splats do not share a common reference frame, in general. Hence, we align the individual Gaussian Splats using four Aruco markers in fixed positions. However, we note that other approaches such as semantics-based alignment can also be used. In each scene, we record a one-minute video as input to the Gaussian Splat. We compute the camera poses for each video using structure-from-motion and subsequently train the semantic 3D Gaussian Splat [36] for 30,000 iterations on an Nvidia L40 GPU using Nerfstudio [38].

Scene Alignment and Annotation. The reconstructed scene representation could have arbitrary reference frame. However, a common reference frame is necessary for data consistency when generating data from multiple scenes. Consequently, we first perform alignment of the scene coordinates by matching Aruco tags detected in the reconstruction with its measured ground-truth position and orientation. Finally, given coordinates of the detected points in world frame, we solve the Perspective-n-Point (PnP) problem and the point-registration problem using RANSAC to compute the camera-to-world and the Gaussian Splat-to-world transforms, respectively. We query the semantic field of the Gaussian Splat to annotate the position and dimension of each target object in the scene, visualized in Figure 10, with different target objects.

2) Ablation: Generating Training Data without Novel Views: We examine the need for data scalability beyond real-world data, ablating WoMAP limited to real-world video frames compared to the data generated from the Gaussian Splats. First, we extract a set of image frames uniformly across each real-world video and use these images in training the Gaussian Splat. Next, we train two world models with: (i) only the video frames of each scene (*Video-Only*) and (ii) on the rendered images from the Gaussian Splat (*GSplat-Data*). The training data of the Video-Only and GSplat-Data world models consist of about 2100 and 9000 images, respectively. We evaluate the trained model using the gradient-based planner WM-Grad and summarize our results in Table III. Across all scenes and initial conditions, the GSplat-Data model outperforms the Video-Only model by significant margins, ranging between 50% and 200%. The relatively poor performance of the Video-Only model can be explained by the lack of sufficient data coverage in the real-world videos, underscoring the importance of our scalable data pipeline. Our data generation pipeline provides not only additional training images, but also *diverse* viewpoints, which makes the GSplat-Data model more robust to initial conditions compared to the Video-Only model (sweeping from *easy* to *hard*).

3) Details on Trajectory Data Generation: To get sufficient coverage of diverse viewpoints in the the PyBullet environment and Gaussian Splat scenes, we generate synthetic collision-free trajectories that start from randomized initial positions and navigate toward various target locations. Specifically, we leverage the RRT* planner [20] to compute feasible, diverse paths between the start and goal. We concatenate these camera poses and add random linear and angular perturbations to further increase data diversity to cover a more realistic range of viewpoints encountered at deployment time. Figure 11 provides a visualization of trajectories generated in our training dataset. We collect observations at relatively low frequencies where the delta distance between consecutive observations is around 1-5cm. The largest model that we train contains about 10,000 observation-camera pose pairs for 500 trajectories (or 20 observations per trajectory), which is considerably smaller in scale than the training data used in many other imitation learning or reinforcement learning-based visual navigation policies [33, 37]. Moreover, we show in Appendix B-D1 that WoMAP's performance remains competitive in much smaller training configurations.

B. World Model Implementation

Here, we summarize the implementation details of the world model—composed of the observation encoder, dynamics predictor, and rewards predictor—and provide the hyperparameters used in training the world model. For interpretability, we train a decoder to map latent states to the image space without backpropagating the gradients through the other components of the world model. In addition, to better visualize the objects the world model is focusing on, we train the rewards models to predict bounding-boxes along with the scalar rewards, which we overlay in the decoded RGB images.

Observation Encoder. We encode raw image observations into the latent space using the pre-trained DINOv2 model *dinov2_vits14* [24]. We use the norm of the patch tokens of image o_t as its feature embedding $h_\theta(o_t) \in \mathbb{R}^{384}$. In addition, we freeze the weights of the DINOv2 model during training and do not apply any image data augmentation, e.g., color

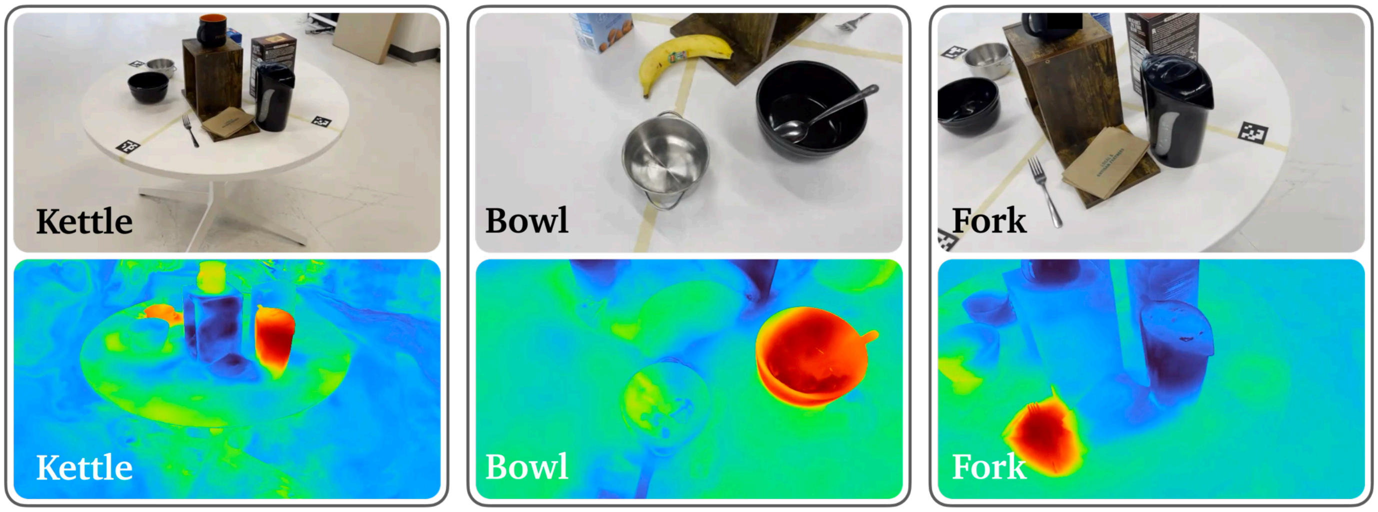

Fig. 10: Querying the semantic field of the Gaussian Splat.

TABLE III: Success score of the world model trained using only real-world video frames (Video-Only model) compared to the (GSplat-Data model). With more diverse data, the GSplat-Data model outperforms Video-Only model across scenes and initial conditions.

Training Data	**GS-Kitchen**			**GS-Office**			**GS-Random**		
	init-easy	init-medium	init-hard	init-easy	init-medium	init-hard	init-easy	init-medium	init-hard
Video-Only	0.06	0.04	0.02	0.24	0.06	0.02	0.04	0	0
GSplat-Data	0.84	0.50	0.14	0.86	0.50	0.22	0.76	0.36	0

jittering and random cropping, since the pre-trained DINOv2 model already utilizes data augmentation for training. Moreover, random perturbation of the image, such as random rotation or cropping, may compromise the fidelity of the ground-truth image-action pairs. In Appendix A-B1, we ablate the observation encoder.

