SCOUT: Planning under Occlusion via Object-Centric World Model Rollouts

Amir Hossain Raj **Xuesu Xiao**
George Mason University
{araj20, xiao}@gmu.edu

Abstract

Robots manipulating in cluttered spaces must often plan from partial views where foreground objects occlude the target, surrounding geometry, and usable free space. Reactive policies often miss these hidden constraints, while exhaustive search becomes computationally prohibitive in dense clutter. We present SCOUT, an occlusion-aware planning framework that bridges high-level deliberation with grounded physical rollouts. SCOUT leverages an object-centric scene representation to reconstruct hidden geometry and derive a relational planning state capturing blocker hierarchies and accessibility corridors. To navigate the vast action space in clutter, an LLM proposes structured action candidates which are then rigorously evaluated via Monte Carlo Tree Search (MCTS) using a learned action-conditioned world model. Evaluations on 'Reveal' and 'Placement' tasks show that SCOUT outperforms Vision-Language-Action (VLA) policies, embodied agents, and 3D world models, with significant gains in long-horizon tasks requiring reasoning over occluded geometry. We further validate the framework's robustness through real-world robotic manipulation in a cluttered shelf environment.

1 Introduction

Manipulation in cluttered, occluded spaces is fundamentally difficult because the objects and free space that matter most are often not directly visible [16, 40]. A target may be buried behind front-row blockers in a shelf, reachable only through a narrow corridor that is itself occluded or only after several intermediate rearrangements [14, 3]. Reasoning about these unobservable spaces to manipulate in clutter requires a robot to move beyond purely reactive visual feedback, necessitating deliberation across possible environment and object layouts and their intricate relationships among each other.

A natural way to enable such deliberation is to plan in belief space, where the robot represents unobserved scene structure and reasons over possible hidden configurations [24]. For occluded manipulation, such a belief must capture more than the mere existence of hidden objects. The planner must also reason about which object is likely to block the target and where free space may exist around it. This suggests an explicit scene representation that captures object-level geometry and spatial structure, rather than only symbolic hidden-state hypotheses. Moreover, representing the hidden scene is only part of the problem. To manipulate in clutter, the robot must also reason about how that scene representation will evolve under candidate actions. It therefore needs a world model that predicts how an action reveals the target, creates free space, or simply changes the occlusion pattern [19].

Recent progress in vision-language and foundation-model-based robotics has improved action selection from observed images and led to strong results in semantic grounding, skill selection, program synthesis, and visually guided planning [23, 29, 38, 22, 33, 30]. These methods are effective when the relevant objects and affordances are visible, showing that large models can provide useful high-level structure for robotic decision making. World models and search offer a complementary route

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

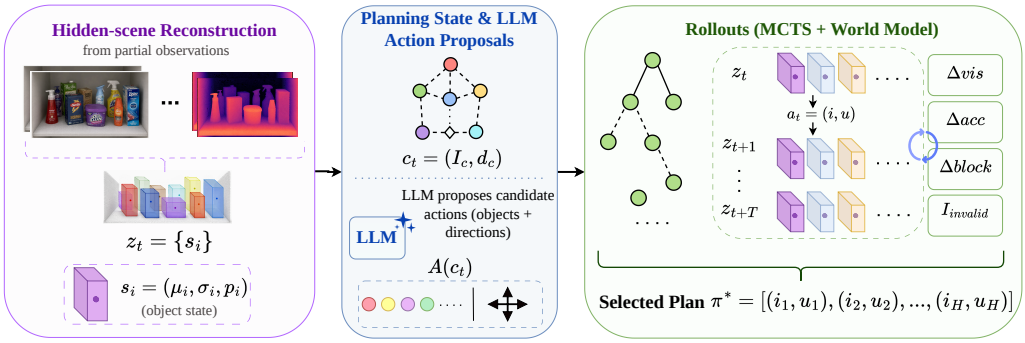

Figure 1: Overview of SCOUT. From partial RGB-D observations, GHOST reconstructs an object-centric hidden scene representation, which is converted into a relational planning state. An LLM generates structured action proposals, and MCTS with a learned world model evaluates simulated rollouts to guide selection of a multi-step plan.

by supporting imagination-based planning over candidate futures [37, 20, 41, 43, 2], and recent language-guided search methods combine Large Language Models (LLMs) with search to improve long-horizon decision making [9, 13]. However, these approaches still largely operate over the currently observed scene or over abstract state representations that do not explicitly capture the hidden 3D structure created by physical clutter, which are essential for occluded manipulation.

We therefore present SCOUT, a planning framework for manipulation under occlusion that combines hidden-scene reconstruction, action-conditioned prediction, and LLM-based deliberation. Given a small number of RGB-D observations from the frontal viewpoints, our reconstruction module, GHOST (Geometry-aware Hallucination for Occluded Scene Topology), produces an object-centric scene representation capturing geometry and occupancy beyond the observed view. A learned world model predicts how this representation evolves under discrete object-level actions [19]. An LLM proposes structured action candidates, and a Monte Carlo Tree Search (MCTS) [6] module evaluates simulated rollouts under the learned world model. The next primitive is selected via grounded search, and planning repeats from the updated state. We evaluate the method on cluttered manipulation tasks in which targets are initially occluded, partially observable, and require multi-step rearrangement to access [16, 40]. In these settings, we find that planning over reconstructed scene structure and simulated futures is more effective than relying on visible-only reasoning or one-shot language predictions alone.

Our contributions are:

- A structured hidden-scene representation for occluded manipulation from sparse frontal RGB-D views.
- A planning method that reasons over this representation using a learned world model and LLM-guided search.
- Simulation and real-robot validation on cluttered multi-step rearrangement tasks.

2 Related Work

Planning under Occlusion with Hidden-Scene Models. Planning under occlusion is classically a belief-space problem [25]. In embodied AI, recent work has addressed hidden targets and incomplete maps through semantic priors, diffusion-based planning, and uncertainty-aware task-and-motion planning. PONI predicts potential locations of unseen targets from partial semantic maps [36], while value-guided diffusion planning explicitly targets navigation under partial observability through multi-step candidate generation and best-plan selection [44]. TAMPURA and related partially observable TAMP methods reason under action uncertainty using symbolic belief representations and information-gathering strategies [12, 25], with extensions incorporating LLM-guided state estimation over hidden objects [26]. Occluded planning further depends on reconstructing hidden 3D structure from limited observations. Prior work in semantic scene completion and open-world 3D scene understanding reconstructs geometry and semantics beyond the field of view [8, 28, 34, 1]. Spatially

aware robot foundation models show that explicit geometric alignment improves manipulation quality and generalization [35, 27], while recent work leverages robot trajectories and test-time adaptation to improve action semantics and value estimation [11, 45]. In contrast, we address planning under occlusion in cluttered 3D scenes by reasoning over a reconstructed hidden scene and using an action-conditioned world model to evaluate how different actions reveal previously unobserved structure.

Language and vision foundation models for robotics. Foundation-model approaches have enabled robots to select skills, synthesize code, and plan trajectories from language and images [23, 29, 38, 22]. PIVOT shows that iterative visual prompting can substantially improve spatial decision-making by repeatedly refining proposals rather than committing to one-shot predictions [33]. OK-Robot demonstrates how open-knowledge components can be composed into a robust mobile-manipulation pipeline, while recent analysis highlights the importance of spatial grounding and operational detail when integrating such modules [30]. Our method differs from these works in two key respects: it explicitly reasons over a hidden-scene belief rather than only visible observations, and it uses the language model as a *search prior*, while the world model and MCTS perform physical evaluation and final action selection.

World models and search for embodied planning. Learned world models have proved effective for planning and policy improvement in games and control, including MuZero, Dreamer, and DayDreamer [37, 20, 41]. More recent simulators and generative world models expand this perspective to real-world video and navigation [43, 2]. Language-guided search has also been explored for object rearrangement and code world-model generation [9, 13]. Our goal is not to simply attach an LLM to search. Instead, the LLM narrows the object-action space using the relational planning state, while MCTS and the learned world model evaluate candidate futures and select the final plan from grounded rollout evidence.

3 Method

We address long-horizon manipulation under partial observability by combining an object-centric scene representation, a learned world model, and an LLM that generates structured action proposals for planning. Our formulation explicitly separates (i) action proposal generation from the relational planning state, from (ii) physical feasibility and multi-step outcome prediction, handled by the learned world model. Fig. 1 provides an overview of SCOUT, from hidden-scene reconstruction and LLM-guided action proposals to world-model rollouts and final plan selection. The complete execution flow of the system is detailed in Algorithm 1.

