Imagine, Verify, Execute: Memory-Guided Agentic Exploration with Vision-Language Models

Seungjae Lee[a*], **Daniel Ekpo**[a*], **Haowen Liu**[a]
Furong Huang[a,b†], **Abhinav Shrivastava**[a†], **Jia-Bin Huang**[a†]
[a] University of Maryland, College Park, [b] Capital One
{sjaelee, daniekpo, hwl, furongh, abhinav, jbhuang}@umd.edu
[*] Equal contribution [†] Equal advising
Project Page: https://ive-robot.github.io/

Abstract: Exploration is essential for general-purpose robotic learning, especially in open-ended environments where dense rewards, explicit goals, or task-specific supervision are scarce. Vision-language models (VLMs), with their semantic reasoning over objects, spatial relations, and potential outcomes, present a compelling foundation for generating high-level exploratory behaviors. However, their outputs are often ungrounded, making it difficult to determine whether imagined transitions are physically feasible or informative. To bridge the gap between imagination and execution, we present IVE (**I**magine, **V**erify, **E**xecute), an agentic exploration framework inspired by human curiosity. Human exploration is often driven by the desire to discover novel scene configurations and to deepen understanding of the environment. Similarly, IVE leverages VLMs to abstract RGB-D observations into semantic scene graphs, imagine novel scenes, predict their physical plausibility, and generate executable skill sequences through action tools. We evaluate IVE in both simulated and real-world tabletop environments. The results show that IVE enables more diverse and meaningful exploration than RL baselines, as evidenced by a 4.1 to 7.8× increase in the entropy of visited states. Moreover, the collected experience supports downstream learning, producing policies that closely match or exceed the performance of those trained on human-collected demonstrations.

Keywords: Exploration, Agentic System, Vision-Language Model

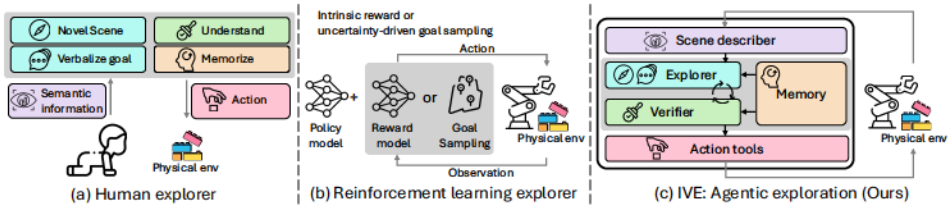

Figure 1: Comparison of human, RL, and IVE exploration strategies. **(a)** Humans explore by seeking novel scene configurations and understanding the environment [1, 2], often enhanced by goal verbalization [3]. **(b)** RL agents explore using a range of techniques, including intrinsic reward or goal sampling, to maximize the coverage of visited states. **(c)** IVE (ours) leverages VLMs to structure exploration via scene description, exploration, verification, memory, and action tools, each aligned with key aspects of human exploration.

1 Introduction

Exploration is a fundamental capability for general-purpose robotic learning, particularly in open-ended environments where dense rewards, explicit goals, or demonstrations are scarce. In such settings, agents must autonomously discover diverse and meaningful interactions to support downstream learning and generalization [4, 5, 6, 7]. Reinforcement learning (RL) has been a dominant paradigm, often using intrinsic rewards to promote novelty [8, 9, 10] or goal sampling to broaden

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

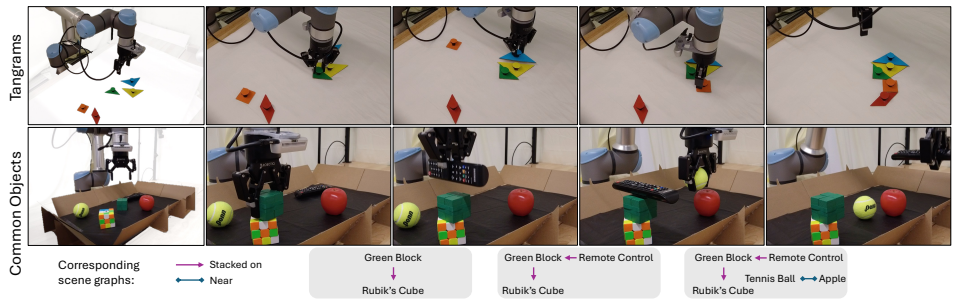

Figure 2: Autonomous scene exploration with IVE. IVE enables autonomous exploration of the scene with diverse objects, e.g., a tangram (*top*) or common objects (*bottom*).

state-space coverage [11, 12]. While effective in simulation or low-dimensional tasks, RL methods often struggle in real-world robotic settings where environments are high-dimensional, semantically rich, and subject to physical constraints and safety risks. In these settings, the undirected or stochastic behaviors often encouraged by RL not only inefficient but also potentially hazardous.

Vision-language models (VLMs) offer a promising alternative. Their broad semantic knowledge and reasoning capabilities have enabled advances in robotic perception, reasoning, and high-level decision making [13, 14, 15, 16]. Building on these strengths, VLMs also offer a promising foundation for guiding exploration by generating hypothetical transitions, or quantifying novelty [17, 18, 19, 20]. However, their imagination often lacks grounding in physical dynamics: transitions may appear semantically plausible yet prove physically infeasible, redundant, or unsafe to execute [21, 22, 23]. Moreover, VLMs operate without structured memory of prior interactions, making it difficult to reason about which states have already been visited or which actions have been attempted. This absence of memory and grounding often leads to redundant, implausible, or low-diversity generations that hinder effective exploration and downstream learning.

To address this challenge, we introduce IVE (**I**magine, **V**erify, **E**xecute), a fully automated, VLM-guided system for agentic exploration, inspired by human exploration—by generating self-directed goals, reacting to new information, and refining their understanding through experience [1, 2, 3, 24]. IVE enables agents to imagine novel future configurations, predict their feasibility based on recent interaction history, and execute selected behaviors via a library of skills. IVE integrates imagination, verification, memory, and action execution into a closed loop, enabling exploration that is both semantically rich and physically grounded. The experience generated by IVE is not only physically grounded and semantically rich, but also directly reusable for downstream policy learning and world model learning. In this work, we make the following key contributions:

- **Curiosity-Driven Exploration via Imagination and Verification.** We introduce IVE, that combines memory-guided imagination with physical plausibility prediction to emulate human-like curiosity in embodied agents, achieving a 4.1 to 7.8× increase in state entropy over RL baselines.
- **Reward-Free Data Collection with VLMs.** We develop a fully automated, vision-language model-guided agentic system for generating semantically meaningful interaction data—without requiring external rewards, demonstrations, or predefined goals, achieving 82% to 122% of the scene diversity exhibited by expert humans.
- **Validation Across Downstream Tasks.** We provide extensive experiments in both simulated and real-world tabletop environments, demonstrating that IVE improves exploration diversity and enables stronger policy learning and world model training compared to RL-based baselines.

2 Related Work

Exploration for Robotic Learning. Exploration is a fundamental challenge in enabling robots, and plays a critical role in building accurate models of environmental dynamics, discovering affordances, and identifying effective strategies for control. To tackle the challenge of exploration, many prior methods have often leveraged RL via intrinsic reward or goal sampling [25]. Intrinsic reward methods

encourage exploration using prediction error [8, 26, 9], entropy [10, 27], or state visitation [28, 29], but often lack semantic understanding about the task, leading the agent to focus on perceptual novelty rather than task-relevant behaviors. In contrast, goal sampling methods guide exploration by sampling goals using temporal distance [30, 31, 32], uncertainty [33, 12], or coverage [11, 34, 35, 36]. While often more directed than intrinsic rewards, these methods still lack mechanisms to identify meaningful goals, and struggle in high-dimensional observation spaces, where learning a reliable latent representation or estimating uncertainty becomes challenging.

Vision-language models for Robotics. VLMs have emerged as powerful tools for bridging visual perception and language-driven understanding. VLMs have demonstrated strong generalization across diverse tasks, enabling applications in robotic task generation [37], autonomous data collection [38], evaluation [39], and serving as high-level planners for low-level action tools [22, 40]. While VLMs offer rich semantic understanding, leveraging them specifically for guiding exploration remains relatively underexplored. Recent efforts include using VLMs to improve goal-conditioned policies [38] or explore by ranking observations based on semantic interestingness [20]. Similarly, we use a VLM in our work to guide exploration by prompting the VLM to imagine and propose physically plausible actions that will lead to novel scene configurations.