Dynamics Predictor. To condition the ViT-based dynamics predictor on both the latent state z_t and action a_t, we map z_t and a_t to a 384-dimensional embedding space using an affine transformation and concatenate the resulting embeddings. In preliminary experiments, we found that a longer historical context for dynamics prediction did not provide any significant improvement in prediction accuracy. As a result, we provide only the last latent state to the dynamics predictor. To improve the multi-step prediction accuracy, we supervise the dynamics predictor over a sequence of observation-action pairs, recursively passing in the previous prediction into the model. Consequently, we do not optimize the dynamics predictor using teacher forcing [41]. Although teacher forcing facilitates faster training through parallelism, teacher forcing contributes to significant accumulation of dynamics errors over multi-step predictions.

Rewards Predictor. Like the dynamics predictor, we condition the rewards predictor on z_t and the language embedding e_g, each mapped to $\mathbb{R}^{384}$. We concatenate the embeddings and apply full cross-attention to predict the scalar reward for that latent state and target object. We train the rewards predictor using the binary cross-entropy loss function given by: $\ell_t = -[r_{\text{sp},t} \log(r_t) + (1 - r_{\text{sp},t}) \log(1 - r_t))]$, for datapoint $(r_{\text{sp},t}, r_t)$ with ground-truth reward $r_{\text{sp},t}$ and predicted reward r_t.

Training Setup and Hyperparameters. We train the world model on a single Nvidia L40 GPU with 48GB of GPU VRAM using a batch size of 25 for between 8 to 10 hours, depending on the task environment. In Table IV, we present the hyperparameters used in training the world model. We warmup training with a learning rate (LR) of $1e^{-3}$ before training for the full number of epochs (100) with an LR of $5e^{-4}$ for stable training. We observed that the training loss diverged for learning rates greater than $1e^{-3}$. Further, in Table V, we report the number of trainable and non-trainable parameters in each component of the world model. The dynamics q_ψ and rewards v_ϕ predictors are similarly-sized, with about 7.6 and 4.1 million trainable parameters, respectively. Meanwhile, we do not fine-tune the observation encoder h_θ.

1) Ablations: We ablate different components of the world model, examining the effects of training an encoder from scratch, fine-tuning a pre-trained encoder, and using an image reconstruction loss. We report our findings in Figure 12, 13, and 14, where *ViT-R* and *ViT-NR* denote ViT trained with image reconstruction and without image reconstruction, respectively, *DINO-Frozen* denote a frozen DINOv2 model, and *DINO-R-FT* and *DINO-NR-FT* denote a finetuned DINOv2 model trained with and without image reconstruction. In

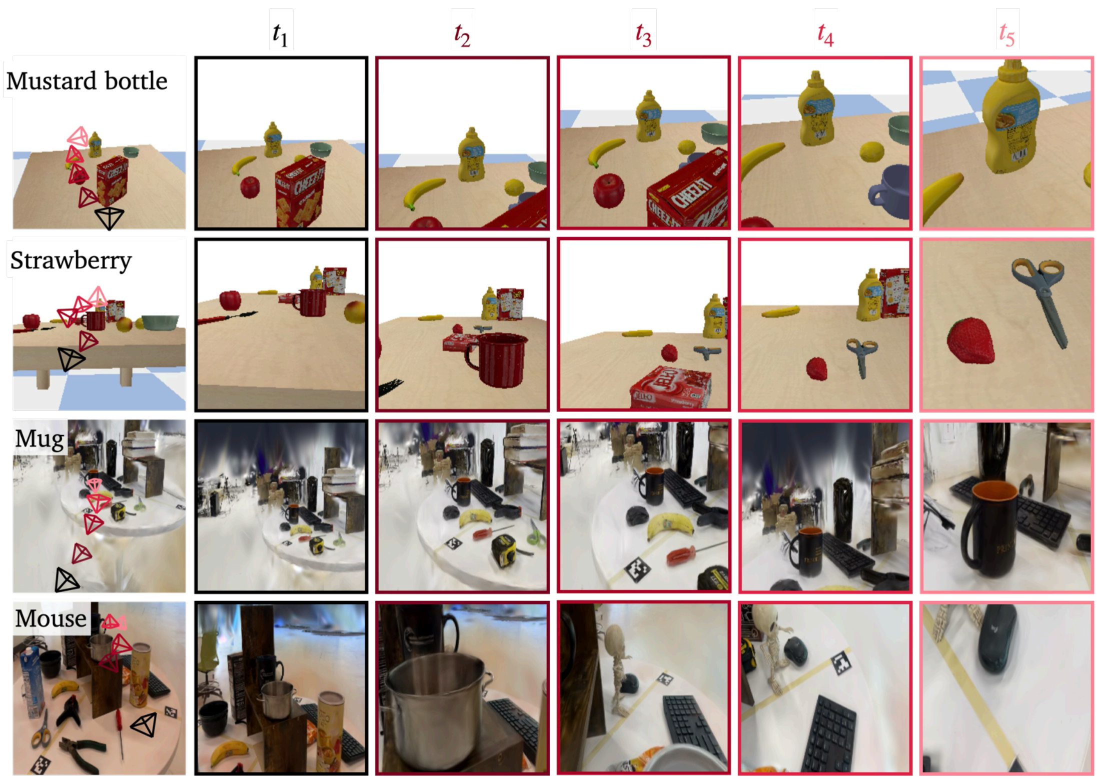

Fig. 11: Visualization of training trajectories generated in PyBullet and Gaussian Splat.

TABLE IV: WoMAP's Hyperparameters.

Name	Value
Image Size	224
Lang. Embed Dim	384
Pred. Embed Dim	384
Start LR	1e-3
LR	5e-4
Warmup Epochs	2
Weight Decay	4e-2
Final Weight Decay	0.4
Batch Size	25
Total Epoch	100
Planning Horizon	4

TABLE V: WoMAP's Number of Parameters (in millions).

Name	# trainable	# non-trainable
h_θ	0	22
q_ψ	7.6	0.1
v_ϕ	4.1	0.1

all applicable plots, we represent the success rate of each method by the solid-color bars and the efficiency scores by the translucent bars. We discuss these results in the following subsections.

Training the Observation Encoder from Scratch. We compare a ViT-based observation encoder trained from scratch (ViT-NR) to a frozen pre-trained DINOv2 model (DINO-NR-FZ), without an image reconstruction objective. From Figure 12 and 13, DINO-NR-FZ achieves higher success rates and efficiency scores across the PyBullet and GSplat scenes. Although both models have access to the same data when training the world model, the results suggest that DINO-NR-FZ model benefits from large-scale pre-training, which provides a robust latent state for dynamics and rewards prediction even without any fine-tuning. Likewise, when trained with an image reconstruction loss, DINO-R-FZ also outperforms ViT-R for similar reasons. Further, we observed that the ViT was more unstable to train, given the total number of trainable parameters. In general, the ViT-NR and ViT-R models may require more training data to learn more useful visual features compared to the frozen DINOv2 models.