3.1 Problem Formulation

We consider manipulation under partial observability, where an agent must plan a sequence of object-level actions from a limited set of observations. Each instance is defined by a hidden scene state s, a goal specification g, and a finite action budget T. The agent receives a set of RGB-D observations $o = \{o^{(1)}, \ldots, o^{(V)}\}$ captured from frontal viewpoints, providing partial visibility of objects, free space, and access corridors. The objective is to compute a finite-horizon action sequence $\mathbf{a}_{1:T} = (a_1, \ldots, a_T)$, where $a_t \in \mathcal{A}$, that achieves the task goal. The action space $\mathcal{A}$ consists of object-centric manipulation primitives $a = (i, u)$, where i indexes the manipulated object and $u \in \{\texttt{shift_left}, \texttt{shift_right}, \texttt{shift_back}, \texttt{pull_forward}\}$. In the initial-view protocol, the scene is reconstructed once from o, and subsequent planning states are updated using the world model without re-observation.

We consider four task families instantiated from a shared distribution of cluttered shelf scenes: **Reveal**, **Reveal_Multi**, **Place**, and **Place_Multi**. Each task defines a binary success condition evaluated at the end of the sequence. Reveal tasks require the target to be visible and accessible, while placement tasks require the object to lie within a goal region, with Reveal_Multi and Place_Multi being the counterparts over multiple objects.

The system operates over a reconstructed scene representation z_0 obtained from observations, and a derived planning state x_0 capturing task-relevant structure (Sec. 3.2). A learned world model predicts state transitions $z_{t+1} = f_\phi(z_t, a_t)$ (Sec. 3.3), enabling simulation of action-conditioned future states.

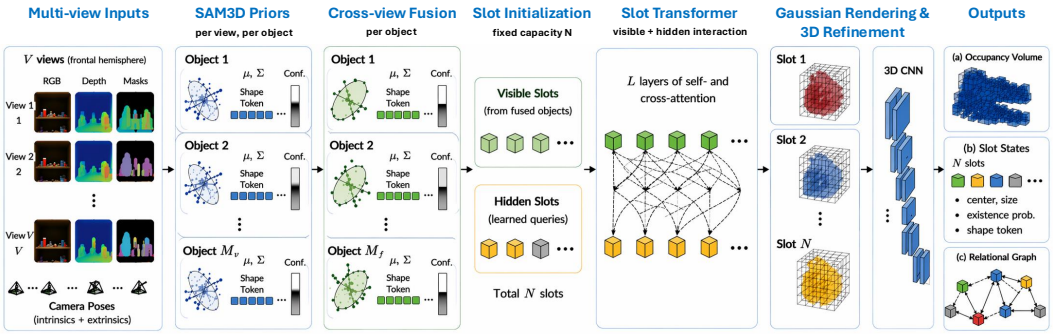

Figure 2: GHOST scene representation pipeline. Multi-view RGB-D observations are fused into object-centric slots, refined via slot interactions, and rendered into a unified 3D scene representation.

The planner can be viewed as a mapping $\Pi : (o, g) \to \mathbf{a}_{1:T}$, which uses the reconstructed scene representation, LLM action proposals, and world-model predictions to evaluate candidate action sequences up to horizon T. The objective is to select a plan that maximizes task success:

$$\pi^* = \underset{\mathbf{a}_{1:T} \in \mathcal{A}^T}{\operatorname{argmax}} \ \mathbb{E}[\text{Success}(z_T, g)]. \tag{1}$$

3.2 Object-Centric Scene Representation

Given observations o, we construct an object-centric state representation $z_t = \textsc{Ghost}(o)$, where $z_t = \{s_i^t\}_{i=1}^K$ and each slot $s_i^t = (\mu_i^t, \sigma_i^t, p_i^t)$ encodes a hypothesized object instance with center $\mu_i^t \in \mathbb{R}^3$, 3D size $\sigma_i^t \in \mathbb{R}^3$, and existence probability $p_i^t \in [0, 1]$.

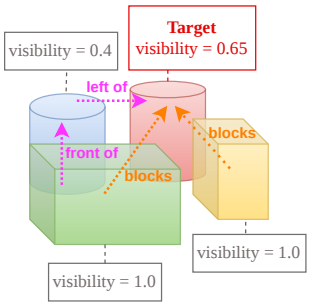

Figure 3: Relational planning state derived from object-centric slots.

As shown in Fig. 2, GHOST takes RGB-D views from the front-facing viewpoints, together with camera intrinsics and extrinsics. For each view, SAM3D [39] provides object-level priors from segmented object regions, including coarse 3D geometry estimated from the mask and depth, a confidence score indicating how reliable the object hypothesis is, and a compact shape token that summarizes the object's local appearance and geometry in a learned feature vector. These per-view object hypotheses are fused across views into object-level estimates with uncertainty, producing visible slots for objects supported by observations. To account for occluded structure, GHOST also maintains a fixed-capacity set of hidden slots initialized as learned queries. The visible and hidden slots are jointly refined using a slot transformer with self-attention and cross-attention, allowing observed objects and latent hypotheses to exchange spatial context.

The refined slots are rendered through a Gaussian-based 3D refinement module to produce an occupancy volume, updated slot states, and a relational graph. To support reasoning about task structure, we derive a relational planning state $x_t = \mathcal{P}(z_t, g)$ from the reconstructed slot set z_t and the goal g, as illustrated in Fig. 3. Each of the K slots contains 7 values: a 3D center, a 3D size, and an existence probability. The resulting state x_t includes (i) the target object index, (ii) blocker relationships indicating which objects obstruct access to the target, (iii) visibility indicators, and (iv) coarse spatial relations. These relations are computed deterministically from slot geometry using geometric predicates. The planner operates on x_t, while the world model evolves z_t; the mapping $\mathcal{P}$ serves as the interface between perception and planning.

3.3 Action-Conditioned World Model Dynamics

We learn a dynamics model $z_{t+1} = f_\phi(z_t, a_t)$. The model predicts per-object updates $\{(\Delta\mu_i, m_i, v_i)\}_{i=1}^K$, where $\Delta\mu_i \in \mathbb{R}^3$ is the displacement, $m_i \in \{0, 1\}$ indicates whether object i moves, and $v_i \in \{0, 1\}$ denotes whether the predicted update is physically valid. The next state is obtained by $\mu_i^{t+1} = \mu_i^t + m_i \cdot \Delta\mu_i$, while σ_i and p_i remain unchanged. This abstraction targets

rigid-object, no-add/remove primitives; orientation is not explicitly represented, and visibility is recomputed from geometry. The world model is trained in two phases: an initial phase on simulated object trajectories, followed by a fine-tuning phase on GHOST-reconstructed scene representations. During fine-tuning, hallucinated slots (those without a matched ground-truth object) are supervised to produce zero displacement, improving robustness to reconstruction noise.

3.4 LLM-Guided Action Proposals

We use the LLM to generate a structured set of action proposals from the task goal g and initial planning state x_0. The LLM is instructed to read g and x_0, then propose task-relevant object subsets and coarse motion directions; feasibility and final plan selection are left to the world model and MCTS. Each proposal is parameterized as $c = (\mathcal{I}_c, d_c)$, where $\mathcal{I}_c \subseteq \{1, \ldots, K\}$ is a subset of object indices and $d_c \in \{\texttt{left}, \texttt{right}, \texttt{back}, \texttt{forward}\}$ is a motion direction. Each proposal $c = (\mathcal{I}_c, d_c)$ restricts search to a smaller candidate action set. Here, $\mathcal{I}_c$ specifies the subset of objects to consider, and d_c specifies the preferred motion direction. We therefore construct

$$\mathcal{A}(c) = \{(i, d_c) \mid i \in \mathcal{I}_c\} \cup \mathcal{A}_{\text{aux}}, \qquad (2)$$

where $\mathcal{A}_{\text{aux}}$ contains a small set of fallback actions generated at each root from geometric predicates in the current relational state, such as pulling the target-facing blocker forward or shifting the nearest corridor blocker laterally. These auxiliary actions prevent the proposal set from becoming overly narrow when the LLM omits a useful object or direction. If multiple proposals are generated, we take the union of their candidate sets. MCTS then searches over this reduced action space instead of all object-primitive combinations.