Scene graph for observation abstraction. Scene graphs, which encode objects and their relationships as graph structures [41], emerged as a powerful tool for semantic scene understanding in computer vision [42, 43, 44, 45, 46, 47]. Scene graphs have been integrated into robotic pipelines for grounding language instructions into executable actions [48, 49], verifying plan feasibility [50], and supporting open-vocabulary understanding [51]. Scene graph representation also enables hierarchical systems [52, 53, 54], highlighting their role in bridging high-level semantic reasoning with low-level physical execution. Similar to prior methods [50, 48], we use scene graphs as an intermediate representation to the VLM. Beyond this role, we use scene graphs to measure the novelty of new scenes, encouraging the VLM to explore diverse scenes.

3 Method

Overview. We propose a fully automated, VLM-driven exploration system, IVE, built on an agentic architecture composed of three **core** modules: the *Scene Describer*, the *Explorer*, and the *Verifier*. (An overview of the IVE system is shown in Figure 3.) IVE begins by constructing an abstract, semantic representation of the current observation using the *Scene Describer* (Section 3.1). The *Explorer* then proposes candidate future scenes along with skill sequences intended to achieve them (Section 3.2). These candidate plans are evaluated by the *Verifier*, which predicts their physical plausibility and utility before execution (Section 3.3). In addition to these **core** components, two **auxiliary** modules support the system: the *Memory module* retrieves relevant past experiences to inform both the Explorer and Verifier (Section 3.4), and the *Action Tools module* translates the skill sequences into executable robot actions (Section 3.5).

3.1 Scene Describer

The Scene Describer module, powered by a VLM, constitutes the frontmost component of our system, abstracting raw observations into structured scene graphs that capture semantic object relationships. By converting high-dimensional visual inputs into compact, symbolic representations, this abstraction reduces the complexity of reasoning over raw sensory data, enabling more efficient exploration of novel scene configurations.

Given an observation o_t (RGB image at timestep t), the scene describer produces a scene graph $\mathcal{G}_t = (V, E)$, where V denotes the set of objects (nodes) and E encodes directed, typed semantic relations (edges) between pairs of objects. Each object $v \in V$ refers to a unique object name, and edges E include spatial and functional relations such as `Stacked on` and `Near`. The design of the scene graph can be adapted depending on the task, allowing flexibility in the level of abstraction and the types of relations captured.

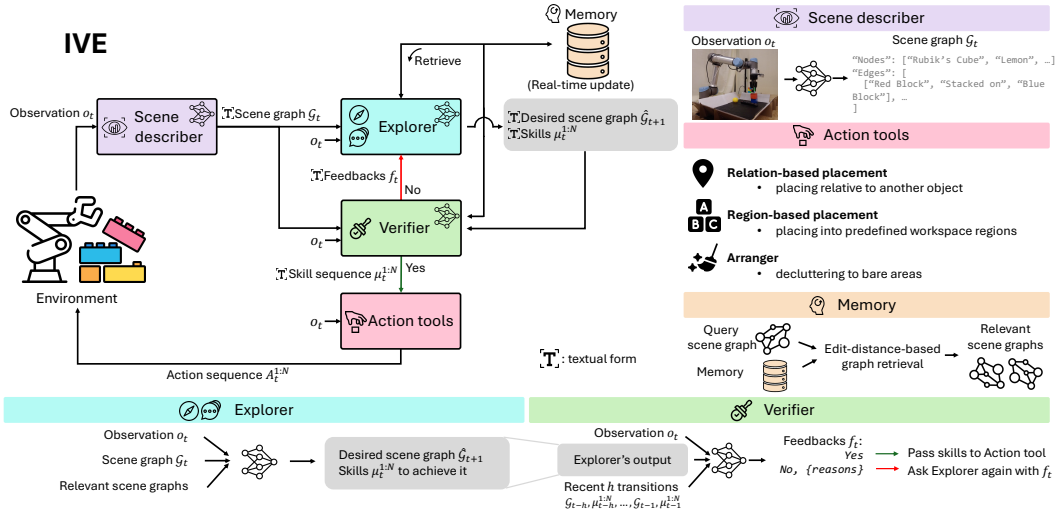

Figure 3: Overview of IVE. Given an observation o_t, the *Scene Describer* constructs a semantic scene graph $\mathcal{G}_t$. The *Explorer* leverages this representation, along with the current observation and retrieved past scene graphs, to generate ("imagine") a candidate future scene graph $\hat{\mathcal{G}}_{t+1}$ and a sequence of skills $\mu_t^{1:N}$. The *Verifier* evaluates the feasibility of these imagined transitions using recent interaction history. If verified, the skills are instantiated into low-level actions via the *Action Tools* and executed by the robot. Otherwise, the Explorer receives feedback and replans. The iterative process enables structured, curiosity-driven exploration grounded in semantic reasoning and informed by physical feasibility.

Illustrative Example. Consider a tabletop scene containing a red cup, a blue block, and a tray. The Scene Describer may produce:

$$V = \{\text{Red cup}, \text{Blue block}, \text{Tray}, \ldots\}$$
$$E = \{(\text{Blue block}, \text{Stacked on}, \text{Tray}), (\text{Red cup}, \text{Near}, \text{Tray})\}$$

This abstract graph captures spatial relations without requiring dense 3D reconstruction. It allows the Explorer module to hypothesize meaningful future configurations (e.g., "move the cup onto the tray") while enabling the Verifier to assess feasibility (e.g., "Is the tray already full?") based on prior experiences.

3.2 Explorer

The Explorer module is a VLM-based component that takes as input the current RGB observation o_t, the corresponding scene graph $\mathcal{G}_t$, and a set of relevant past experiences retrieved from memory (Figure 3). Using this contextual information, the Explorer imagines a future scene graph, $\hat{\mathcal{G}}_{t+1}$, that preserves the object set of $\mathcal{G}_t$ but alters the edge structure—representing new spatial or functional relationships among objects. These imagined transitions enable the agent to reason about potential next states beyond those encountered previously.

To promote novelty and avoid redundancy, the Explorer compares $\hat{\mathcal{G}}_{t+1}$ with retrieved scene graphs from memory, encouraging transitions that are diverse and previously unseen. This memory-aware imagination supports a form of curiosity-driven exploration over structured symbolic representations.

In parallel with the imagined graph, the Explorer generates a sequence of N high-level skills $\mu_t^{1:N}$ intended to transition the agent from $\mathcal{G}_t$ to $\hat{\mathcal{G}}_{t+1}$. Each skill corresponds to a discrete, interpretable action primitive from a predefined skill library. These skill sequences are then passed to the Verifier for physical feasibility assessment *prior* to execution.

Continuing Example. Returning to our earlier example, suppose the current scene graph encodes that the block is `Stacked on` the tray and the cup is `Near` the tray. The Explorer may propose a future graph where the cup is now `Stacked on` the tray and the block is `Near` the tray. It may then generate a skill sequence such as:

$$\mu_t^{1:2} = \{\text{move}(\text{Red cup}, \text{Stacked on}, \text{Tray}), \text{move}(\text{Blue block}, \text{To the left of}, \text{Tray})\}$$

3.3 Verifier

The Verifier module supervises the sequence of skills $\mu_t^{1:N}$ proposed by the Explorer, assessing whether the imagined transition is plausible, physically feasible, and stable. Unlike the Explorer—which operates on current context—the Verifier considers a broader temporal window by accessing recent transitions $\{\mathcal{G}_{t-h}, \mu_{t-h}^{1:N}, \cdots, \mathcal{G}_{t-1}, \mu_{t-1}^{1:N}\}$. This history provides insight into the agent's prior actions and their outcomes, allowing the Verifier to make informed judgments grounded in embodied experience.

Given the current observation and proposed plan, the Verifier predicts the likely outcome and compares it with the Explorer's intended goal graph. Beyond semantic alignment, it also performs stability checks—such as detecting precarious object placements, occlusions, or workspace clutter—that may compromise successful execution. If the proposed plan is unsafe or unlikely to succeed, the Verifier recommends corrective interventions, such as reordering skills, repositioning obstructing objects, or decluttering the workspace.