Finetuning the Pre-trained Observation Encoder. We explore fine-tuning the DINOv2 encoder, comparing its performance to that of the frozen model. We find that fine-tuning the DINOv2 encoder leads to training instability that adversely impacts the performance of the world model. In fact, in many of our experiments, the training loss failed to decrease or raised Not-a-Number (NaN) errors. In Figure 14, we show the training loss for the dynamics predictor across the four PyBullet environment, highlighting the increase in the training loss at the initial stages of the training procedure in the fine-tuned DINOv2 model. This training instability may be attributed to the more complicated loss landscape with many local minima during fine-tuning. In contrast, the training loss for the frozen DINOv2 models decreases relatively monotonically. These training dynamics are reflected in the success rates and efficiency scores achieved by both models. The fine-tuned models DINO-NR-FT and DINO-R-FT have notably lower

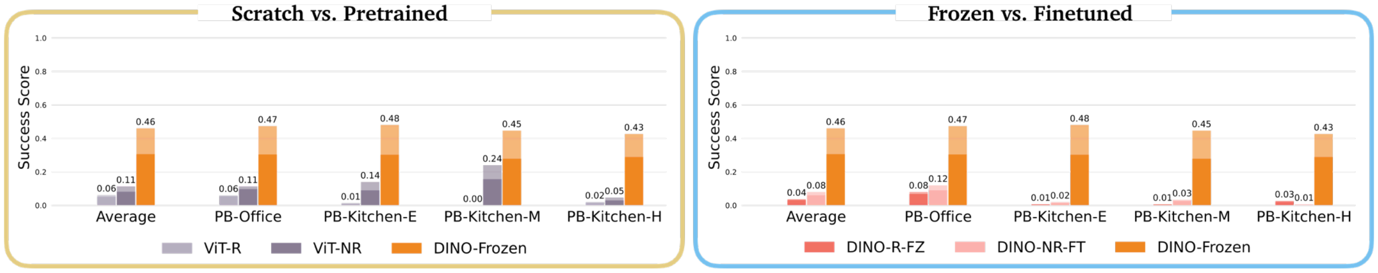

Fig. 12: **World Model Architecture Ablations in the PyBullet Scenes.** We explore training the observation encoder from scratch, finetuning and training the observation encoder with image reconstruction, where *ViT-R* denotes a ViT **with** image reconstruction, *ViT-NR* denote a ViT trained **without** image reconstruction, respectively, *DINO-Frozen* denotes a **frozen** DINOv2 encoder, *DINO-R-FT* denotes a finetuned DINOv2 model trained **with** image reconstruction, and *DINO-NR-FT* denotes a **finetuned** DINOv2 encoder trained **without** image reconstruction. WoMAP uses DINO-Frozen.

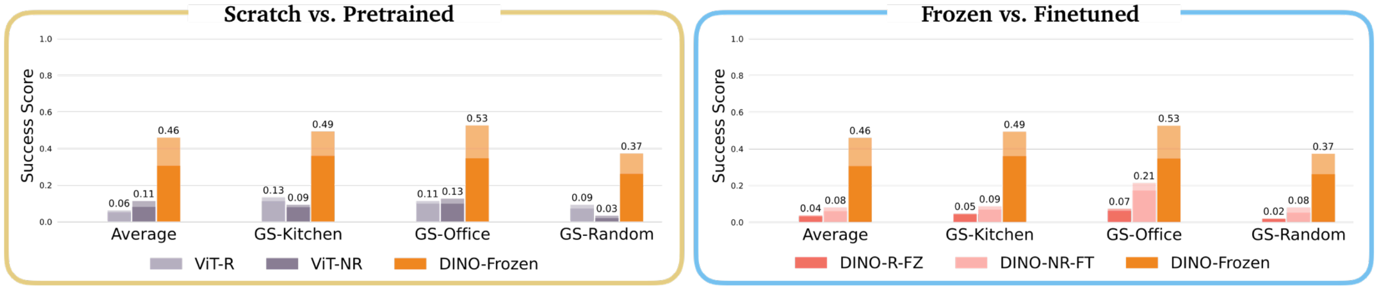

Fig. 13: **World Model Architecture Ablations in the Gaussian Splatting Scenes.** We ablate training the observation encoder from scratch, finetuning and training the observation encoder with image reconstruction, where *ViT-R* denotes a ViT **with** image reconstruction, *ViT-NR* denote a ViT trained **without** image reconstruction, respectively, *DINO-Frozen* denotes a **frozen** DINOv2 encoder, *DINO-R-FT* denotes a finetuned DINOv2 model trained **with** image reconstruction, and *DINO-NR-FT* denotes a **finetuned** DINOv2 encoder trained **without** image reconstruction. WoMAP uses DINO-Frozen.

scores on the performance metrics compared to frozen DINO encoder *DINO-Frozen*.

Training with a Reconstruction Objective. Here, we investigate training the world model with an image reconstruction objective. From Figures 12 and 13, we find that training with an image reconstruction objective generally leads to a degradation in the performance of the world model, e.g., in the ViT-R and DINO-R-FT models. Although image reconstruction objectives can provide dense rewards supervision, training instability often eliminates this potential advantage, underscoring the challenge with image reconstruction-based training. In some cases, the training instability can lead to significant drops in performance, e.g., in the ViT-R model when trained in the PB-Kitchen-M scene. However, with a non-frozen observation encoder, training with an image reconstruction loss degrades the performance of the world model, e.g., in the ViT-R and DINO-R-FT models. This finding underscores the challenge with image reconstruction-based training. In some cases, the training instability can lead to significant drops in performance, e.g., in the ViT-R model when trained in the PB-Kitchen-M scene. In summary, the results from the ablations indicate that the frozen pretrained DINOv2 provides generalizable latent features that enable accurate dynamics and rewards prediction and effective action grounding.

C. *Details on Planning Integration*

1) Planning with VLMs: Using the world model as a local planner poses many challenges given the noisy reward landscape, particularly in the low training data regime. However, world models can serve as a good evaluator/optimizer on imperfect action proposals generated from another policy. For example, generating good 6D action proposals with VLMs is difficult due to the limited spatial understanding ability of VLMs, a challenge that can be addressed using a world model. We experimented with the following prompting strategies using GPT-4o [17]:

Direct 6D action output. We first provide rough dimensions of the scene and ask the VLM to directly output 6D actions (x, y, z, roll, pitch, yaw). However, due to inconsistencies in coordinate system conventions across its pretraining data (e.g., variations in the orientation of the positive x and y axes), the vision-language model often fails to consistently interpret spatial directions and rotations correctly.

Direction output. To minimize confusion on the coordinate axes definition, we experimented with expressing the translation and rotation actions with natural language descriptions, such as move forwards/backwards, tilt up/down, etc. However, we have found that these directions are not fully descriptive of exploration behavior. For example, if the VLM suggests looking behind an obstacle that is directly in front, its action outputs are often axis-aligned, which is not expressive enough to encode

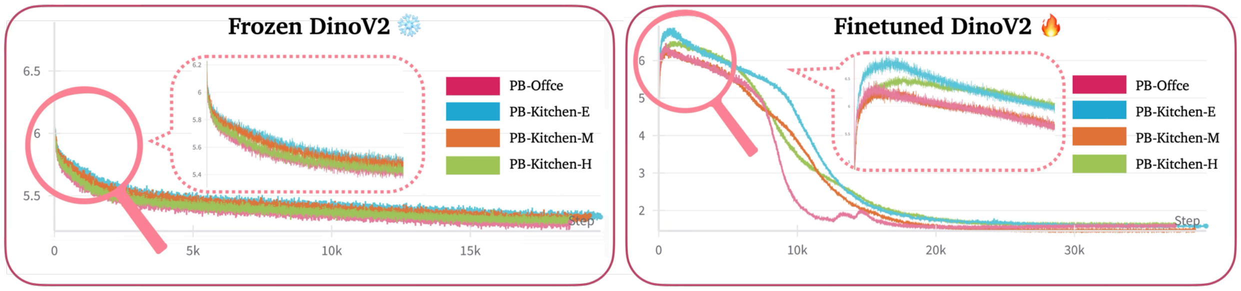

Fig. 14: **Frozen vs. Finetuned DINOv2 Encoder.** Fine-tuning the DINOv2 encoder generally leads to training instability, negatively impacting performance.

this "look around" behavior.