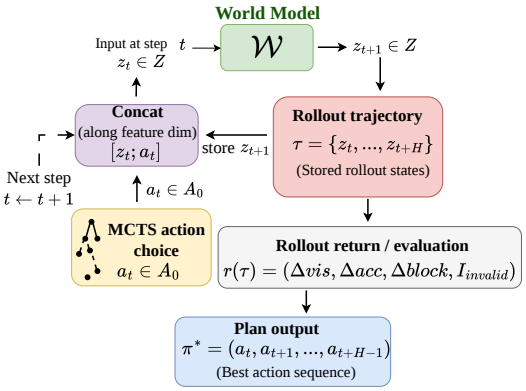

Figure 4: **WM-Guided Planning:** Starting from the reconstructed scene state, future states are recursively predicted by the action-conditioned world model during MCTS rollouts, and the resulting summaries guide final action selection.

3.5 Grounded Selection and Search

Given the candidate action set, we simulate future trajectories τ using the world model, as illustrated in Fig. 4. Each rollout is summarized by $r(\tau) = \big(\Delta\text{vis}(\tau),\ \Delta\text{acc}(\tau),\ \Delta\text{block}(\tau),\ \mathbb{I}_{\text{invalid}}(\tau)\big)$, where Δvis measures visibility gain, Δacc measures accessibility progress, and $\mathbb{I}_{\text{invalid}}$ flags physically invalid transitions. These summaries are produced by an MCTS procedure that constructs a search tree over predicted scene states using the learned world model. At each decision, MCTS evaluates depth-$H = 2$ rollouts and commits only the first primitive of the highest-scoring branch. The root state is updated and the tree rebuilt until predicted success or the $T = 12$ action budget, enabling long-horizon task completion through receding-horizon planning. The initial LLM proposals are retained throughout. In Algorithm 1, rollout indices are local to each search, z_0 and x_0 denote the current root, and $\text{Success}(x_0, g)$ denotes predicted success. In the closed-loop variant, new RGB-D observations refresh z_0 at the specified sensing interval before recomputing x_0.

Algorithm 1 SCOUT Planning

Require: observations o, task goal g, planning horizon H, learned dynamics f_ϕ, action budget T
Ensure: Selected action sequence π^*

$z_0 \leftarrow \text{GHOST}(o)$; $\pi^* \leftarrow [\,]$
$x_0 \leftarrow \mathcal{P}(z_0, g)$
$\mathcal{C} \leftarrow \text{LLM}(x_0)$
while $|\pi^*| < T$ and $\neg\text{Success}(x_0, g)$ **do**
  $\mathcal{A}_0 \leftarrow \bigcup_{c \in \mathcal{C}} \mathcal{A}(c)$
  Initialize search tree $\mathcal{T}$ with root node z_0
  while within search budget **do**
    $\tau \leftarrow \text{SelectAndExpand}(\mathcal{T}, \mathcal{A}_0)$
    for $t = 0$ to $H - 1$ **do**
      $z_{t+1} \leftarrow f_\phi(z_t, a_t)$
    end for
    $r(\tau) \leftarrow \text{Summarize}(\tau)$
    $\text{Backpropagate}(r(\tau), \mathcal{T})$
  end while
  $\hat{\pi} \leftarrow \text{argmax}_{\mathbf{a}_{1:H} \in \mathcal{T}} \mathbb{E}[\text{Success}(z_H, g)]$
  $a \leftarrow \hat{\pi}_1$; execute a; append a to π^*
  $z_0 \leftarrow f_\phi(z_0, a)$
  $x_0 \leftarrow \mathcal{P}(z_0, g)$
end while
return π^*

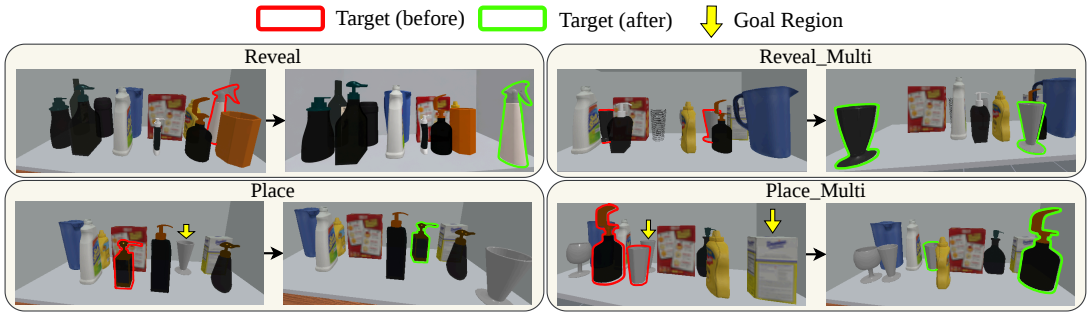

Figure 5: Representative examples from the four task families used for evaluation.

4 Experiments

We conduct a comprehensive set of experiments to evaluate the effectiveness, generality, and robustness of SCOUT for manipulation under occlusion. Our evaluation is organized around four questions: 1) Does SCOUT improve task success in cluttered, partially observable manipulation? 2) How well does it generalize to unseen objects, view layouts, and harder occlusion settings? 3) Which components of the planner contribute most to performance? 4) Can our method successfully transfer to the real world?

Task suite. We evaluate SCOUT on a cluttered shelf benchmark designed to isolate the hidden-scene planning problem under controlled occlusion. Scenes are generated in ManiSkill2 [18] using a procedurally constructed shelf and object assets from the ManiSkill2-YCB set [7], ProcTHOR [15], and RoboCasa [32]. Each scene contains 8–16 objects, including task objects, foreground blockers, and distractors; object poses are sampled so that the target object or placement region is initially occluded from the frontal view. In the initial-view protocol, each scene is observed from a small set of RGB-D viewpoints sampled around the shelf opening, and the planner must produce a finite-horizon sequence of object-centric primitives from these sparse observations alone. We evaluate the four task families introduced in Sec. 3.1 and illustrated in Fig. 5: **Reveal**, **Reveal_Multi**, **Place**, and **Place_Multi**. Additional details on shelf generation, object assets, camera sampling, and task construction are provided in Section A.1.

Implementation Details. We use Qwen2.5-7B-Instruct [42] as the open-source action-proposal model in SCOUT. We set the rollout scoring weights to $\alpha_{\text{vis}} = 1.0$, $\alpha_{\text{acc}} = 1.0$, $\alpha_{\text{block}} = 0.25$, and $\alpha_{\text{invalid}} = 1.0$. All learned components are trained with AdamW [31] on simulator-generated shelf data. Experiments are conducted on four NVIDIA L40S-48GB GPUs with 64 CPU cores and 512GB RAM; full architecture, hyperparameter, compute, and inference details are provided in Section A.

Baselines. For the initial-view comparison, we adapt four reference methods to the same sparse-observation, no-reobservation, and discrete-primitive setting. The VLA comparisons characterize performance under this shared interface rather than native continuous-control performance.

- $\pi_{0.5}$ [4] is an end-to-end vision-language-action policy designed for open-world robot manipulation, mapping visual observations and task specifications to continuous low-level robot actions. For fair comparison, we provide the same initial RGB-D views and a templated task specification, decode each predicted action to the nearest valid primitive and execute the resulting primitive sequence under the same no-reobservation protocol as SCOUT.
- Long-VLA [17] is a long-horizon VLA policy designed to extend vision-language-action models to multi-step manipulation. Since it originally produces continuous robot actions, we evaluate it with the same initial observation set and task specification as SCOUT, map its outputs to the shared primitive vocabulary, and impose the same $T = 12$ action budget.
- OWMM-Agent [10] is an embodied agent that combines scene understanding, state tracking, and action generation for open-world mobile manipulation. To adapt it to our stationary shelf benchmark, we disable navigation and auxiliary tool use, provide only the same initial RGB-D observations, and require all decisions to be emitted through the same object-centric primitive API.

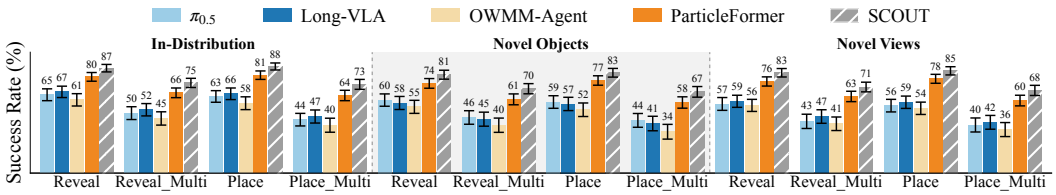

Figure 6: Results on the simulated shelf task suite. Success rate (in %) is averaged across 3 seeds and uncertainty is given by the standard error. SCOUT consistently outperforms the adapted baselines across in-distribution, novel-object, and novel-view settings.

- ParticleFormer [21] is a 3D point-cloud world model for predicting action-conditioned scene evolution rather than a complete high-level planner. We therefore use it as a drop-in replacement for SCOUT's learned transition model, while keeping the reconstructed planning state, candidate action set, rollout objective, and search budget fixed.