The Verifier module returns a structured feedback signal f_t composed of:

- a binary decision: "Yes" if the plan is executable or "No" otherwise;
- and, for "No" cases, an explanation detailing rejection reasons (e.g., infeasibility, instability, or deviation from the goal).

Continuing Example. In our tabletop scenario, if the Explorer suggests stacking a cup on a tray that is already full, the Verifier may reject the plan due to instability: It may return:

f_t = No: Cannot place cup on tray — unstable configuration. Suggest removing the Blue block from the Tray first, then placing the Red cup on the tray.

3.4 Memory Retrieval

To support novelty-based exploration and informed decision-making, IVE maintains a dynamic memory module $\mathcal{M}$ that stores previously encountered scene graphs $\mathcal{G}$ derived from past observations. These structured graphs serve as compact, symbolic summaries of prior interactions, enabling the system to reason over what has already been seen and done.

Each $\mathcal{G}$ is instantiated using the NetworkX library, allowing efficient graph storage, manipulation, and querying. At each timestep t, the Explorer queries $\mathcal{M}$ to retrieve a set of past scene graphs that are structurally similar to the current scene graph $\mathcal{G}_t$. Specifically, the retrieved set is defined as

$$\{\mathcal{G}_j \in \mathcal{M} \mid \text{dist}(\mathcal{G}_t, \mathcal{G}_j) < \tau\}, \tag{1}$$

where $\text{dist}(\mathcal{G}_t, \cdot)$ denotes an edit-based graph distance from the current scene $\mathcal{G}_t$, and τ is a predefined similarity threshold. This retrieval process surfaces relevant prior experiences that guide the Explorer in imagining transitions that are both novel (i.e., not previously observed) and physically plausible given historical outcomes.

In parallel, the Verifier leverages the same memory to assess the feasibility of proposed skill sequences, using previously executed transitions as empirical priors for prediction. Thus, memory plays a **dual role**: enabling semantic novelty in the imagination process and providing a grounded context for physical verification.

Continuing Example. Suppose the current scene $\mathcal{G}_t$ encodes a cup next to a tray, with a block positioned above. The memory might retrieve a prior scene where the block was stacked on the cup and the tray was empty, helping the Explorer avoid redundant imagination and providing the Verifier with context on whether that configuration was previously successful or unstable.

3.5 Action Tools

To bridge the high-level skill sequence $\mu_t^{1:N}$ generated by the Explorer with real-world execution, we employ an *Action Tools* module that translates each skill into a corresponding low-level action sequence $A_t^{1:N}$ using RGB-D observations of the current scene. We first instantiate each skill μ_t^i with a predefined set of task-specific primitives that are aligned with the robot's embodiment and operational constraints. After executing a skill, IVE records the resulting state, constructs the updated scene graph $\mathcal{G}_{t+1}$, and stores the transition tuple $(\mathcal{G}_t, \mu_t^{1:N}, \mathcal{G}_{t+1})$ in memory, completing a full imagination-verification-execution cycle. Details of the categories and modularities of the action tools are provided in the appendix D.

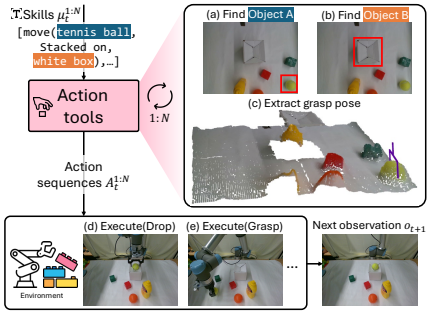

Figure 4: Example of transforming skill to action in a real-world environment.

4 Experiments

Our experiments address three main questions. **Exploration quality**: How does IVE compare to conventional exploration strategies in producing diverse and meaningful interactions (Figure 5)? **Component effectiveness**: How does each module contribute to the overall performance of the system (Figure 6)? **Downstream utility**: How useful is the collected data for downstream tasks (Tables 1, 2)?

4.1 Experimental Setup

Simulation Environment. We use VimaBench [55], a simulated tabletop environment containing multiple rigid objects with varied shapes and colors. The robot is equipped with a suction end-effector and can execute pick-and-place in continuous action space. To efficiently explore this environment, reasoning over spatial configurations while maintaining physical interaction constraints is required.

Real-World Environment. We use a 6-DoF UR5e robot arm with a parallel gripper to interact with tabletop objects. A VLM, GPT-4o [56], observes the current scene and proposes high-level actions, which are then parsed and executed through a low-level controller. We predict the Grasp poses using AnyGrasp [57], with target objects segmented via LangSAM [58], closely following the approaches introduced in [59, 60]. We apply a heuristic preference for top-down grasps for improved reliability. We compute drop positions using depth data and segmentation masks. For the "Stacked on" relationship, we first calculate the midpoint of the target object using the object's segmentation mask, then we calculate the drop height using the depth data for the object region. After each execution, we collect the robot pose, task success status, and RGB data and add them to the experience buffer for downstream tasks.

4.2 IVE Outperforms RL and Human Baselines in Exploration Diversity

We compare IVE against intrinsic motivation RL methods and three human-controlled strategies. For RL baselines, we implement novelty-based intrinsic reward approaches such as *RND* [26] and *RE3* [10]. Human baselines include: *Expert*, given explicit objectives and operating the robot via action tools; *Novice*, interacting freely via action tools, without instructions; and *Moved by hand*, directly manipulates objects without involving the robot arm.

Figure 5 shows that IVE achieves **4.1 to 7.8×** increase in state entropy over RL baselines and **82% to 122%** of the scene diversity exhibited by expert humans. [1] We attribute this to humans gradually forgetting previously encountered scenes, whereas IVE tracks all past configurations via retrieval.

[1] Refer to the Appendix A for details on how entropy and scene counts are computed.

When considering only the first 50 transitions, human performance in simulation closely matches that of IVE. In the real world, IVE slightly underperforms compared to a human expert. See Appendix C for details on the comparative capabilities of different VLMs.

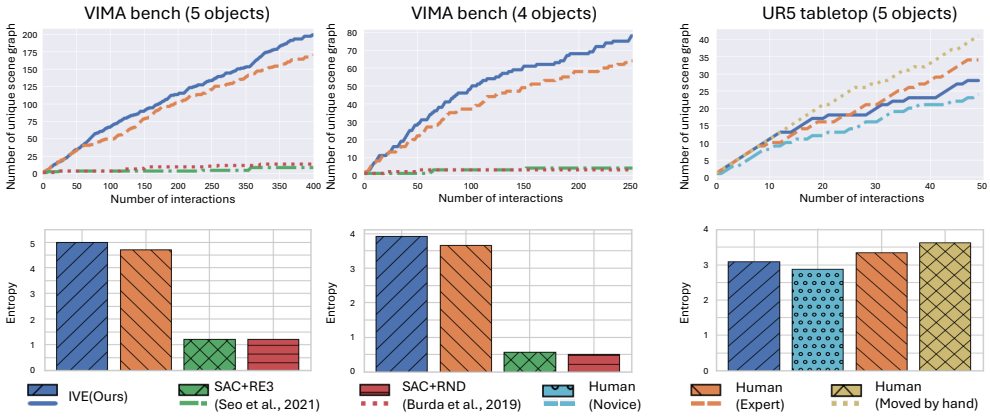

Figure 5: Exploration capability evaluation across simulated and real-world environments. Top: Growth curves of the number of unique scene graphs visited. Bottom: Diversity of visited states, measured by entropy (see Appendix A for the definition and computation details for entropy, and 4.2 for baseline details.

Why RL Falls Short. RL-based baselines perform worse than both IVE and human participants. We believe this reflects a limitation of RL-driven exploration that prioritizes pixel-level novelty over semantically meaningful interactions, especially when they rely solely on intrinsic rewards, making structured and meaningful exploration challenging. In contrast, IVE explores with semantic structure and memory-based recall, enabling higher-level diversity without external rewards.

4.3 Ablations Reveal Memory, Explorer and Verifier Are Critical for Effective Exploration

We ablate components of IVE to quantify their impact on exploration performance (Figure 6). The following variants are tested:

- *Random Tool Selector* uniformly samples action tools with no planning.
- *w/o Explorer (Rule-Based Explorer)* uses simple rules to generate scene graphs; retains VLM for skill generation.
- *w/o Memory* disables retrieval-based grounding.
- *w/o Verifier* skips physical feasibility filtering.