Action Primitives. Our final version frames the prompt as a multiple-choice question, providing the VLM with a list of action primitives. Empirically, we find that using the coarse action primitives as shown in Fig. 15, the VLM is capable of matching its high-level suggestions with the correct action outputs consistently. Despite covering only a limited set of actions with coarse magnitudes, we demonstrate that WoMAP's action optimization capability can still successfully transform these action proposals into grounded actions, as shown in **??**.

Appendix B
Experiment Details

A. Task Design Details

We provide more details on the design choices for the tasks we examine WoMAP on.

1) Designing task environments: As mentioned in Section IV-A, we evaluate WoMAP on four simulation environments and three real environments. Figures 6 and 7 show example scenes for each environment, and Figure 16 shows the different variation for a single environment across different scenes.

Our key motivation to selecting the suite of environments and objects lies on two axes: practicality and difficulty. We design each task based on a particular theme resembling a cluttered living area where active object localization is a a core challenge requiring spatial reasoning, viewpoint planning, and the ability to handle occlusions. For each environment, we pick representative objects along the axis of difficulty and try to cover a diverse set of objects. For *easy* scenes, we pick distinct objects, which are typically small in size, whereas for difficult scenes, we place large shelves and visually similar objects (e.g., apple and peach), creating occlusions and distractions. We provide a comprehensive list of environments and target objects in each environment in Table VI.

2) Evaluating Task Difficulty: To systematically evaluate our framework under various settings, we define task difficulty across two dimensions: *scene difficulty*, which is determined by the diversity of objects present and the level of compactness, and *initial-pose difficulty*, which is determined by a heuristics function that is computed from three factors: initial detection confidence d, ground truth distance to target d, and level of occlusion of the bounding box. The thresholds for these factors are environment-dependent to account for scene dimensions, detection qualities, etc. Though the difficulty metrics are not strictly quantifiable, from empirical results we observe consistent trends in degrading performance as task difficulty increases, and show promising trend on WoMAP's robustness to exhibiting less performance degradation compared to the baselines. Figure 17 and 18 show a more comprehensive visualization of different initial condition levels in two selected PyBullet and Gaussian Splat environments to offer readers a better intuition.

B. Hardware Experiment Setup

We evaluate our policy trained entirely in the Gaussian Splat directly on a TidyBot [42] platform with a Panda Franka arm. With a mobile base and a 6-DOF arm equipped with an onboard Intel RealSense camera, the TidyBot is well suited for active object localization tasks since it has a larger effective workspace, compared to tabletop, fixed-base Franka robots that are constrained to mostly top-down views. We interface with the TidyBot through an onboard NUC, which publishes images to a desktop for inference using the world model or VLM.

C. Evaluation Setup

1) Choice of Baselines and Ablations: In the following sections, we discuss in more detail how we setup the evaluation framework, and motivation for the choice of baseline and ablations that we choose to include. Selecting the proper baseline is particularly challenging in our case. For one, many policies are not open-vocabulary and requires more constrained problem/action spaces. Secondly, the setup of our task (large environment variations and observations from only an on-board camera) also makes fine-tuning state-of-the-art manipulation policies difficult, since they are not designed for the task. In our experiments, we compare WoMAP to a VLM-based planner with GPT-4o [17] as the VLM. We compute the gradients during planning using automatic differentiation in the WM-Random, WM-HR, and WoMAP.

Diffusion Policy. One might expect the diffusion policy to perform better in our experiments. Upon further investigation, we observed that *single-task* diffusion policy (trained to localize a single object in an environment) performs well; however, the *multi-task* diffusion policy (DP), which is more relevant to the active object localization problem, failed to perform well. We found that the DP generally moved forward in the

```
This is what you currently see. Please carefully analyze the current observation
and think about what is the best action to take given what you see.
If you think the current observation is a good enough view of {self.target}
and you can’t get any closer, please say ‘DONE’.
Otherwise, please select the top {self.k} choices from the action options
below that you think would help you achieve the goal given the current observation.
The options are:
    (A) Move directly forward for 15 cm -- this lets you approach the objects in view
    (B) Move directly to the left for 15 cm -- this expands your left left by a bit
    (C) Move directly to the right for 15 cm -- this expands your right left by a bit
    (D) Look to the left by 45 degrees -- this lets you look around
    (E) Look to the right for 45 degrees -- this lets you look around
    (F) Move forward-left for 15 cm -- this lets you approach objects on the
    left side of your view
    (G) Move forward-right for 15 cm -- this lets you approach objects on the
    right side of your view
    (H) Move forward-left for 15 cm and then look right by 45 degrees -- this lets
    you look behind an object
    (I) Move forward-right for 15 cm and then look left by 45 degrees -- this lets
    you look behind an object
All these actions should be executed relative to the current your position.
Please think carefully step by step and reason about why your choice can help you
get the desired observation.
Please also review your movement history to see where you’ve already explored.
Output the results in a structured JSON format as follows:
{
    "descriptions": <what you observe in the the current scene and
    whether there are hints of the {self.target}.>
    "actions": [
        {"rank": 1, "choice": <choice1>, "confidence": <confidence_score1>,
        "explanation": <explanation1>},
        ...
        {"rank": {self.k}, "choice": <choice{self.k}>, "confidence":
        <confidence_score{self.k}>, "explanation": <explanation{self.k}>},
    ]
}
<choice1>, ... <choice{self.k}> should be a single letter representing one of the
7 choices above.
Ensure the confidence scores are in descending order.
Do not include any extra explanation outside of the JSON structure.
```

Fig. 15: Prompt provided to the Vision-Language Model

direction the robot was initialized at, without exhibiting any intelligent exploration behavior, e.g., looking behind objects and in occluded areas. We visualize some of the DP trajectories in Figure 19 in the PB-Kitchen-Easy and PB-Kitchen-Hard scenes. The arrows in the figure indicate the direction of travel. From Figure 19, we see that the DP does not seem to make intentional decisions to move towards the target objects, annotated in the figure.

2) Choice of Metrics: We make two key observations when designing the evaluation metrics. First, in order to make sure that the final observation is of good quality, our *success score* is defined to be 1 if both (i) the detection confidence labeled by GroundingDINO is above a certain threshold, and (ii) the labeled bounding box proportion is above a certain threshold. Since GroundingDINO's detection confidence and bounding box size on different objects can behave differently depending on the object identity, shape, or size, we choose the object-specific scaling parameter from the best view.