Evaluation Metrics. We evaluate performance using task-completion success rates over 500 independent rollouts with randomly initialized shelf configurations. Since SCOUT outputs high-level manipulation primitives rather than low-level controls, we measure trajectory length in primitive actions and impose a maximum budget of $T = 12$ actions per episode. Reported success is computed from simulator ground-truth geometry: Reveal rollouts are counted as successful if the target object's visibility and accessibility scores both exceed 0.6 within T actions, while Place rollouts are successful if the specified object reaches its assigned goal region within T actions. During planning, these quantities are estimated from the relational planning state $x_t = P(z_t, g)$ for rollout scoring. In addition to success rate, we report the mean number of primitive actions per successful episode, target visibility gain, target accessibility gain, and invalid-action rate. The visibility score is computed from a fixed front-facing evaluation camera. For Reveal tasks, it measures how much of the target object is visible; for Place tasks, it measures how much of the goal region is visible, normalized by its empty-shelf projection. The accessibility score measures whether the task site is reachable from the shelf opening: it is high when the path to the target object or goal region is clear, and low when other objects block this path. Actions are marked invalid when the predicted transition violates geometric feasibility constraints. Metric construction details are provided in Section A.3.

4.1 Simulation Results

Figure 6 shows the success rates for the simulation tasks under in-distribution, novel-object, and novel-view settings. SCOUT achieves the highest success rate across all task families and distributions, reaching an in-distribution average success of 80.8% compared with 72.8% for the strongest baseline. The gains are largest in the multi-object settings, where SCOUT improves over the strongest baseline by 9.0 points on Reveal_Multi and 9.0 points on Place_Multi. These settings require reasoning over hidden blockers, intermediate rearrangements, and narrow access corridors. The same trend holds in the single-object tasks, where SCOUT reaches 86.8% on Reveal and 88.2% on Place, improving over the strongest baseline by 7.2 points in each case. These results use the initial-view protocol; Appendix A.7 reports closed-loop comparisons and additional proposal, search, calibration, and reconstruction diagnostics.

Generalization Performance. SCOUT remains strong under distribution shift. On novel objects, it reaches an average success of 75.4%, compared with 67.6% for the strongest baseline. On novel views, it reaches 76.8%, compared with 69.3%. The smaller degradation under viewpoint and object shifts suggests that planning over reconstructed object-centric structure provides a stable basis for manipulation under occlusion.

Planning Efficiency. The plan-level diagnostics in Table 1 show that successful SCOUT rollouts are shorter in the single-object settings and longer in the multi-object settings. Reveal and Place require 6.3 and 6.0 primitive actions on average, whereas Reveal_Multi and Place_Multi require 8.7 and 9.1. The multi-object tasks also produce

Table 1: Trajectory rollout metrics.

Task	Steps	Δ Visibility	Δ Access.	Invalid Rate
Reveal	6.3±1.4	+0.46±0.07	+0.41±0.06	3.8%
Reveal_Multi	8.7±1.8	+0.59±0.09	+0.53±0.08	5.9%
Place	6.0±1.2	+0.34±0.06	+0.37±0.05	3.2%
Place_Multi	9.1±2.0	+0.51±0.08	+0.49±0.07	6.4%

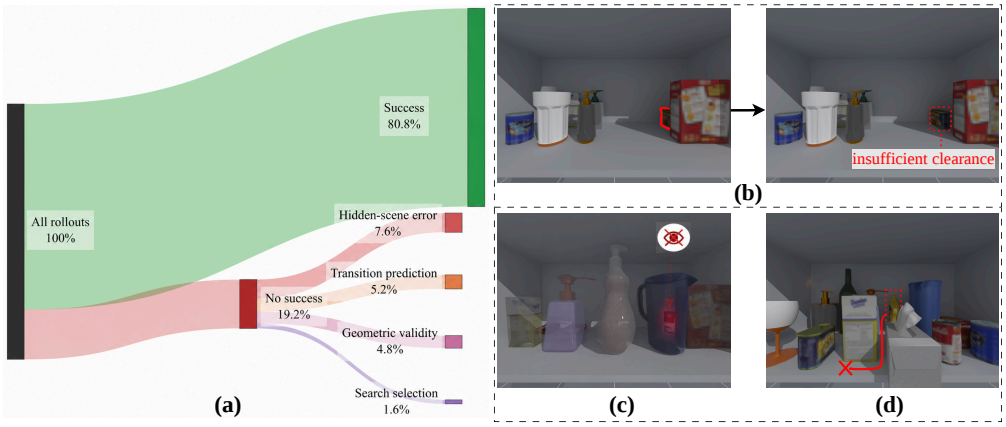

Figure 7: **Failure analysis of SCOUT. (a)** Failure decomposition as percentages of all rollouts. Most failures stem from hidden-scene reconstruction errors, followed by transition-prediction and geometric-validity errors. **(b)** Transition-prediction failure: the selected action does not create enough free space to retrieve the target. **(c)** Hidden-scene failure: a small back-row target remains too occluded to be reconstructed from partial observations. **(d)** Geometric-validity failure: dense clutter leaves no feasible retrieval corridor under the current primitive set.

larger visibility and accessibility gains, reflecting that more occlusion and access constraints must be cleared before the target or goal region becomes usable. Invalid-action rates remain low across all four families, ranging from 3.2% to 6.4%, indicating that search rarely selects geometrically infeasible primitives.

4.2 Failure Analysis and Qualitative Results

Failure Modes. To analyze where unsuccessful rollouts break down, we categorize failures in Fig. 7a into four sources: hidden-scene errors, transition-prediction errors, geometric-validity errors, and search-selection errors. The largest source is hidden-scene error, accounting for 7.6% of all rollouts, where the reconstructed state misses a blocker or overestimates the free corridor to the task site. Transition-prediction errors account for 5.2%, typically when the rollout predicts that an action will clear visibility or access but the executed transition leaves the site partially blocked. Geometric-validity failures contribute 4.8%, while search-selection errors are the smallest category at 1.6%, indicating that most failures arise before final action selection rather than from the scoring rule itself.

Panels (b)–(d) show representative failure cases behind the decomposition in Fig. 7. In (b), the world model predicts that the selected action will create enough clearance, but execution leaves the target only partially exposed and still not retrievable. In (c), the target is too small and occluded in the back row, so it is not recovered by the hidden-scene reconstruction. In (d), the scene is so congested that the available shift and pull primitives cannot open a feasible access corridor without an explicit remove action.

Scaling with Clutter Density. In Fig. 8, we evaluate how task success changes as the number of shelf objects increases from 8 to 16. Performance decreases gradually across all task families, showing that denser clutter makes occlusion reasoning and rearrangement more difficult. The drop is mild for single-object Reveal and Place tasks, while Reveal_Multi and Place_Multi degrade more sharply as additional objects introduce more blockers and longer rearrangement chains. Even at 16 objects, SCOUT maintains reasonable success, indicating that the planner remains effective under high clutter.

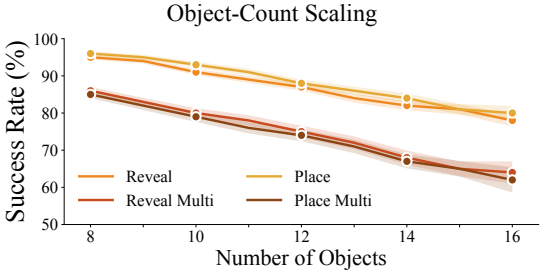

Figure 8: Success rate as clutter density increases.



Figure 9: Real-world rollouts of SCOUT on the REVEAL and PLACE tasks in cluttered shelf scenes.

Table 2: Ablation study of SCOUT components.

Variant	Avg. Succ. (%)	Avg. Act.
Single-view input	69.7±4.6	7.9±0.5
w/o relational projection	68.1±4.4	8.2±0.5
w/o rollout search	70.6±4.2	6.8±0.4
Full SCOUT	80.8±3.5	7.5±0.4

Ablation Study. Table 2 shows that each component contributes to final performance. Using a single input view reduces average success to 69.7%, indicating that multi-view observations improve the reconstructed scene state under occlusion. Removing the relational projection gives the lowest success, 68.1%, and also increases the average plan length to 8.2 actions, showing that explicit task-site, blocker, and spatial relations are important for efficient search. Replacing rollout search with greedy depth-1 selection reaches 70.6% success and uses fewer actions on average, but the shorter plans are less reliable because they do not evaluate longer rearrangement effects. The full model achieves the best success, 80.8%, while maintaining a moderate action length of 7.5 primitives.