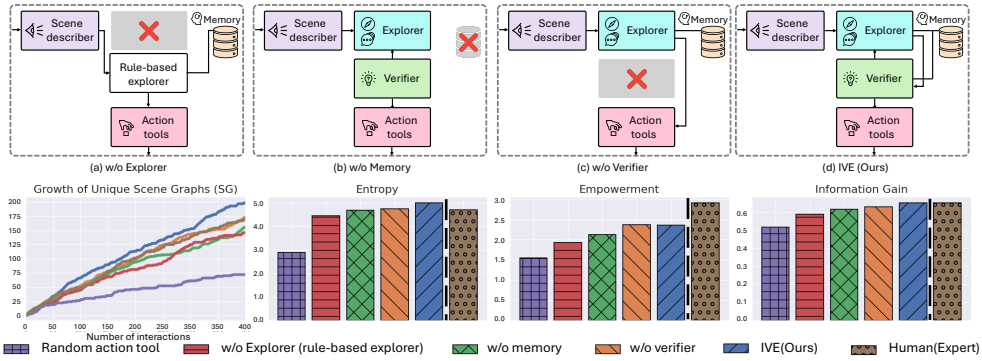

Figure 6: Ablation study of IVE. (Top) Illustration of each variant, highlighting removed modules in gray. (Bottom) Exploration performance is measured by the number of unique scene graphs, entropy, empowerment, and information gain (see Appendix A for metric details).

We evaluate these variants based on the number of unique scene graphs visited, entropy, empowerment, and information gain. Figure 6 shows that removing the memory module and the explorer results in a

22% and 27% drop in unique scenes discovered, respectively, along with a reduction in entropy and information gain—highlighting its importance for novelty-aware planning. Verifier removal also degrades performance, although empowerment remains relatively stable—likely because the verifier proposes fewer decluttering actions, and such actions are inherently more stochastic and less goal-directed. The random tool selector baseline performs the worst, achieving less than half the number of unique graphs discovered by IVE.

Table 1: Performance of goal-conditional and non-conditional behavior cloning across tasks in simulation. Our method achieves human-level performance and significantly outperforms exploration RL baselines (RND [26] and RE3 [10]), demonstrating the effectiveness of our exploration strategy in generating diverse and semantically meaningful data. Example goal images used for the goal-conditioned tests are provided in Appendix E.

		Non-conditional		Goal-conditional
	Exploration Method	# of achieved tasks	Entropy	Success rate
VIMA Bench (5 objects)	SAC [61] + RND [26]	2.0	1.907	8.33%
	SAC [61]+ RE3 [10]	2.1	1.754	0.00%
	IVE (Ours)	**4.1**	**2.283**	**58.33%**
	Human	3.6	2.021	50.00%
VIMA Bench (4 objects)	SAC [61] + RND [26]	2.0	1.907	0.00%
	SAC [61] + RE3 [10]	1.2	1.959	0.00%
	IVE (Ours)	3.1	1.528	**41.67%**
	Human	**4.2**	**1.897**	33.33%

Table 2: Quantitative evaluation of World Model (WM) predictions using datasets collected by different exploration methods, trained with DINO-WM [62].

	Sim Env 1		Sim Env 2		Real World	
Exploration Method	SSIM (↑)	LPIPS (↓)	SSIM (↑)	LPIPS (↓)	SSIM (↑)	LPIPS (↓)
SAC [61] + RND [26]	0.812 ± 0.039	0.198 ± 0.060	0.855 ± 0.036	0.168 ± 0.061	-	-
SAC [61] + RE3 [10]	0.814 ± 0.040	0.199 ± 0.057	0.850 ± 0.034	0.168 ± 0.059	-	-
IVE (Ours)	**0.837 ± 0.032**	0.129 ± 0.044	0.853 ± 0.032	0.160 ± 0.058	0.634 ± 0.075	**0.181 ± 0.056**
Human	0.833 ± 0.032	**0.126 ± 0.042**	**0.862 ± 0.027**	**0.139 ± 0.047**	**0.653 ± 0.072**	0.194 ± 0.056

4.4 IVE Enables Stronger Policy Learning and World Modeling

We evaluate two downstream tasks to validate the usefulness of the data from the exploration. Performance is compared across datasets collected by three strategies: our method, RL-based exploration (SAC [61] with RND [26] and RE3 [10]), and human demonstrations (WM only).

- **World Model Accuracy (WM)**: Measures the prediction accuracy of a learned dynamics model trained on the exploration data using DINO-WM [62].
- **Behavior Cloning (BC)**: Evaluates how well a Diffusion Policy [63], trained on the exploration dataset, can achieve novel goals presented as images.

As shown in Table 1, policies trained on IVE data outperform those trained on RL exploration data by up to **+58% in task success**, and achieve performance **on par with human demonstrations**. Similarly, in world model (WM) prediction tasks, Table 2 shows that IVE achieves performance closest to that of human-collected data. The results on both policy learning and world model prediction highlight the benefits of its structured and diverse exploration strategy for downstream tasks.

5 Conclusion

We introduced IVE, an agentic exploration framework that integrates imagination, verification, and execution to enable efficient exploration in robotic systems. By leveraging the broad knowledge of VLMs, our method enables robots to explore and interact with their environment autonomously. The imagine-verify-execute cycle in IVE promotes high-level semantic diversity during exploration, resulting in rich datasets for learning downstream tasks. In future work, we plan to expand the set of action tools available to the system, enabling more complex interactions and improving the generality of agentic exploration.

Limitations. While our method performs well, it shares common limitations of real-world robotic systems. VLM-based reasoning introduces some latency, which could be reduced with lighter or distilled models. Our action tools are manually defined, which limits scalability for complex tasks; integrating learned policies as tools could address this limitation. Finally, our reliance on open-vocabulary object detection can lead to failures with occluded or novel objects—future work could incorporate multi-view perception or interactive discovery to improve robustness.

Acknowledgments

This work was partially supported by (a) NSF CAREER Award (#2238769) to AS, and (b) DARPA TIAMAT (#80321), NSF Award (#2147276 FAI), DOD AFOSR Award (#FA9550-23-1-0048), and Adobe, Capital One and JP Morgan faculty fellowships to FH. The authors acknowledge UMD's supercomputing resources made available for conducting this research. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF, Adobe, Capital One, JP Morgan, or the U.S. Government. We thank Amir-Hossein Shahidzadeh, Eric Zhu, and Mara Levy for their help and advice.

References

[1] A. Ten, P. Kaushik, P.-Y. Oudeyer, and J. Gottlieb. Humans monitor learning progress in curiosity-driven exploration. *Nature communications*, 2021.

[2] A. Modirshanechi, K. Kondrakiewicz, W. Gerstner, and S. Haesler. Curiosity-driven exploration: foundations in neuroscience and computational modeling. *Trends in Neurosciences*, 2023.

[3] A. Lidayan, Y. Du, E. Kosoy, M. Rufova, P. Abbeel, and A. Gopnik. Intrinsically-motivated humans and agents in open-world exploration. *arXiv preprint arXiv:2503.23631*, 2025.

[4] A. Aubret, L. Matignon, and S. Hassas. An information-theoretic perspective on intrinsic motivation in reinforcement learning: A survey. *Entropy*, 25(2), 2023.

[5] S. Lee, D. Cho, J. Park, and H. J. Kim. Cqm: Curriculum reinforcement learning with a quantized world model. *Advances in Neural Information Processing Systems*, 2023.

[6] B. Sukhija, S. Coros, A. Krause, P. Abbeel, and C. Sferrazza. MaxinfoRL: Boosting exploration in reinforcement learning through information gain maximization. In *International Conference on Learning Representations*, 2025.

[7] G.-C. Anthony, K. Marino, and R. Fergus. Efficient exploration and discriminative world model learning with an object-centric abstraction. In *International Conference on Learning Representations*, 2024.

[8] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, 2017.

[9] R. Mendonca, O. Rybkin, K. Daniilidis, D. Hafner, and D. Pathak. Discovering and achieving goals via world models. *Advances in Neural Information Processing Systems*, 2021.

[10] Y. Seo, L. Chen, J. Shin, H. Lee, P. Abbeel, and K. Lee. State entropy maximization with random encoders for efficient exploration. In *International Conference on Machine Learning*, volume 139, 2021.