Secondly, the quality of task completion should also be dependent on the amount of distance traveled to reach the target as some policies are more efficient than others. To this end, we define the *efficiency score* as efficiency $= r * \exp(-d/d^\star)$, where r denotes the success rate, d denotes the distance traveled, and $d^\star$ the optimal distance to the object. In practice, we use an estimate of $d^\star$.

D. Additional Experiment Results

1) Ablation Studies on Training Data: Though we performed all of our experiments with a fixed training setup (50 scenes-500 trajectories for the PyBullet environments and 10 scenes-300 trajectories for the Gaussian Splat/Real environments), we conducted additional ablation studies to investigate the influence of the size and diversity of the training data on model performance. We perform this ablation study in the PB-Kitchen-Easy scene, since we can easily vary the size and diversity of the training data in PyBullet.

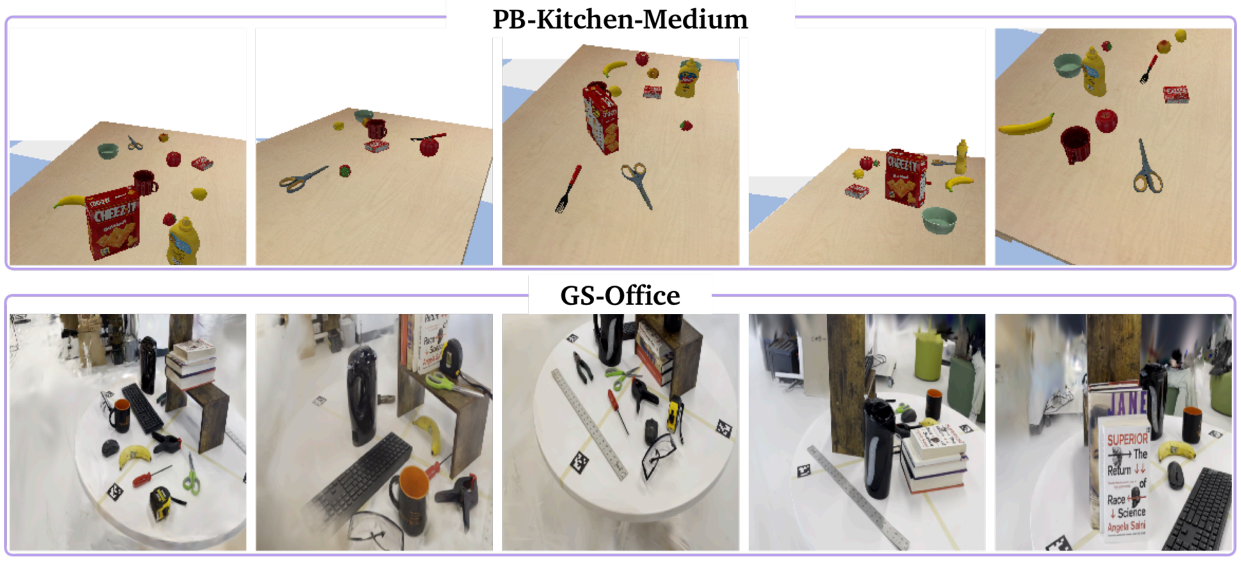

Fig. 16: Scene variations for a single environment. We create multiple scenes within each environment by varying the configuration and degree of occlusions of the objects in the environment. We show a few scenes in the PB-Kitchen-Medium and GS-Office environments.

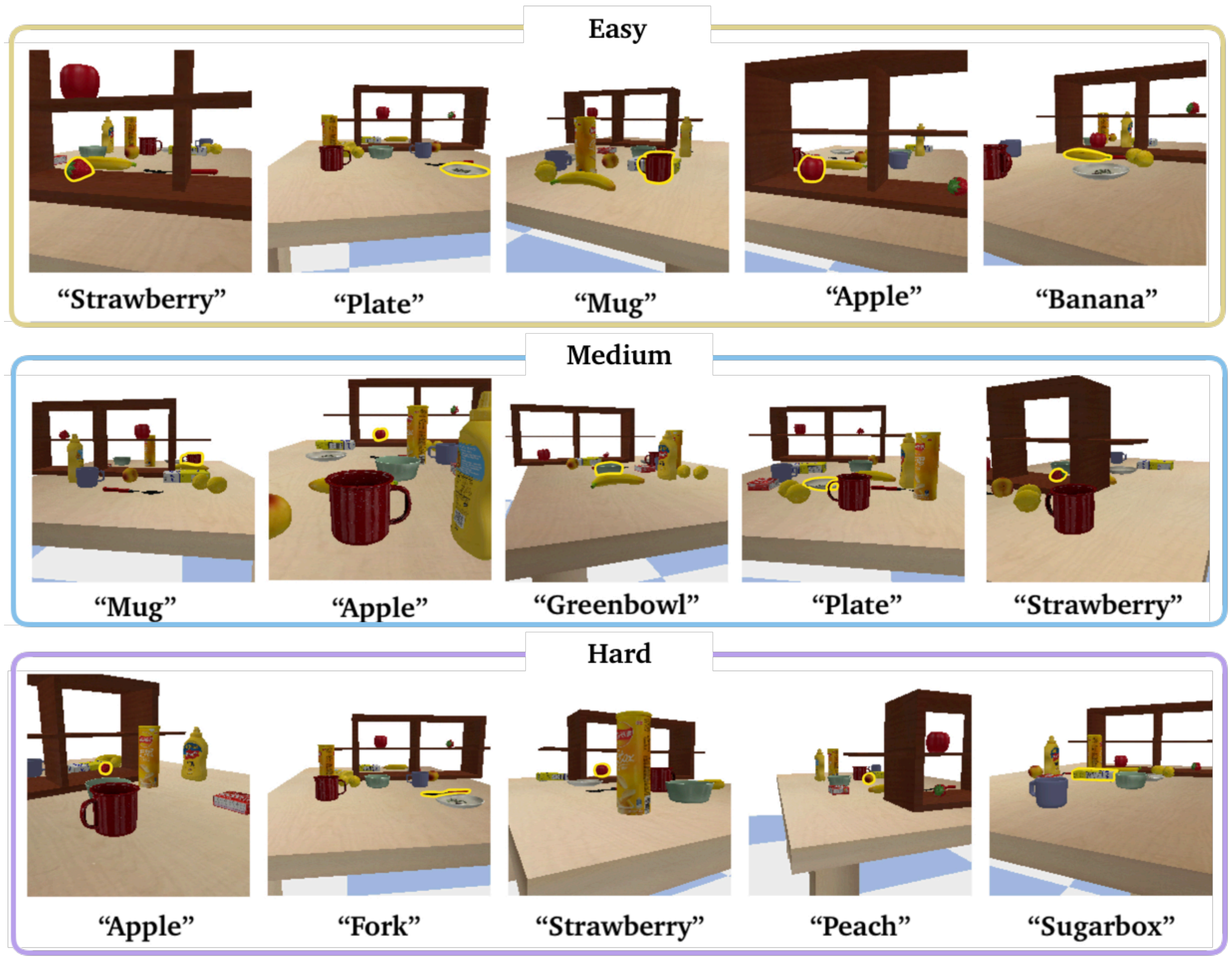

Fig. 17: **PB-Kitchen-Hard initialization difficulty.** Initial observations and target query for different initial-pose difficulty levels in the PB-Kitchen-Hard scene.

TABLE VI: Target objects used for each environments.