4.3 Real-World Experiments

To evaluate real-world performance, we deploy SCOUT on a Boston Dynamics Spot platform [5] equipped with a 6-DOF manipulator and its integrated gripper-mounted 4K RGB and Time-of-Flight (ToF) depth sensors. Observations are captured from static frontal viewpoints to maintain simulation-consistent partial visibility, with action candidates executed via a kinematic motion controller. We perform 10 trials each for the **Reveal** and **Place** tasks using a variety of household objects on a cluttered shelf.

Across these 20 trials, SCOUT achieves a **60% (12/20)** success rate. As illustrated in Fig. 9, the system successfully reasons over hidden topology to plan strategic rearrangements, such as shifting a blocking container (**orange arrows**) to expose an occluded target (**red to green highlight**) or clearing an access corridor to reach a goal region (**yellow arrow**). Failures were distributed between low-level hardware issues, such as gripper slippage on compliant surfaces, and plan-level errors. The latter primarily involved the world model proposing physically invalid actions or MCTS rollouts failing due to noisy GHOST reconstructions in high-glare environments. Detailed hardware specifications and individual failure case analysis in Section A.8.

5 Conclusion and Limitations

In this work, we presented SCOUT, a planning framework for manipulation under partial observability that integrates hidden-scene reconstruction, object-centric action abstraction, and world-model-guided rollout evaluation. SCOUT enables structured reasoning over occluded environments by combining LLM-guided action proposals with grounded evaluation of action-conditioned futures. We show that this approach improves task success across cluttered shelf manipulation settings, including multi-step rearrangement and generalization to unseen configurations. The framework operates from sparse RGB-D observations and a discrete primitive interface, supporting consistent behavior across simulation and real-world deployment.

Broader Impact. SCOUT improves the ability of robotic systems to operate in cluttered and partially observable environments, with potential applications in warehouse automation, assistive robotics, and service tasks. By reasoning about hidden objects and access corridors, it can reduce manipulation errors and improve reliability in real-world settings. The structured planning interface may also support more interpretable decision-making in human-robot interaction scenarios.

Limitations. Several limitations remain. The performance of SCOUT depends on the accuracy of hidden-scene reconstruction, and errors in estimated geometry, object existence, or blocker relationships can propagate through planning. The learned world model may produce inaccurate predictions under complex contact interactions or out-of-distribution object arrangements. In addition, the current formulation assumes a static scene during planning and relies on a fixed set of discrete manipulation primitives, which may limit applicability to dynamic environments or tasks requiring continuous contact-rich control.

Acknowledgments and Disclosure of Funding

This work has taken place in the RobotiXX Laboratory at George Mason University. RobotiXX research is supported by National Science Foundation (NSF, 2350352), Army Research Office (ARO, W911NF2320004, W911NF2520011), Army Ground Vehicle Systems Center (GVSC), Google DeepMind (GDM), Microsoft Research (MSR), Clearpath Robotics, FrodoBots Lab, Raytheon Technologies (RTX), Tangenta, 4-VA, Mason Innovation Exchange (MIX), and Walmart. The authors declare no competing interests.

References

[1] Aditya Agarwal, Gaurav Singh, Bipasha Sen, Tomas Lozano-Perez, and Leslie Pack Kaelbling. Scenecomplete: Open-world 3d scene completion in cluttered real world environments for robot manipulation. *arXiv preprint arXiv:2410.23643*, 2024.

[2] Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation world models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

[3] Wissam Bejjani, Wisdom C. Agboh, Mehmet R. Dogar, and Matteo Leonetti. Occlusion-aware search for object retrieval in clutter. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4678–4685, 2021. doi: 10.1109/IROS51168.2021.9636230. URL `https://doi.org/10.1109/IROS51168.2021.9636230`.

[4] Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, et al. $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization. In *Proceedings of The 9th Conference on Robot Learning*, 2025. URL `https://openreview.net/forum?id=vlhoswksBO`. CoRL 2025 Oral.

[5] *Spot Arm Specifications*. Boston Dynamics, 2021. URL `https://support.bostondynamics.com/s/article/Spot-Arm-Specifications-151694`. Accessed: 2026-05-06.

[6] Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012.

[7] Berk Calli, Aaron Walsman, Arjun Singh, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M. Dollar. Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set. *IEEE Robotics & Automation Magazine*, 22(3):36–52, 2015. doi: 10.1109/MRA.2015.2448951.

[8] Anh-Quan Cao and Raoul de Charette. Monoscene: Monocular 3d semantic scene completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[9] Haonan Chang, Kai Gao, Kowndinya Boyalakuntla, Alex Lee, Baichuan Huang, Harish Udhaya Kumar, Jinjin Yu, and Abdeslam Boularias. Lgmcts: Language-guided monte-carlo tree search for executable semantic object rearrangement. *arXiv preprint arXiv:2309.15821*, 2024.

[10] Junting Chen, Haotian Liang, Lingxiao Du, Weiyun Wang, Mengkang Hu, Yao Mu, Wenhai Wang, Jifeng Dai, Ping Luo, Wenqi Shao, and Lin Shao. Owmm-agent: Open world mobile manipulation with multi-modal agentic data synthesis. In *Advances in Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=vSLzoUoJt6. NeurIPS 2025 Poster.

[11] Kaiyuan Chen, Shuangyu Xie, Zehan Ma, and Ken Goldberg. Robo2vlm: Visual question answering from large-scale in-the-wild robot manipulation datasets. *arXiv preprint arXiv:2505.15517*, 2025.

[12] Aidan Curtis, George Matheos, Nishad Gothoskar, Vikash Mansinghka, Joshua Tenenbaum, Tomas Lozano-Perez, and Leslie Pack Kaelbling. Partially observable task and motion planning with uncertainty and risk awareness. *arXiv preprint arXiv:2403.10454*, 2024.

[13] Nicola Dainese, Matteo Merler, Minttu Alakuijala, and Pekka Marttinen. Generating code world models with large language models guided by monte carlo tree search. In *Advances in Neural Information Processing Systems*, 2024.

[14] Michael Danielczuk, Andrey Kurenkov, Ashwin Balakrishna, Matthew Matl, David Wang, Roberto Martín-Martín, Animesh Garg, Silvio Savarese, and Ken Goldberg. Mechanical search: Multi-step retrieval of a target object occluded by clutter. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1614–1621, 2019. doi: 10.1109/ICRA.2019.8794143. URL https://doi.org/10.1109/ICRA.2019.8794143.

[15] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Kiana Ehsani, Jordi Salvador, Winson Han, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. ProcTHOR: Large-scale embodied ai using procedural generation. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

[16] Mehmet R. Dogar and Siddhartha S. Srinivasa. A planning framework for non-prehensile manipulation under clutter and uncertainty. *Autonomous Robots*, 33(3):217–236, 2012. doi: 10.1007/s10514-012-9306-z.

[17] Yiguo Fan, Shuanghao Bai, Xinyang Tong, Pengxiang Ding, Yuyang Zhu, Hongchao Lu, Fengqi Dai, Wei Zhao, Yang Liu, Siteng Huang, Zhaoxin Fan, Badong Chen, and Donglin Wang. Long-vla: Unleashing long-horizon capability of vision language action model for robot manipulation. In *Proceedings of The 9th Conference on Robot Learning*, 2025. URL https://openreview.net/forum?id=irh5o90Mj1. CoRL 2025 Poster.

[18] Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, Xiaodi Yuan, Pengwei Xie, Zhiao Huang, Rui Chen, and Hao Su. Maniskill2: A unified benchmark for generalizable manipulation skills. In *International Conference on Learning Representations (ICLR)*, 2023.

[19] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2555–2565. PMLR, 2019. URL https://proceedings.mlr.press/v97/hafner19a.html.

[20] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640:647–653, 2025. doi: 10.1038/s41586-025-08744-2. URL https://www.nature.com/articles/s41586-025-08744-2.

[21] Suning Huang, Qianzhong Chen, Xiaohan Zhang, Jiankai Sun, and Mac Schwager. Particleformer: A 3d point cloud world model for multi-object, multi-material robotic manipulation. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 4941–4957. PMLR, 2025. URL https://proceedings.mlr.press/v305/huang25c.html.

[22] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 540–562. PMLR, 2023. URL `https://proceedings.mlr.press/v229/huang23b.html`.