[11] V. H. Pong, M. Dalal, S. Lin, A. Nair, S. Bahl, and S. Levine. Skew-fit: state-covering self-supervised reinforcement learning. In *International Conference on Machine Learning*, 2020.

[12] E. S. Hu, R. Chang, O. Rybkin, and D. Jayaraman. Planning goals for exploration. In *International Conference on Learning Representations*, 2023.

[13] 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. In *Conference on Robot Learning*, 2023.

[14] D. Shah, B. Osiński, S. Levine, et al. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action. In *Conference on Robot Learning*, 2023.

[15] H. Etukuru, N. Naka, Z. Hu, S. Lee, C. Paxton, S. Chintala, L. Pinto, and N. M. M. Shafiullah. Robot utility models: General policies for zero-shot deployment in new environments. In *CORL Workshop*, 2024.

[16] A. H. Tan, A. Fung, H. Wang, and G. Nejat. Mobile robot navigation using hand-drawn maps: A vision language model approach. *arXiv preprint arXiv:2502.00114*, 2025.

[17] Y. Kuang, H. Lin, and M. Jiang. Openfmnav: Towards open-set zero-shot object navigation via vision-language foundation models. In *ICLR Workshop*, 2024.

[18] W. Jiang, B. Lei, K. Ashton, and K. Daniilidis. Multimodal llm guided exploration and active mapping using fisher information. *arXiv preprint arXiv:2410.17422*, 2024.

[19] H. Jiang, B. Huang, R. Wu, Z. Li, S. Garg, H. Nayyeri, S. Wang, and Y. Li. RoboEXP: Action-conditioned scene graph via interactive exploration for robotic manipulation. In *Conference on Robot Learning*, 2024.

[20] C. Sancaktar, C. Gumbsch, A. Zadaianchuk, P. Kolev, and G. Martius. SENSEI: Semantic exploration guided by foundation models to learn versatile world models. In *Workshop on Training Agents with Foundation Models at RLC*, 2024.

[21] L. Li, J. Xu, Q. Dong, C. Zheng, X. Sun, L. Kong, and Q. Liu. Can language models understand physical concepts? In *Conference on Empirical Methods in Natural Language Processing*, 2023.

[22] Y. Hu, F. Lin, T. Zhang, L. Yi, and Y. Gao. Look before you leap: Unveiling the power of gpt-4v in robotic vision-language planning. In *ICRA Workshop*, 2024.

[23] M. Elnoor, K. Weerakoon, G. Seneviratne, R. Xian, T. Guan, M. K. M. Jaffar, V. Rajagopal, and D. Manocha. Robot navigation using physically grounded vision-language models in outdoor environments. *CoRR*, 2024.

[24] L. Gaven, T. Carta, C. Romac, C. Colas, S. Lamprier, O. Sigaud, and P.-Y. Oudeyer. Magellan: Metacognitive predictions of learning progress guide autotelic llm agents in large goal spaces. In *International Conference on Machine Learning*. Proceedings of Machine Learning Research, 2025.

[25] C. Colas, T. Karch, O. Sigaud, and P.-Y. Oudeyer. Autotelic agents with intrinsically motivated goal-conditioned reinforcement learning: a short survey. *Journal of Artificial Intelligence Research*, 74, 2022.

[26] Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019.

[27] D. Kim, J. Shin, P. Abbeel, and Y. Seo. Accelerating reinforcement learning with value-conditional state entropy exploration. *Advances in Neural Information Processing Systems*, 2023.

[28] J. Martin, S. S. Narayanan, T. Everitt, and M. Hutter. Count-based exploration in feature space for reinforcement learning. In *International Joint Conference on Artificial Intelligence*, 2017.

[29] A.-H. Shahidzadeh, S. J. Yoo, P. Mantripragada, C. D. Singh, C. Fermüller, and Y. Aloimonos. Actexplore: Active tactile exploration on unknown objects. In *International Conference on Robotics and Automation*, 2024.

[30] I. Durugkar, S. S. Hansen, S. Spencer, and V. Mnih. Wasserstein distance maximizing intrinsic control. In *NeurIPS Workshop*, 2021.

[31] M. Klissarov and M. C. Machado. Deep laplacian-based options for temporally-extended exploration. In *International Conference on Machine Learning*, 2023.

[32] J. Bae, K. Park, and Y. Lee. Tldr: Unsupervised goal-conditioned rl via temporal distance-aware representations. In *Conference on Robot Learning*, 2024.

[33] D. Cho, S. Lee, and H. J. Kim. Outcome-directed reinforcement learning by uncertainty\& temporal distance-aware curriculum goal generation. In *International Conference on Learning Representations*, 2023.

[34] S. Pitis, H. Chan, S. Zhao, B. Stadie, and J. Ba. Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. In *International Conference on Machine Learning*, 2020.

[35] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Reinforcement learning with prototypical representations. In *International Conference on Machine Learning*. Proceedings of Machine Learning Research, 2021.

[36] S. V. Mahankali, Z.-W. Hong, A. Sekhari, A. Rakhlin, and P. Agrawal. Random latent exploration for deep reinforcement learning. In *International Conference on Machine Learning*, 2024.

[37] 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.

[38] Z. Zhou, P. Atreya, A. Lee, H. R. Walke, O. Mees, and S. Levine. Autonomous improvement of instruction following skills via foundation models. In *Conference on Robot Learning*, 2024.

[39] Z. Zhou, P. Atreya, Y. L. Tan, K. Pertsch, and S. Levine. Autoeval: Autonomous evaluation of generalist robot manipulation policies in the real world. In *CORL Workshop*, 2025.

[40] R. Shah, A. Yu, Y. Zhu, Y. Zhu, and R. Martín-Martín. Bumble: Unifying reasoning and acting with vision-language models for building-wide mobile manipulation. *arXiv preprint arXiv:2410.06237*, 2024.

[41] J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. Shamma, M. Bernstein, and L. Fei-Fei. Image retrieval using scene graphs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[42] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal on Computer Vision*, 123, 2017.

[43] J. Ji, R. Krishna, L. Fei-Fei, and J. C. Niebles. Action genome: Actions as compositions of spatio-temporal scene graphs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[44] J. Johnson, A. Gupta, and L. Fei-Fei. Image generation from scene graphs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[45] O. Ashual and L. Wolf. Specifying object attributes and relations in interactive scene generation. *International Conference Computer Vision*, 2019.

[46] H. Dhamo, A. Farshad, I. Laina, N. Navab, G. Hager, F. Tombari, and C. Rupprecht. Semantic image manipulation using scene graphs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[47] R. Herzig, A. Bar, H. Xu, G. Chechik, T. Darrell, and A. Globerson. Learning canonical representations for scene graph to image generation. In *European Conference on Computer Vision*, 2020.

[48] 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 *Conference on Robot Learning*, 2023.

[49] Z. Ni, X. Deng, C. Tai, X. Zhu, Q. Xie, W. Huang, X. Wu, and L. Zeng. Grid: Scene-graph-based instruction-driven robotic task planning. In *International Conference on Intelligent Robots and Systems*, 2024.

[50] D. Ekpo, M. Levy, S. Suri, C. Huynh, and A. Shrivastava. Verigraph: Scene graphs for execution verifiable robot planning. *arXiv preprint arXiv:2411.10446*, 2024.

[51] Q. Gu, A. Kuwajerwala, S. Morin, K. M. Jatavallabhula, B. Sen, A. Agarwal, C. Rivera, W. Paul, K. Ellis, R. Chellappa, et al. Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning. In *International Conference on Robotics and Automation*, 2024.

[52] Z. Ravichandran, L. Peng, N. Hughes, J. D. Griffith, and L. Carlone. Hierarchical representations and explicit memory: Learning effective navigation policies on 3d scene graphs using graph neural networks. In *International Conference on Robotics and Automation*, 2022.

[53] G. Zhai, X. Cai, D. Huang, Y. Di, F. Manhardt, F. Tombari, N. Navab, and B. Busam. Sg-bot: Object rearrangement via coarse-to-fine robotic imagination on scene graphs. In *International Conference on Robotics and Automation*, 2024.

[54] S. Amiri, K. Chandan, and S. Zhang. Reasoning with scene graphs for robot planning under partial observability. *IEEE Robotics and Automation Letters*, 2022.