PB-Kitchen-Easy	PB-Kitchen-Medium	PB-Kitchen-Hard	PB-Office	GS-Kitchen	GS-Office	GS-Random
banana	banana	banana	gum	banana	banana	banana
apple	apple	apple	lipton tea	mug	mug	mug
green bowl	green bowl	green bowl	small marker	bowl	scissors	bowl
lemon	lemon	lemon	glue	fork	books	fork
mustard	mustard	mustard	book	pot	keyboard	keyboard
cracker box	cracker box	sugar box	stapler	kettle	mouse	mouse
blue cup	mug	mug	mug		screwdriver	screwdriver
	scissors	scissors	scissors		eyeglass	eyeglasses
	peach	peach	remote			pot
	fork	fork	cleanser			scissors
	strawberry	strawberry	potato chip			milk box
	jello box	jello box				skeleton
		plate				
		potato chip				

In Figure 20, we observe that the performance of WoMAP and WM-Grad increases as the total number of trajectories in the training dataset increases. We show the performance of WM-Grad alongside WoMAP to indicate the base performance of the world model without VLM action proposals. As expected, WoMAP's performance is strongly correlated with the size of the training data, which influences the dynamics and rewards prediction accuracy of the world model. Notably, even with only 200 training trajectories, WoMAP outperforms the VLM and DP baselines by about 1033% and 17%, respectively, (see Figure 6), showing its remarkable data efficiency.

Figure 21 shows the performance of WoMAP as we vary the scene diversity while keeping the total number of trajectories fixed. Overall, we do not observe a strong correlation between the number of training scenes and the success score. However, when training with fewer trajectories (e.g., 100 or 200 trajectories), the performance of the models decreases with the number of scenes. This finding may be attributed to the tradeoff between learning more specialized (scene-specific) features versus more generalizable features across multiple scenes, with a tight budget on the number of training trajectories. In contrast, when training with 500 trajectories, we find that the success rate improves as the number of scene increases, resulting in much higher success rates.

2) Full Experiment Results: We provide the full results for the experiments used to compute the values in Figure **??** and **??** in Table VII below for reference. Each experiment under a given environment and initialization difficulty is performed under 50 independent trials with randomized scene configuration and initial position.

3) Semantic Generalization Experiment Results: As discussed in Section IV-E, we evaluate WoMAP's semantic zero-shot generalization to novel tasks. We quantify the semantic similarity between different tasks using the cosine similarity between language embeddings computed from the task instructions by Sentence-Bert [31]. For seen target objects, we vary the task instructions to capture the breadth of possible user descriptions, e.g., asking WoMAP to find a "sweet thing" or a "yellow fruit" with the goal of locating a banana. We utilize partial success scores to better evaluate the degradation in performance with more semantically dissimilar target object queries, where the success scores is linearly scaled based on the bounding-box proportion and the detection confidence. From Figure 9, we find that WoMAP achieves strong generalization with a correlation coefficient between 0.6 and 0.71 across the three object categories. Likewise in Figure 9, we observe that WoMAP generalizes well to unseen target objects. For example, WoMAP localizes a pair of "pliers" and a "hammer," which WoMAP was not trained on. We attribute the strong generalization performance of WoMAP to the generalizable semantic features captured in WoMAP's latent space.

4) Additional Discussion: Here, we provide additional discussion for main results that are touched upon in Section IV.

World Model-Heuristics (WM-HR) Baseline: We use the set of candidate actions provided to the VLM as the heuristic actions in the W-HR baseline. This clever set of actions enabled the WM-HR to perform similarly to WoMAP, since the world model performed well at evaluating these candidate actions and selecting the most promising one. However, the WM-HR baseline lacks any high-level intelligence, posing a limitation in practical situations, since a brute-force approach would not scale, in general.

Sim-to-Real Performance for VLM: We observe that the performance of the VLM drops significantly when moving from simulation to the real-world. This drop in performance is in part due to the limitations in directly executing the VLM's action proposals on the physical robot. We find that the robot reaches its range limits often when evaluating the VLM, which effectively ends the experiments, leading to an increase in the failure of the VLM. Further, the VLM occasionally struggled with fully approaching the object in the real-world and often stopped a good distance away from the object.

5) Extension: Non-tabletop Scenes: Despite using tabletop scenes for easy standardized benchmarking in this paper, we also illustrate that our framework works for more general environments. We provide a video in the supplementary material, demonstrating WoMAP on a larger scale, particularly in a living room.

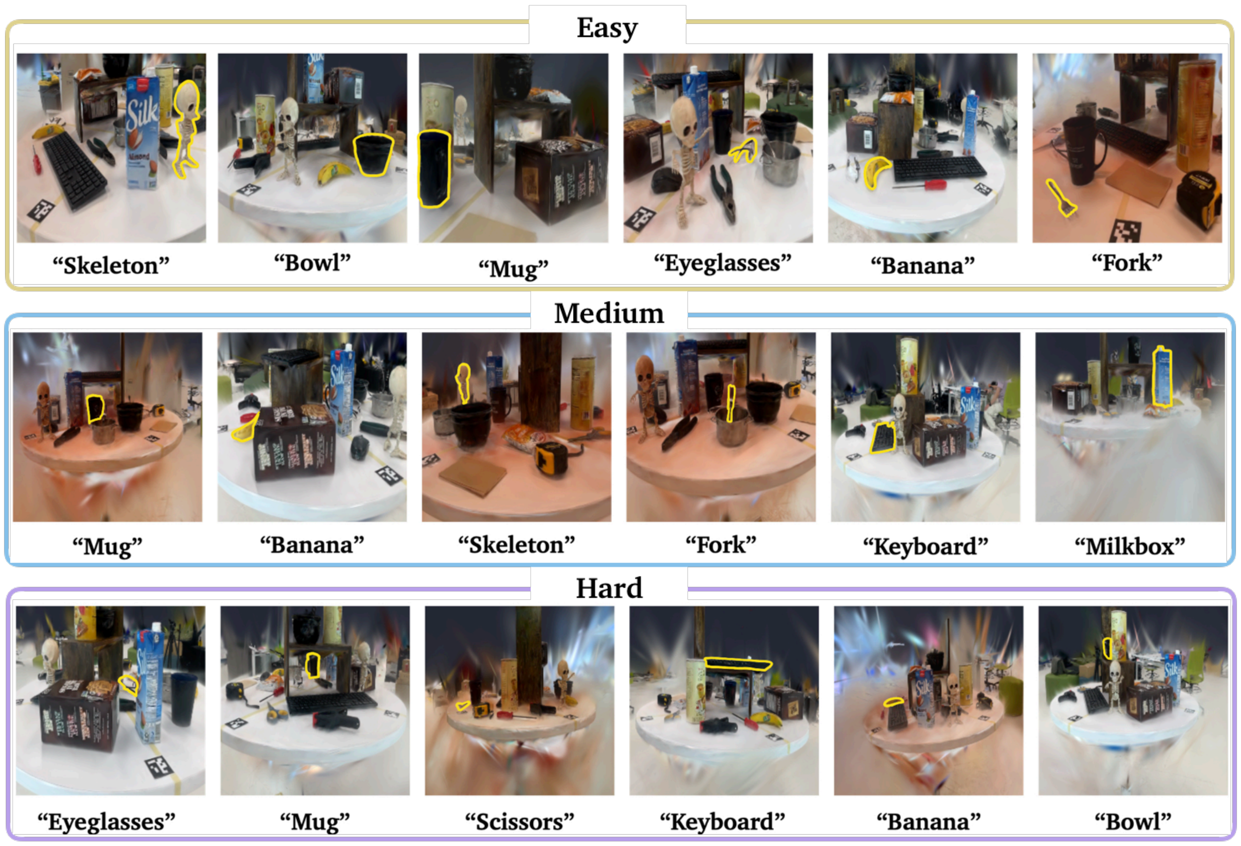

Fig. 18: **GS-Random initialization difficulty.** Initial observations and target query for different initial-pose difficulty levels in the GS-Random scene.