[23] Brian Ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander T. Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Mengyuan Yan, Noah Brown, Michael Ahn, Omar Cortes, Nicolas Sievers, Clayton Tan, Sichun Xu, Diego Reyes, Jarek Rettinghouse, Jornell Quiambao, Peter Pastor, Linda Luu, Kuang-Huei Lee, Yuheng Kuang, Sally Jesmonth, Nikhil J. Joshi, Kyle Jeffrey, Rosario Jauregui Ruano, Jasmine Hsu, Keerthana Gopalakrishnan, Byron David, Andy Zeng, and Chuyuan Kelly Fu. Do as i can, not as i say: Grounding language in robotic affordances. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 287–318. PMLR, 2023. URL `https://proceedings.mlr.press/v205/ichter23a.html`.

[24] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Integrated task and motion planning in belief space. *The International Journal of Robotics Research*, 32(9–10):1194–1227, 2013. doi: 10.1177/0278364913484072.

[25] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998.

[26] Yoonwoo Kim, Raghav Arora, Roberto Martin-Martin, Peter Stone, Ben Abbatematteo, and Yoonchang Sung. Large-language-model-guided state estimation for partially observable task and motion planning. *arXiv preprint arXiv:2603.03704*, 2026.

[27] Fuhao Li, Wenxuan Song, Han Zhao, Jingbo Wang, Pengxiang Ding, Donglin Wang, Long Zeng, and Haoang Li. Spatial forcing: Implicit spatial representation alignment for vision-language-action model. *arXiv preprint arXiv:2510.12276*, 2025.

[28] Yiming Li, Zetong Yu, Christopher Choy, Chaowei Xiao, Jose M. Alvarez, Sanja Fidler, Chen Feng, and Anima Anandkumar. Voxformer: Sparse voxel transformer for camera-based 3d semantic scene completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[29] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500, 2023. doi: 10.1109/ICRA48891.2023.10160591.

[30] Peiqi Liu, Yaswanth Orru, Jay Vakil, Chris Paxton, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Ok-robot: What really matters in integrating open-knowledge models for robotics. *arXiv preprint arXiv:2401.12202*, 2024.

[31] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

[32] Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*, 2024.

[33] Soroush Nasiriany et al. Pivot: Iterative visual prompting elicits actionable knowledge for vlms. *arXiv preprint arXiv:2402.07872*, 2024.

[34] Phuc D. A. Nguyen, Tuan Duc Ngo, Evangelos Kalogerakis, Chuang Gan, Anh Tran, Cuong Pham, and Khoi Nguyen. Open3dis: Open-vocabulary 3d instance segmentation with 2d mask guidance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. doi: 10.1109/CVPR52733.2024.00385.

[35] Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, Jiayuan Gu, Bin Zhao, Dong Wang, and Xuelong Li. Spatialvla: Exploring spatial representations for visual-language-action model. In *Robotics: Science and Systems*, 2025.

[36] Santhosh Kumar Ramakrishnan, Devendra Singh Chaplot, Ziad Al-Halah, Jitendra Malik, and Kristen Grauman. Poni: Potential functions for objectgoal navigation with interaction-free learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. doi: 10.1109/CVPR52688.2022.01832.

[37] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[38] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Program generation for situated robot task planning using large language models. *Autonomous Robots*, 47:999–1012, 2023. doi: 10.1007/s10514-023-10135-3.

[39] SAM 3D Team, Xingyu Chen, Fu-Jen Chu, Pierre Gleize, Kevin J Liang, Alexander Sax, Hao Tang, Weiyao Wang, Michelle Guo, Thibaut Hardin, Xiang Li, Aohan Lin, Jiawei Liu, Ziqi Ma, Anushka Sagar, Bowen Song, Xiaodong Wang, Jianing Yang, Bowen Zhang, Piotr Dollár, Georgia Gkioxari, Matt Feiszli, and Jitendra Malik. Sam 3d: 3dfy anything in images. *arXiv preprint arXiv:2511.16624*, 2025.

[40] Lawson L. S. Wong, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Manipulation-based active search for occluded objects. In *2013 IEEE International Conference on Robotics and Automation*, pages 2814–2819, 2013. doi: 10.1109/ICRA.2013.6630966.

[41] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. Daydreamer: World models for physical robot learning. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 2226–2240. PMLR, 2023. URL `https://proceedings.mlr.press/v205/wu23c.html`.

[42] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[43] Sherry Yang, Yilun Du, Kamyar Ghasemipour, Jonathan Tompson, Leslie Pack Kaelbling, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. In *International Conference on Learning Representations*, 2024.

[44] Gengyu Zhang, Hao Tang, and Yan Yan. Versatile navigation under partial observability via value-guided diffusion policy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[45] Christos Ziakas and Alessandra Russo. Vita: Zero-shot value functions via test-time adaptation of vision-language models. In *International Conference on Learning Representations*, 2026.

A Implementation Details

This appendix provides additional implementation details for SCOUT, complementing the method and experimental setup in the main paper. We describe the simulator, data generation, scene reconstruction module, relational state construction, world-model training, MCTS rollout evaluation, and real-robot deployment.

A.1 Simulator and Dataset Generation

Shelf environment. All simulated scenes are generated in ManiSkill2 with SAPIEN physics [18]. We use a procedurally instantiated cluttered shelf with approximate dimensions $0.36\,\mathrm{m} \times 0.90\,\mathrm{m} \times 0.40\,\mathrm{m}$ and a fixed shelf-frame convention: depth points from the shelf opening toward the back wall, the lateral axis spans left–right, and the vertical axis points upward. Candidate object placements are sampled from a 2×6 slot grid inside the shelf volume.

Object assets. We construct cluttered shelf scenes from ManiSkill2-YCB objects [7], ProcTHOR / AI2-THOR assets [15], and RoboCasa kitchen assets [32]. YCB provides everyday graspable objects such as cans, boxes, mugs, and bottles, while ProcTHOR and RoboCasa provide additional household and kitchen objects. Objects are sampled from shelf-scale categories including mugs, cans, boxes, bottles, and jars. For stable simulation, visual meshes are rendered from the original assets, while collision geometry is approximated by box collision shapes. We use 77 YCB objects, 97 ProcTHOR objects, and 16 RoboCasa objects.

Scene sampling. Each scene is sampled by choosing a difficulty level and populating the shelf with objects from the asset pool. Easy, medium, and hard scenes contain 4–8, 7–12, and 10–16 objects, respectively; the benchmark settings in the main paper use 8–16 objects. Each scene contains task objects, foreground blockers, and distractors. Distractors are objects that are not directly part of the task but may still affect planning if they block the access corridor. Single-object tasks contain one task object, while multi-object tasks contain two task objects. Each scene contains 2–5 foreground blockers, with the remaining objects treated as distractors. Candidate poses are sampled with jitter and rejected if they violate shelf bounds, intersect other objects, or fail the initial occlusion requirement.

Task families. As described in Sec. 3.1 and illustrated in Fig. 5, we evaluate four task families: **Reveal**, **Reveal_Multi**, **Place**, and **Place_Multi**. Reveal specifies one occluded target object, while Reveal_Multi specifies two targets. Place specifies one object and an axis-aligned goal region, while Place_Multi specifies two objects and a shared goal region. In the reported experiments, the goal region occupies the front 30% of the shelf depth, spans the valid lateral workspace, and uses the lower half of the shelf height as the placement volume. All tasks use a maximum episode budget of $T = 12$ primitive actions.

Table 3: Task construction details for the simulated shelf benchmark.

Task	Targets	Success condition
Reveal	1	Target visibility and accessibility exceed threshold
Reveal_Multi	2	Both targets satisfy Reveal predicate
Place	1	Target object lies inside assigned goal region
Place_Multi	2	Both target objects lie inside the shared goal region

Camera sampling. Each scene is observed from sparse front-facing RGB-D viewpoints, matching the partial-observation setting in Fig. 2. We generate 16 candidate views by varying camera yaw, pitch, lateral offset, height offset, and distance from the shelf opening, and select up to 5 views per episode. The RGB-D resolution is 896×504. Depth is represented in metres, clamped to $[0.05, 3.0]$, and back-projected using the camera intrinsics. RGB images are normalized with ImageNet mean and standard deviation. At this resolution, we use the approximate intrinsics

$$K = \begin{bmatrix} 640.0 & 0 & 448.0 \\ 0 & 640.0 & 252.0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Table 4: Camera sampling ranges used for simulated RGB-D observations.