[55] Y. Jiang, A. Gupta, Z. Zhang, G. Wang, Y. Dou, Y. Chen, L. Fei-Fei, A. Anandkumar, Y. Zhu, and L. Fan. Vima: General robot manipulation with multimodal prompts. In *International Conference on Machine Learning*, 2023.

[56] OpenAI. Gpt-4o: Openai's new multimodal model. https://openai.com/index/gpt-4o, 2024. Accessed: 2024-04-30.

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

[58] L. Medeiros. Lang segment anything. https://github.com/luca-medeiros/lang-segment-anything, 2023.

[59] P. Liu, Z. Guo, M. Warke, S. Chintala, C. Paxton, N. M. M. Shafiullah, and L. Pinto. Dynamem: Online dynamic spatio-semantic memory for open world mobile manipulation. In *CORL Workshop*, 2024.

[60] P. Liu, Y. Orru, J. Vakil, C. Paxton, N. M. M. Shafiullah, and L. Pinto. Ok-robot: What really matters in integrating open-knowledge models for robotics. In *ICRA Workshop*, 2024.

[61] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 2018.

[62] G. Zhou, H. Pan, Y. LeCun, and L. Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

[63] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2023.

A Evaluation Metrics for Exploration Capability

To quantitatively evaluate the agent's ability to explore its environment, we use the following metrics that capture diversity, informativeness, and control capacity over future states, as described in [3]. We treat the state space $\mathcal{S}$ ($s \in \mathcal{S}$) as a discrete set of scene graphs to enable interpretable analysis.

- **Unique Scene Graphs**: The number of distinct scene graphs encountered during exploration. A higher count reflects greater semantic diversity.
- **State Entropy**: Measures the entropy of the agent's state visitation distribution. Let N_s denote the number of visits to state s. Then $p(s) = N_s / \sum_{s'} N_{s'}$, and entropy is given by:

$$H(S) := -\sum_{s} p(s) \log p(s)$$

- **Information Gain (IG)**: Quantifies how much new information is acquired in each episode. Define $N^e_{s,a}$ as the count of action a taken in state s up to episode e. The information gain for episode e is:

$$IG^e := \frac{\sum_{(s,a)} IG^e_0(s,a) - IG^{e-1}_0(s,a)}{\sum_{(s,a)} N^e_{s,a} - N^{e-1}_{s,a}},$$

  where $IG^e_0(s,a) := \log(1 + N^e_{s,a})$.
- **Empowerment**: Captures the agent's control over future outcomes. Defined as the mutual information between actions and resulting states:

$$E := \max_{p(a)} I(s'; a \mid s) = \max_{p(a)} \sum_{s',a} p(s'|a)p(a) \log \frac{p(s'|a)}{\sum_{a'} p(s'|a')p(a')}$$

  Since exact computation is intractable, we approximate $p(s'|a)$ via sampling and scene graph transition statistics.

For fair comparison, we quantize observations into scene graphs using methods that differ from those employed in IVE. In simulation, we construct scene graphs using a heuristic based on ground-truth object positions, encoding relative distances and spatial relationships. In the real world, where ground-truth positions are unavailable, we generate scene graphs using a separate perception pipeline. Importantly, both the prompt design and scene graph structure used for evaluation are distinct from those used in IVE. Please note that in both settings, none of the agents—including IVE and all baselines—have access to the ground-truth object positions or the internal graphs used for evaluation.

B Real world robot setup

The system is implemented on a Universal Robot UR5e robot arm with a Robotiq 2F-85 gripper. An Intel Realsense D435i depth camera is mounted on the robot end-effector. The workspace is a tabletop workspace with predefined boundaries. All robot poses are clipped to the workspace boundaries for safety.

B.1 Grasp Planning and Execution

The system implements two grasp strategies. The primary strategy uses Anygrasp [57] for grasp pose detection, which provides the grasp pose in the camera frame. We transform this to the robot base frame using the camera calibration and convert the 4×4 matrix to an axis-angle rotation vector using the Rodrigues method. The execution sequence moves to a clearance height 0.1m above the grasp pose, aligns rotation, descends to the grasp pose with a z-offset of -0.048m. The object is lifted to a clearance height to avoid collision and then moved to the destination pose.

The tangram grasp strategy uses a centroid-based approach. We compute the object centroid using image moments, project it to 3D using the pinhole camera model, and transform to the robot base frame. The grasp pose is set to the centroid position with a z-offset of -0.01m and a fixed rotation of $[0, -\pi, 0]$.

B.2 Relation-based Placement and Tangram Manipulation

The system implements six spatial relationships: STACKED ON, IN_FRONT_OF, BEHIND, TO_LEFT_OF, TO_RIGHT_OF, and ARRANGE. For each relationship, we compute the drop point using a relationship-specific algorithm.

- STACKED ON relationship calculates the drop height by finding the maximum z-coordinate of the target object in the depth image and adding a drop height offset of 0.01m.
- IN_FRONT_OF and BEHIND relationships compute a y-offset of ± 0.08m from the target object's bounding box.
- TO_LEFT_OF and TO_RIGHT_OF relationships use an x-offset of ± 0.08m.
- ARRANGE relationship employs a depth-based ground plane detection algorithm. We create a ground mask by thresholding the depth image at the table height, erode it using a kernel size proportional to the manipulated object's dimensions, and apply boundary constraints excluding regions very close to the workspace edges. From the valid placement regions, we randomly sample a placement point to introduce diversity while maintaining safety constraints.

For tangram manipulation, we implement a specialized edge alignment system that first detects polygon edges using contour detection with an epsilon ratio of 0.02. The system then compares the source and destination masks to find the optimal edge alignment. For each pair of edges, we compute the alignment angle by finding the angle between the edge vectors, considering both parallel and anti-parallel alignments. Since tangram pieces are constrained to rotate only in the z-axis, we compute the rotation matrix around the z-axis using the alignment angle. The translation is determined by computing the vector between the midpoints of the aligned edges, with additional jitter sampling to account for small variations in placement. We evaluate each potential alignment by computing the contact length between the edges and applying an occlusion penalty based on the overlap between the source and destination masks. The system selects the alignment with the maximum contact length while minimizing occlusion. All coordinate transformations between camera and robot frames are handled using standard eye-in-hand calibration and the pinhole camera model.

B.3 Region-based Placement

To enable the Vision-Language Model (VLM) to refer to specific spatial locations in the workspace, we introduce a Region-Based Placement Tool that discretizes the environment into a labeled grid map (Figure 7). The workspace is overlaid with a checkerboard-style grid, where each cell is uniquely indexed (e.g., A1 to E10). This grid is rendered as an image and passed to the VLM.

Given this structured input, the VLM can issue explicit placement instructions using symbolic coordinates: `move(object_name, target_grid)`. For instance, the command `move(red_cross, B8)` indicates that the object referred to as `red_cross` should be placed in cell `B8`. This discrete representation allows the model to generate unambiguous spatial commands and simplifies the mapping from language to robot actions.

Figure 7: The Region-Based Placement Tool overlays the workspace with a labeled grid, allowing the VLM to reference specific spatial locations when issuing placement commands.

C Comparative Exploration Capabilities Across VLMs

In this section, we present the exploration performance of IVE when paired with different types of Vision-Language Models (VLMs).

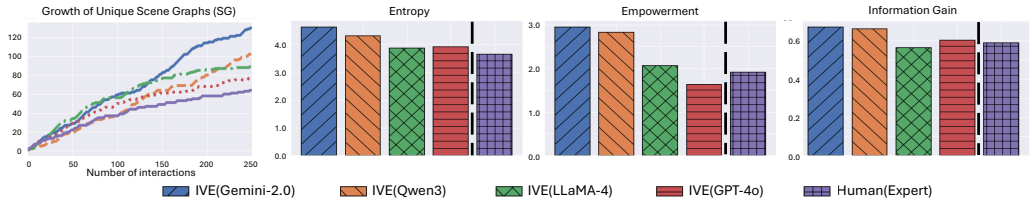

Figure 8: Exploring with Embodied Agents: This figure compares the exploration capabilities of our method, IVE, powered by different Vision-Language Models (VLMs) rather than GPT-4o. Notably, IVE, regardless of the VLM used, matches or surpasses the human expert in generating unique scene graphs, achieving higher state diversity, and gaining more information.