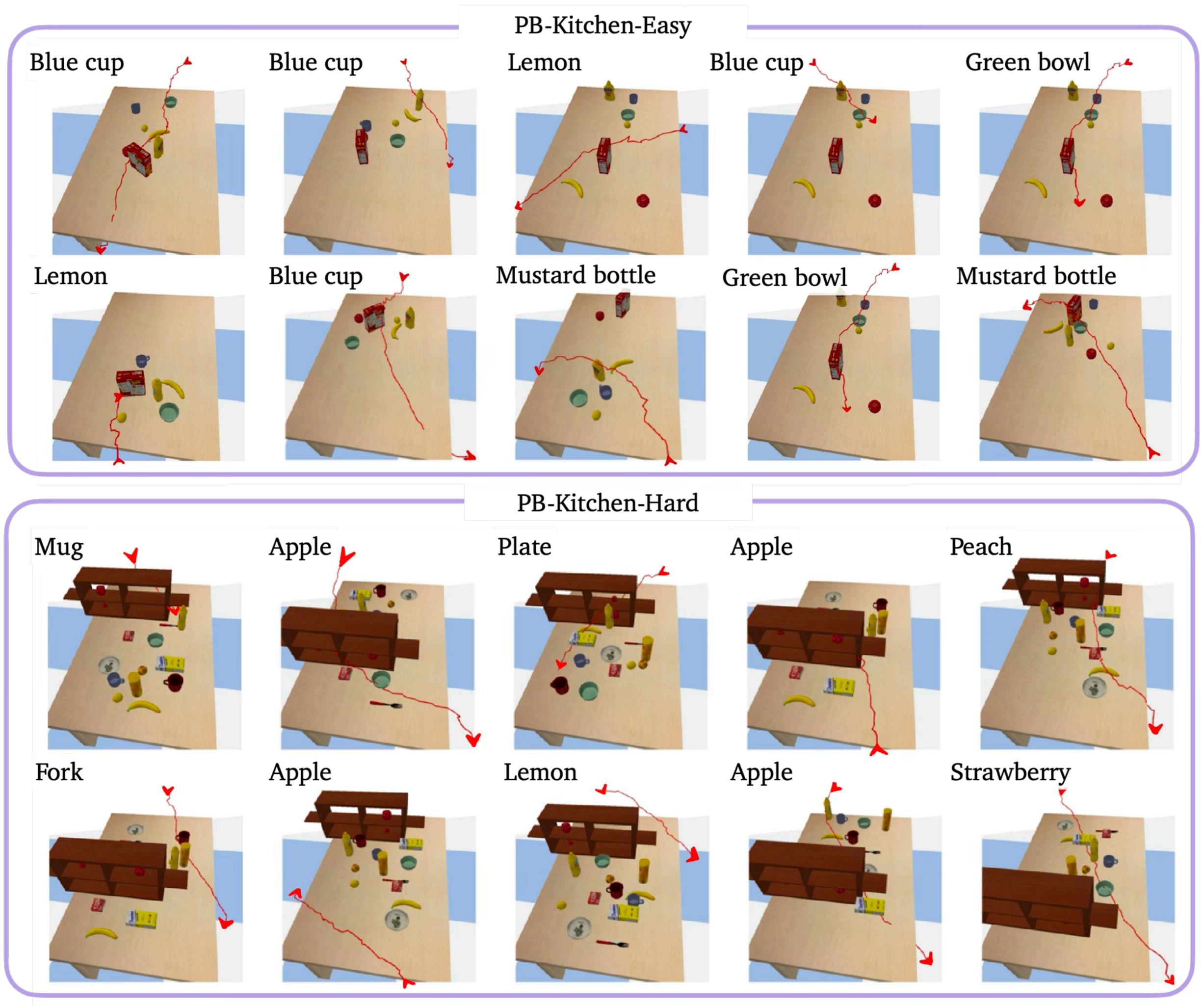

Fig. 19: Diffusion policy trajectory visualizations. Top two rows: PB-Kitchen-Easy with easy initial conditions, Bottom two rows: PB-Kitchen-Hard with hard initial conditions. The DP generally moved forward in the direction the robot was initialized at, without showing any intelligent exploration behavior, e.g., looking behind objects.

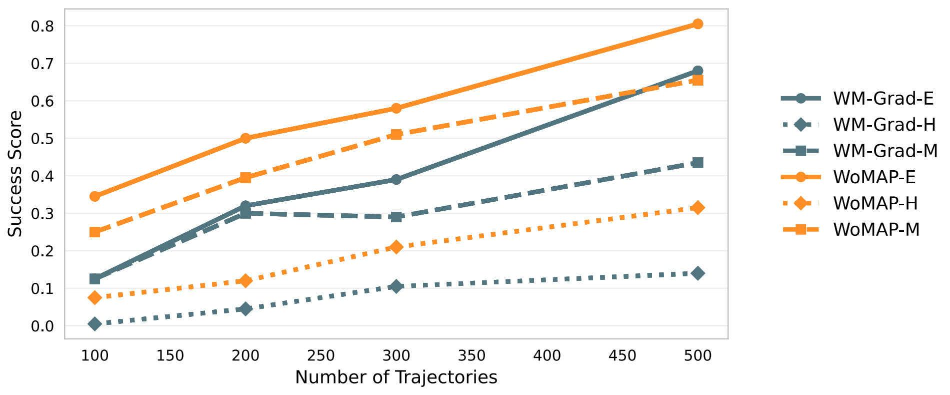

Fig. 20: Average success score for different number of trajectories sampled from 50 scenes in the PB-Kitchen-Easy task. We observe a positive correlation between the number of training trajectories and the success rate of WM-Grad and WoMAP across different initial conditions: Easy (E), Medium (M), and Hard (H). With only 200 training trajectories, WoMAP outperforms the VLM and DP baselines (see Figure 6).