Parameter	Values / Range
Candidate views	16
Selected views	up to 5
Distance from shelf opening	0.34–0.46 m
Yaw	$\{-20, -12, 0, 12, 20\}^\circ$
Pitch	$\{0, -8\}^\circ$
Lateral offset	$\{-0.05, 0.0, 0.05\}$ m
Height offset	$\{-0.08, 0.0, 0.08\}$ m
RGB-D resolution	896×504
Depth range	$[0.05, 3.0]$ m

Transition data for world-model training. For each simulated state, we store oracle object poses, object sizes, relational labels, RGB-D observations, segmentation masks, occupancy volumes, object-slot ground truth, and reconstructed slot states. To construct the transition dataset, we sample a valid cluttered scene, record the pre-action state, execute an object-centric primitive (i, u) in simulation, allow physics to settle, and record the post-action state. Transition labels are obtained by comparing the pre- and post-action object states. The displacement target is $\Delta\mu_i = \mu_i^{t+1} - \mu_i^t$, the moved-object mask marks objects whose centers move by more than 5×10^{-4} m, and the validity label indicates whether the transition remains geometrically feasible within the shelf bounds. Invalid transitions include no-motion actions, missing-object actions, severe collisions, and motions that move objects outside the valid workspace.

Primitive actions for transition generation are sampled by a heuristic back-to-front policy that prioritizes front-row blockers and actions likely to expose deeper objects, while still sampling lateral shifts and forward pulls to cover diverse local interactions. The reported training set contains 2000 simulated scenes and the validation set contains 500 simulated scenes. Each scene contributes up to 12 primitive transitions, yielding approximately 24,000 training transitions and 6,000 validation transitions before filtering invalid or duplicate transitions.

Evaluation splits. Each simulation setting in Fig. 6 is evaluated over 500 rollouts and 3 random seeds. The in-distribution split uses the same asset pool, camera ranges, and object-count distribution as training. The novel-object split holds out 20% of assets from each source and samples task objects from the held-out pool. The novel-view split uses yaw angles $\{\pm 28, \pm 35\}^\circ$, camera distances 0.48–0.56 m, and height offsets up to ± 0.12 m. For clutter-density evaluation, we use object counts $\{8, 10, 12, 14, 16\}$.

A.2 GHOST Scene Reconstruction

Input representation. GHOST receives RGB-D observations, camera intrinsics, camera extrinsics, and object-level priors. For each view, the input consists of an RGB image, a depth map in metres, an intrinsic matrix K, a world-from-camera transform T_{wc}, and object masks. In simulation, object masks are obtained from simulator segmentation when available; otherwise SAM3D is used as an object-level prior [39]. The planner uses up to 5 views from the candidate view set described above. Fig. 10 shows representative RGB-D observations and segmentation masks used as input to GHOST before cross-view fusion.

Object priors and cross-view fusion. For each RGB-D view, SAM3D provides an object mask, coarse 3D geometry from mask and depth, a confidence score, and a learned shape token. We use a confidence threshold of 0.5 and shape-token dimension 64. Per-view hypotheses are associated in 3D using coarse object boxes. Detections with 3D-IoU above 0.25 are merged into a single visible-slot initialization. Object centers and sizes are averaged using confidence weights, existence probability is set to the maximum confidence in the cluster, and shape tokens are aggregated by confidence-weighted averaging.

Visible and hidden slots. Visible slots are initialized from fused observed object hypotheses. Hidden slots are initialized as persistent learned queries and allow the model to represent unobserved objects that may affect planning. We use 16 hidden slots per scene. During training, hidden slots are

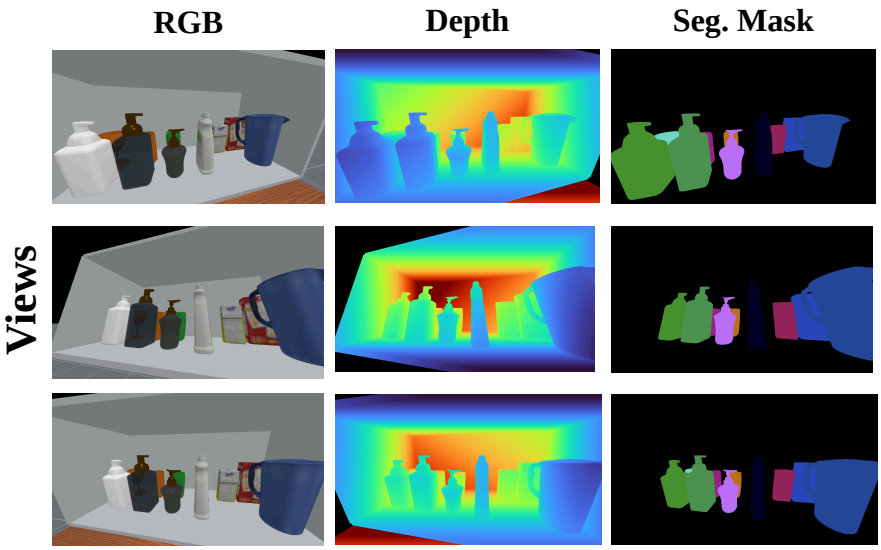

Figure 10: Example sparse RGB-D observations used by GHOST. Each row shows a different front-facing view of the same cluttered shelf scene, with corresponding RGB image, depth map, and object segmentation mask.

matched to unobserved ground-truth objects when available; unmatched hidden slots are supervised toward low existence probability or zero displacement during world-model fine-tuning.

Slot transformer. Visible and hidden slots are refined using a slot transformer with model dimension 128, 8 attention heads, 4 layers, feed-forward dimension 256, and dropout 0.1. Each layer performs self-attention over the combined slot set, followed by cross-attention to per-view shape and geometry tokens. The geometry-token dimension is 64, the maximum number of observed tokens is 2048, and learned positional embeddings are used for slot and view tokens.

Gaussian refinement and outputs. After slot refinement, each slot emits 8 local Gaussians with anchor positions tied to the slot center. Gaussian scales are constrained to $[0.01, 0.15]$ m, with maximum offset 0.08 m and maximum anchor drift 0.16 m. The Gaussian representation is rendered into a sparse occupancy volume at voxel size 0.015 m over the shelf bounds with 0.05 m padding. The final slot output is

$$s_i^t = (\mu_i^t, \sigma_i^t, p_i^t), \tag{3}$$

where $\mu_i^t \in \mathbb{R}^3$ is the center, $\sigma_i^t \in \mathbb{R}^3$ is the size, and $p_i^t \in [0, 1]$ is the existence probability. Thus, each slot contains 7 scalar values. The complete reconstructed state is $z_t = \{s_i^t\}_{i=1}^K$.

Relational graph. As shown in Fig. 3, GHOST produces a relational graph over object slots. Contact edges are added when object AABBs lie within 0.005 m. Support edges require vertical contact and non-intersecting AABBs. Blocker edges are computed using projected overlap between an object and the target or goal corridor. We use a 5% projected-overlap threshold for blocker edges.

A.3 Relational Planning State and Metrics

Planning state. The relational planning state is derived as

$$x_t = \mathcal{P}(z_t, g), \tag{4}$$

where z_t is the reconstructed slot state and g is the task goal. Each object entry contains its id, category, center, size, visibility fraction, visible/hidden state, and target flag. The state also contains blocker, contact, and in-front-of relations, along with free-space estimates in the left, right, and back directions. The per-slot planning feature vector has length 13 and includes normalized center, normalized size, visibility, target indicator, blocker indicator, directional free-space values, and a slot bias.

Blocker relation. A blocker relation from object i to target t is added if i lies in front of t along the shelf depth axis and overlaps the target or access corridor in the lateral-vertical cross-section. Specifically, object i is considered in front of t if

$$x_i + 10^{-3} < x_t, \tag{5}$$

and its projected YZ overlap exceeds 5%.

Visibility. The visibility score measures how much of the task site is visible from a fixed front-facing evaluation camera. For Reveal tasks, it is the visible fraction of the target object. During rollout evaluation, we approximate visibility by projecting objects in front of the target onto the target's YZ cross-section:

$$\mathrm{Vis}(t) = 1 - \mathrm{clip}\left(\frac{A_{\mathrm{blocked}}(t)}{A_{\mathrm{YZ}}(t)}, 0, 1\right). \tag{6}$$

For Place tasks, visibility is computed over the goal region and normalized by the pixel area of the same region under an empty-shelf rendering.

Accessibility. Accessibility measures whether the task site can be reached from the shelf opening. For Reveal tasks, it is computed from side room, back room, and blocker count:

$$\mathrm{Acc}(t) = \frac{0.5\min(\mathrm{side_room}/0.06, 1) + 0.5\min(\mathrm{back_room}/0.08, 1)}{1 + \mathrm{blocker_count}(t)}. \tag{7}$$

For Place tasks, accessibility is computed using a front-to-site corridor from the shelf opening to the goal-region center. The corridor has width 0.08 m, height 0.12 m, and depth equal to the distance from the shelf opening to the task site.