D Action Tool Details

Tool Categories. Our action toolset includes three discrete types of manipulators:

1. **Relation-Based Placement Tools:** Execute relational actions that position objects with respect to others (e.g., "To the left of," "Stacked on") as shown in Figure 4.
2. **Region-Based Placement Tools:** Place objects at specific locations on a predefined 2D layout (e.g., grid cells).
3. **Arranger Tool:** Manages workspace cleanliness by moving unreferenced or obstructive objects to free, uncovered regions, enabling subsequent actions.

Modularity. The design of the Action Tools module is deliberately modular and extensible—new primitives or skills can be incorporated seamlessly without requiring architectural changes to other components. This modularity ensures that IVE can adapt to diverse task domains and hardware platforms by *swapping* or *extending* action capabilities.

E Evaluations on Downstream Tasks

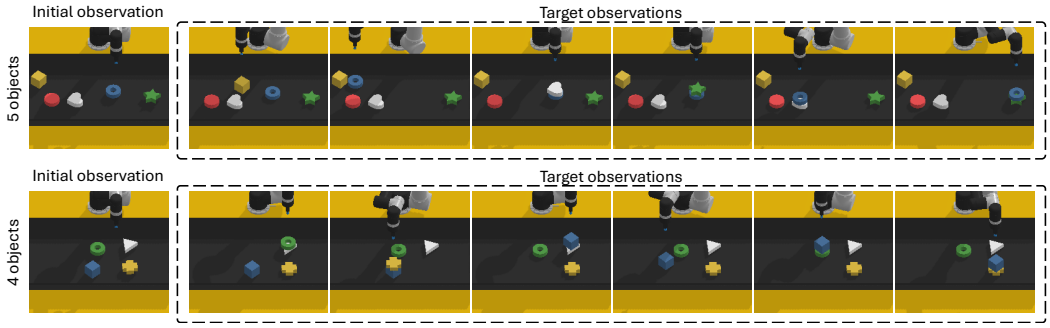

Figure 9: To evaluate the performance of the behavior cloning policy, we train Diffusion Policy [63] on each dataset and evaluate it on goal-conditioned tasks, where the initial observation is fixed and the agent is tested with six different goals.

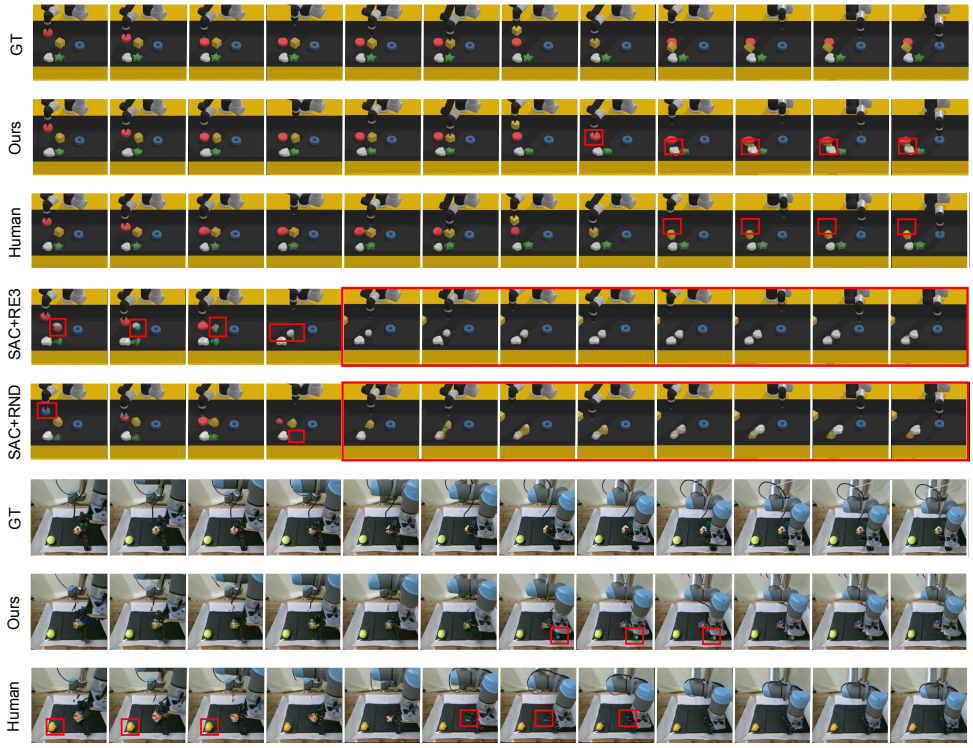

Figure 10: Qualitative examples of World Model (WM) predictions using datasets collected by different exploration methods. Red rectangles highlight regions with notable prediction errors.

F Prompts for IVE

F.1 Scene Describer

The Scene Describer takes an RGB image and produces a structured scene graph that captures object identities and spatial relations. This module enables symbolic reasoning by abstracting the raw observation into a graph-structured representation. The prompt below guides a vision-language model to construct this graph iteratively.

The process consists of three main stages:

- Step 1 - QnA Section: For each object, the model predicts its closest objects and describes their spatial relationship from that object's perspective.
- Step 2 - Iterative Scene Graph Construction: The scene graph is built incrementally by adding one object at a time and determining its relation to previously introduced objects.
- Step 3 - Final Scene Graph Output: The final graph is compiled from all previously gathered information, listing nodes (objects) and edges (relations) using only allowed relation types.

Below is the exact prompt used to guide the model:

Prompt.

```
## Your Task
You are an expert image analyzer tasked with identifying the **exact**
     placement and spatial relationships of specific objects. Your job
     is to generate a scene graph describing these spatial relations
    **solely** based on the objects' visible positions in the image.

As an image analyzer, Follow Step 1~3 below.
```

```
---

## Step 1: Fill the Answer in QnA Section

---

## Step 2: Iterative Scene Graph Construction
1. Begin with one object.
2. Add one new object at a time, to your partial scene graph.
3. For each newly added object:
   - Determine its spatial relation(s) to the objects already in the
     scene graph.
   - **Use only** the Allowed Relations in the scene graph.
   - Do not assign more than one relation for the same object pair `(
     new_object, existing_object)` == `(existing_object, new_object)
     `
   - You may introduce multiple relations at once if the new object
     relates to multiple existing objects.

---

## Step 3: Final Scene Graph Output
1. **Once all objects** have been introduced and verified, compile a
   **complete scene graph**:
   - **List all nodes** (the objects in the final scene).
   - **List all verified relations** between pairs of objects, using
     the Allowed Relations in the scene graph.
2. **Use only** objects from the "Global Object Names."
3. Even if there's missing nodes or edges in a final scene graph (
   because at least one object is missing), you must still provide a
   complete **scratch pad** and **scene graph** with existing
   relations.

---

## Scene Graph Representation

- Nodes: Objects present in the scene.
- Relations: Spatial relationships between object pairs.
- Allowed Relations in the scene graph:
    - **Stacked On**: Object A is physically resting on Object B. This
      requires clear direct contact-Object A is visibly supported
      by Object B from below.
    - **Near**: Object A is positioned close to Object B without being
      stacked. Use this only when the objects are almost touching.

---
## Global Object Names
`<GLOBAL_OBJECTS_HERE>`
---
## Output Format
Please structure your final output exactly as shown below (without the
    lines). **Use the precise section titles**:
-------------
[Step 1: Fill the Answer in QnA Section]
<QNA_FOR_OBJECT_RELATION>
[Step 2: Iterative Scene Graph Construction]
Iteration 1:
- Added obj_a.
- Explanation of how you confirmed its presence in the image.
Iteration 2:
- Added obj_b.
- <obj_b, relation_type, obj_a> or <obj_a, relation_type, obj_b> (
   include any additional relations or notes)
```

```
- Explanation of how you verified this relation.
... (continue until all objects are added and checked)
[Step 3: Final Scene Graph Output]
<start_graph>
Nodes: obj_a, obj_b, ...
Relations: <obj_a, Near, obj_b>, <obj_b, Near obj_c>, <obj_d, Stacked
    On, obj_c>, ...
<end_graph>
-------------
```

QnA section. Below is an example of a generated QnA section, filled out based on a set of sample object names:

```
[Step 1: Fill the Answer in QnA Section]
Object 1: red cube
--------------------------------
What are the closest 0~3 objects from red cube? What are their
    relations from red cube's perspective?
Answer: The red cube is near the blue cylinder and stacked on the
    green base.
--------------------------------
Object 2: blue cylinder
--------------------------------
What are the closest 0~3 objects from blue cylinder? What are their
    relations from blue cylinder's perspective?
Answer: The blue cylinder is near the red cube.
--------------------------------
Object 3: green base
--------------------------------
What are the closest 0~3 objects from green base? What are their
    relations from green base's perspective?
Answer: The green base has the red cube stacked on top.
--------------------------------
```

F.2 Explorer

The Scene Explorer module performs planning over an environment with objects. It receives a current scene graph and predict a valid action sequence that results in a novel configuration. This task challenges the model to reason about physics, constraints, and symbolic novelty.