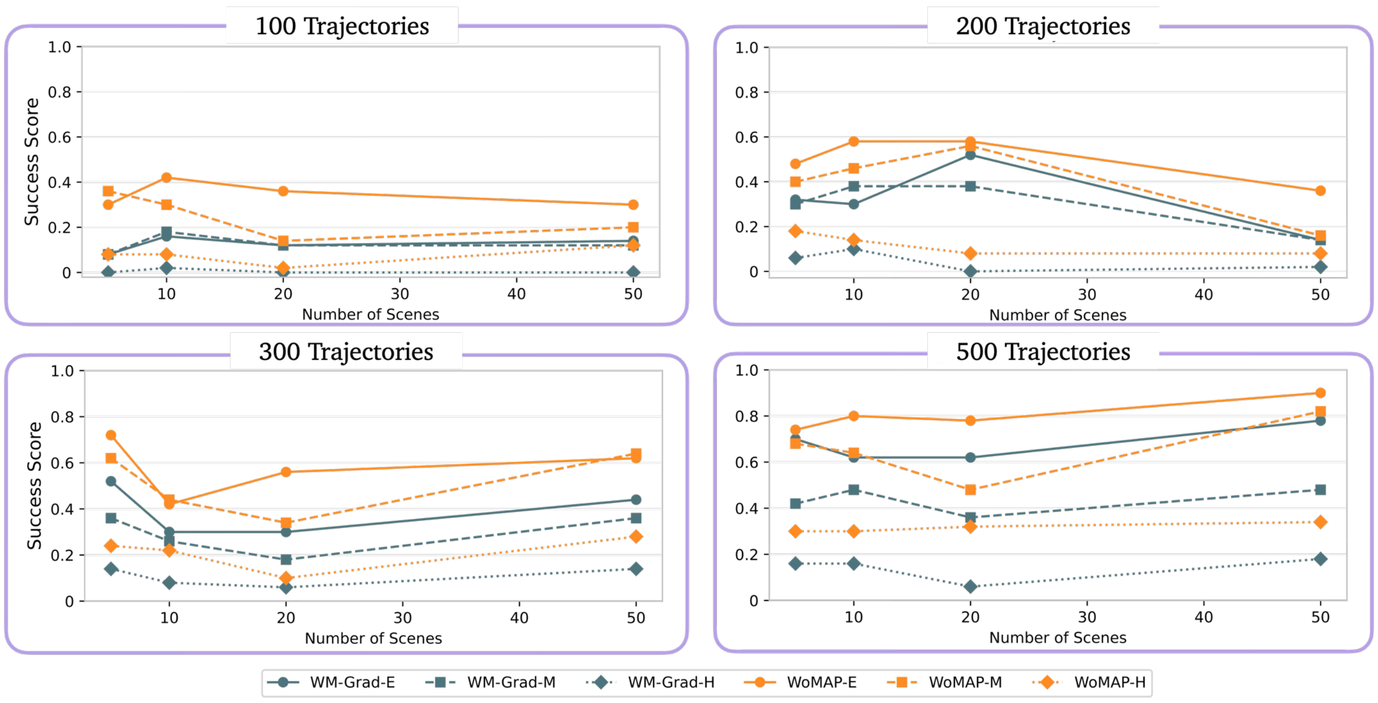

Fig. 21: Average success score for total number of trajectories $\in \{100, 200, 300, 400\}$ sampled from different numbers of scenes in the PB-Kitchen-Easy task. When trained with very few trajectories (e.g., 100 or 200 trajectories), the model performance decreases when the number of training scenes increases, due to the tradeoff between learning scene-specific features versus generalizable features. However, we observe a positive correlation between the number of training scenes and the success rate when training with more trajectories, e.g., 500 trajectories.

TABLE VII: Full Experiment Results

IC: Easy	**PB-Office**		**PB-Kitchen-Easy**		**PB-Kitchen-Medium**		**PB-Kitchen-Hard**	
Planner	Success	Efficiency	Success	Efficiency	Success	Efficiency	Success	Efficiency
VLM	0.30	0.28	0.06	0.05	0.00	0.00	0.02	0.02
DP	0.28	0.25	0.28	0.25	0.22	0.19	0.16	0.15
WM-CEM	0.30	0.24	0.18	0.13	0.10	0.08	0.16	0.13
WM-Grad	0.60	0.41	0.78	0.50	0.66	0.41	0.66	0.46
WM-HR	0.74	0.63	0.74	0.61	0.68	0.56	0.62	0.54
WoMAP	0.88	0.73	0.90	0.72	0.78	0.60	0.80	0.68

IC: Medium	**PB-Office**		**PB-Kitchen-Easy**		**PB-Kitchen-Medium**		**PB-Kitchen-Hard**	
	Success	Efficiency	Success	Efficiency	Success	Efficiency	Success	Efficiency
VLM	0.24	0.22	0.04	0.04	0.04	0.04	0.02	0.02
DP	0.42	0.36	0.26	0.23	0.28	0.25	0.08	0.07
WM-CEM	0.26	0.22	0.24	0.17	0.24	0.18	0.16	0.11
WM-Grad	0.58	0.39	0.48	0.30	0.52	0.34	0.46	0.32
WM-HR	0.74	0.61	0.66	0.52	0.50	0.41	0.44	0.38
WoMAP	0.70	0.55	0.82	0.63	0.64	0.52	0.50	0.41

IC: Hard	**PB-Office**		**PB-Kitchen-Easy**		**PB-Kitchen-Medium**		**PB-Kitchen-Hard**	
Planner	Success	Efficiency	Success	Efficiency	Success	Efficiency	Success	Efficiency
VLM	0.08	0.05	0.00	0.00	0.00	0.00	0.00	0.00
DP	0.14	0.10	0.32	0.28	0.20	0.18	0.18	0.16
WM-CEM	0.06	0.04	0.02	0.01	0.02	0.02	0.02	0.02
WM-Grad	0.24	0.11	0.18	0.11	0.16	0.09	0.16	0.09
WM-HR	0.38	0.26	0.24	0.19	0.24	0.19	0.22	0.19
WoMAP	0.58	0.39	0.34	0.25	0.40	0.32	0.34	0.28

IC: Easy	**GS-Kitchen**		**GS-Office**		**GS-Random**	
Planner	Success	Efficiency	Success	Efficiency	Success	Efficiency
DP	0.50	0.47	0.54	0.49	0.28	0.26
VLM	0.12	0.11	0.14	0.13	0.08	0.07
WM-CEM	0.58	0.48	0.40	0.35	0.40	0.30
WM-Grad	0.84	0.67	0.86	0.63	0.76	0.56
WM-HR	0.78	0.66	0.96	0.83	0.58	0.49
WoMAP	0.86	0.74	0.96	0.77	0.72	0.59

IC: Medium	**GS-Kitchen**		**GS-Office**		**GS-Random**	
Planner	Success	Efficiency	Success	Efficiency	Success	Efficiency
VLM	0.06	0.06	0.06	0.05	0.02	0.01
DP	0.42	0.36	0.32	0.29	0.12	0.11
WM-CEM	0.26	0.20	0.22	0.18	0.20	0.13
WM-Grad	0.50	0.32	0.50	0.30	0.36	0.23
WM-HR	0.56	0.45	0.66	0.56	0.30	0.23
WoMAP	0.68	0.55	0.72	0.55	0.46	0.36

IC: Hard	**GS-Kitchen**		**GS-Office**		**GS-Random**	
Planner	Success	Efficiency	Success	Efficiency	Success	Efficiency
VLM	0.00	0.00	0.02	0.02	0.00	0.00
DP	0.18	0.16	0.14	0.11	0.08	0.06
WM-CEM	0.08	0.06	0.08	0.05	0.02	0.01
WM-Grad	0.14	0.09	0.22	0.11	0.00	0.00
WM-HR	0.18	0.15	0.32	0.25	0.08	0.07
WoMAP	0.24	0.18	0.48	0.34	0.12	0.08