The prompt includes:

- Action history: Provides the model with previously executed action sequence.
- Scene graph history: Supplies the model with previously visited scene graphs (which is retrived from memory), encouraging novelty.

```
## Your Task
You are an expert spatial planner. Given the Current Image, your job
    is to generate a sequence of actions that discover a new scene
    configuration-one that has not been seen before.
- In addition to the action sequence, you must provide the predicted
    future scene graph (desired scene graph) that results from these
    actions.
- You have two images taken from different camera viewpoints.
- You should provide at most '<NUM_STEPS_HERE>' actions.
---
## Scene Graph Representation
- Nodes: Objects present in the scene.
- Relations: Spatial relationships between object pairs.
- Allowed Relations in the scene graph:
    - **Stacked On**: Object A is physically resting on Object B. This
        requires clear direct contact-Object A is visibly supported
       by Object B from below.
```

```
    - **Near**: Object A is positioned close to Object B without being
        stacked. Use this only when the objects are almost touching.
---
## Global Object Names
‘<GLOBAL_OBJECTS_HERE>’
---
<ACTION_TYPES>
---
## Current Scene Graph
‘<CURRENT_SCENE_GRAPH>’
---
## Scene Graph History
Shows previously visited scene graphs most similar to your current
    scene.
<SCENEGRAPH_HISTORY>
---
## Action History
‘<ACTION_HISTORY>’
---
## Output Format
Your output format should look exactly like the content between the
    ‘-----’. **Do not** number the actions. It’s important to wrap the
     action sequence between ‘<start_action_sequence>’ and ‘<
    end_action_sequence>’. Also, write down the predicted future scene
     graph (desired scene graph - the final arrangement after all
    actions) between ‘<start_graph>’ and ‘<end_graph>’.
-----
<start_scratch_pad>
Explain your reasoning:
- Why this is a novel scene
- Why the action sequence makes sense
- If there were oddities or contradictions in the histories, how did
    you account for possible collisions, suction errors, or clutter?
<end_scratch_pad>
Predict (Desired) Future Scene Graph:
<start_desired_scene_graph>
Nodes: obj_a, obj_b, ...
Relations: <obj_a, Near, obj_b>, <obj_b, Near obj_c>, <obj_d, Stacked
    On, obj_c>, ...
<end_desired_scene_graph>
Next Action Sequence:
<start_action_sequence>
<ACTION_SEQUENCE_EXAMPLE>
<end_action_sequence>
-----
### Important Considerations
1. Order Matters: Plan your actions so that preconditions are
    satisfied before you move an object.
2. Scene Boundaries: If an object is near the scene boundary, avoid
    pushing it further toward the edge or placing new objects in a
    risky position.
3. Manipulation (Suction) Constraints:
   - The suction can only reliably pick the topmost exposed surface.
   - In cluttered areas, an attempt to move one object may cause
       unintended collisions or shifts in neighboring objects.
   - Stacking another object on top of an unstable object can lead to
       the object toppling over.
4. Note: The list of allowed relations in Action Types and the
    relations used in Scene Graph Representation ([Stacked On, Near])
    may differ. Desired Scene Graph should use relations among <
    SCENEGRAPH_RELATIONS> only, same as other Scene Graphs. Please
    keep this in mind when planning your actions.
```

Action types.

Actions available to the scene explorer fall into the following categories. These are symbolic commands grounded in real-world physical execution, and the model may extend this vocabulary when necessary.

```
### Action Types

Actions are formatted in two ways:

1. 'move(obj_a, RELATION, obj_b)'
   e.g., 'move(white cup, Stacked On, red plate)'
   - Moves one object to a position relative to another.
   - Allowed RELATION list: '[In Front Of, Behind, To The Left Of, To
       The Right Of, Stacked On]'


2. 'move(obj_a, GRID_ID)'
   e.g., 'move(blue ball, B3)'
   - Moves an object to a grid location on the image. ('["A1", "B3",
       ..., "E10"]')

3, 'arrange(obj_a)'
   e.g., 'arrange(red block)'
   - Pick up the objects and organize them in a clear area on the
       Workspace.
```

F.3 Verifier

The Scene Verifier is responsible for checking the validity and physical feasibility of a proposed action sequence in dynamic environments. It assesses whether the actions, when executed from the current scene, would produce the desired result without causing instability or unintended configurations.

One core component of this process is the transition history, a temporally ordered trace of the environment, alternating between scene graphs and actions:

$$\text{Scene Graph}_1 \rightarrow \text{Action}_1 \rightarrow \text{Scene Graph}_2 \rightarrow \text{Action}_2 \rightarrow ... \rightarrow \text{Scene Graph}_n$$

This history provides concrete grounding to reason about object configurations and action effects, enabling the verifier to anticipate unintended side-effects like toppling, occlusions, or manipulation errors.

The verifier simulates outcomes, checks for physical plausibility, and may provide targeted suggestions or recommend a decluttering strategy in edge cases.

```
## Your Task
You are a spatial reasoning expert responsible for **verifying action
    plans** in physically dynamic environments.
You ensure that a proposed sequence of actions logically leads from
    the current state to the desired scene graph, without triggering
    unintended outcomes.
You may also provide **targeted suggestions** or, in rare but
    necessary cases, recommend a **temporary shift to a decluttering
    strategy**.
---
## Goals
Given the current image (from two camera views), transition history,
    desired scene graph, and a proposed action sequence:
1. **Simulate** the effect of the action sequence from the current
    scene
2. **Predict** the resulting scene graph
3. **Compare** the predicted graph with the desired one
4. **Evaluate physical feasibility and execution stability**
5. **Provide a judgment**:
   - Valid and feasible
```

```
   - Invalid (with reason)
   - Valid but risky (suggest a targeted fix)
   - Too unstable to proceed (recommend declutter mode)
---
<ACTION_TYPES>
---
## Transition History
A sequence of alternating scene graphs and actions showing the
    environment's evolution.
'<TRANSITION_HISTORY>'
---
## Output Format
-----
<start_scratch_pad>
Step-by-step analysis:
- Simulate and predict the resulting scene graph.
Scene Stability Check:
- Are any objects in clearly unstable or unreachable positions?
- Do previous transitions indicate failures or ambiguous changes?
- Are cluttered zones, deep stacks, or occlusions affecting safety or
    reliability?
Decision:
- Is the action sequence logically valid and does it produce the
    desired scene graph?
  -> YES or NO
If NO:
- Explain which actions fail and why.
- Point out mismatches or invalid transitions.
If issues are detected:
- Identify objects or areas causing risk (e.g., unstable stacks,
    blocked objects).
- Suggest fine-grained intervention (e.g., "move obj_A before
    continuing").
If the environment is severely cluttered and unsafe:
- Recommend a temporary shift to a decluttering mode
<end_scratch_pad>
<start_decision>
YES or NO
<end_decision>
<start_reason>
[If NO: Brief but clear explanation of what failed or was mismatched]
- risky: Warning message with suggestion, e.g., "Unstable stack: move
    obj_b before continuing"
- Too unstable: "Scene too cluttered. Recommend temporary declutter
    mode."
[If YES and no issues: Leave this part empty]
<end_reason>
-----
---
## Scene Stability Considerations
Clutter or instability **does not always require full decluttering**.
    Consider recommending targeted fixes first.
#### Examples of Minor Intervention:
- '"obj_b is stacked on obj_a, which is already supporting obj_c.
    Recommend moving obj_b first to prevent instability."'
- '"obj_d is partially occluded and may be hard to suction. Recommend
    shifting nearby obj_e first."'
#### Examples of Decluttering (rare):
- '"Multiple overlapping clusters and deep stacks suggest high
    instability. Recommend decluttering of current layout before
    further scene exploration."'
```