Success criteria. Reported success in Fig. 6 is computed from simulator ground-truth geometry, while planning uses the reconstructed state. Reveal succeeds if

$$\mathrm{Vis}(t) \geq 0.6, \quad \mathrm{Acc}(t) \geq 0.6, \quad \mathrm{blocker_count}(t) = 0. \tag{8}$$

Reveal_Multi requires this predicate for every target. Place succeeds if the target object center lies inside the assigned goal region $\mathcal{G}$:

$$\mu_t \in \mathcal{G}. \tag{9}$$

Place_Multi requires all target objects to satisfy this predicate. Actions are marked invalid when the predicted transition violates shelf bounds, object existence, primitive validity, or geometric feasibility constraints.

A.4 Action Space and LLM-Guided Action Proposals

Primitive actions. The action space contains four object-centric primitives:

$$a = (i, u), \quad u \in \{\texttt{shift_left}, \texttt{shift_right}, \texttt{shift_back}, \texttt{pull_forward}\}, \tag{10}$$

where i is the object index. The primitive directions are defined in the shelf frame.

Table 5: Discrete manipulation primitives used by SCOUT.

Primitive	Axis	Sign	Termination
`shift_left`	y	-1	wall contact
`shift_right`	y	$+1$	wall contact
`shift_back`	x	$+1$	wall contact
`pull_forward`	x	-1	front boundary

The maximum primitive magnitude is 0.08 m, the minimum motion threshold is 10^{-4} m, and the contact margin is 0.002 m. The planner does not command object rotations; rotations may arise passively from physics.

LLM input and output. The LLM receives the task goal g and the initial relational state x_0. The serialized input includes target id, target visibility/accessibility summary, object list, blocker relations, contact relations, in-front-of relations, free-space estimates, and valid primitive names. The LLM proposes compact object-direction candidates; it does not predict physical feasibility or choose the final plan. We use Qwen2.5-7B-Instruct as the open-source action-proposal model [42].

```
You are helping a robot plan in a cluttered shelf.
The robot can only use these object-centric primitives:
shift_left, shift_right, shift_back, pull_forward.

You are given a relational planning state with objects,
target information, blocker relations, visibility,
accessibility, and directional free-space estimates.

Your task is to propose compact action candidates for search.
Do not predict physics. Do not choose the final plan.
Only suggest which objects are worth considering and which
coarse direction is likely useful.

Return JSON only with this schema:
{
  "action_proposals": [
    {
      "object_ids": [int, ...],
      "direction": "shift_left|shift_right|shift_back|pull_forward",
      "reason": "short phrase"
    }
  ]
}

Rules:
- Prefer objects that block the target or the goal region.
- Prefer directions that open visibility or a front-facing access corridor.
- Include at most 3 action proposals.
- Each proposal may contain 1 to 4 object ids.
- Use only object ids that appear in the input.
- Use only the four allowed primitive names.

Input:
{TASK_AND_RELATIONAL_STATE_JSON}
```

Each proposal is represented as $c = (\mathcal{I}_c, d_c)$, where $\mathcal{I}_c$ is a subset of object ids and d_c is a coarse primitive direction. The expanded action set is

$$\mathcal{A}(c) = \{(i, d_c) \mid i \in \mathcal{I}_c\} \cup \mathcal{A}_{\text{aux}}. \tag{11}$$

Auxiliary actions are generated deterministically from geometry, including pulling the target-facing blocker forward and shifting the nearest corridor blocker laterally toward the side with more free space. If parsing fails, the system falls back to these deterministic proposals. The LLM is served locally with temperature 0.2, top-$p = 0.9$, and a maximum of 512 new tokens.

A.5 World Model Training

Inputs and targets. The learned world model receives object features, a padding mask, global action features, and a hallucinated-slot mask. Object features have dimension 13 and include normalized center, normalized size, existence probability, visibility fraction, target flag, acted-object flag, hallucinated-slot flag, back-row flag, and a slot bias. Global features have dimension 11 and include the acted-object summary, primitive one-hot encoding, and primitive direction vector.

The world model predicts per-object displacement, moved-object logits, and transition validity:

$$f_\phi(z_t, a_t) \to \{(\Delta\mu_i, m_i, v)\}_{i=1}^{K}, \tag{12}$$

where $\Delta\mu_i \in \mathbb{R}^3$ is the normalized displacement, m_i is a moved-object indicator, and v is a transition-validity flag. Object sizes and existence probabilities are passed through unchanged.

Architecture. The world model uses an object-token encoder, a global-feature encoder, self-attention interaction layers, and prediction heads. The object encoder is a two-layer MLP with GELU activations and hidden dimension 96. The global feature is encoded and broadcast to all object tokens. The interaction module contains 2 transformer-style blocks with 4 attention heads, LayerNorm, and feed-forward dimension 2×96. The trunk hidden dimension is 128. The displacement head outputs a 3D displacement per slot, the moved head outputs one logit per slot, and the valid head outputs one transition-validity logit from the pooled global feature.

Loss. The world-model loss is

$$\mathcal{L}_{\mathrm{WM}} = w_\Delta \mathcal{L}_\Delta + w_m \mathcal{L}_m + w_v \mathcal{L}_v + \mathbf{1}_{\{\mathrm{phase}=2\}} w_h \mathcal{L}_h. \tag{13}$$

Here, $\mathcal{L}_\Delta$ is a masked Smooth-L1 displacement loss, $\mathcal{L}_m$ is binary cross-entropy for moved-object prediction, $\mathcal{L}_v$ is binary cross-entropy for transition validity, and $\mathcal{L}_h$ penalizes nonzero motion on hallucinated slots. We use weights $(w_\Delta, w_m, w_v, w_h) = (1.0, 0.5, 0.25, 0.5)$.

Training schedule. World-model training uses the transition dataset described above. Phase 1 trains on oracle simulator states, and Phase 2 fine-tunes on GHOST-reconstructed states while keeping oracle pre/post states for supervision. Phase 1 trains for 8 epochs with batch size 32 and learning rate 3×10^{-4}. Phase 2 initializes from Phase 1 and trains for 4 epochs with batch size 32 and learning rate 10^{-4}. We use AdamW [31], weight decay 10^{-5}, gradient clipping at 5.0, and fp32 loss accumulation.

A.6 MCTS and Rollout Scoring

Search tree. At each decision, the MCTS module [6] in Fig. 4 builds a new search tree from the current root state z_0. Each node stores the predicted world state, relational planning state, parent pointer, visit count, accumulated value, prior, depth, rollout summary, and applied action. Root actions are generated from the LLM-induced candidate set. Deeper nodes use deterministic geometric proposals and do not call the LLM again.

Selection and expansion. Selection uses a PUCT-style rule:

$$a^* = \arg\max_a \left[Q(s,a) + c_{\mathrm{puct}} P(s,a) \frac{\sqrt{N(s)}}{1 + N(s,a)} \right], \tag{14}$$

with $c_{\mathrm{puct}} = 1.25$. We use 32 MCTS iterations, maximum depth 2, branch factor 6, rollout horizon 2, and discount 0.97. The full episode budget is $T = 12$ actions.

Rollout score. Each rollout τ is summarized by visibility gain, accessibility gain, blocker change, and an invalid-transition flag:

$$r(\tau) = (\Delta\mathrm{vis}(\tau), \Delta\mathrm{acc}(\tau), \Delta\mathrm{block}(\tau), I_{\mathrm{invalid}}(\tau)). \tag{15}$$

The rollout score is

$$V(\tau) = \Delta\mathrm{vis} + \Delta\mathrm{acc} - 0.25\, B_{\mathrm{end}} - 0.10 \max(0, \Delta\mathrm{block}) - 1.0\, I_{\mathrm{invalid}}, \tag{16}$$

where B_{end} is the final blocker count. These terms correspond to the diagnostics reported in Table 1. On one NVIDIA L40S GPU, the median LLM proposal latency is 0.42 s, the mean MCTS wall time per planning step is 0.68 s, and the mean world-model forward time is 4.8 ms.

A.7 Simulation Evaluation

Each simulation setting is evaluated over 500 independent rollouts and 3 random seeds. Results in Fig. 6 are reported as means over seeds, with error bars denoting standard error. Random seeds affect scene sampling, object placement, camera subset selection, and stochastic LLM decoding. The default evaluation distribution uses 8–16 objects, and multi-target tasks use two target objects.

Closed-loop sensing. The initial-view protocol is a controlled sparse-observation stress test. For deployment, closed-loop sensing refreshes the scene reconstruction and replans after each primitive. Table 6 compares the two protocols in simulation using the same primitive executor, $T = 12$ budget, and success criteria. Re-observing every two primitives gives $84.8 \pm 3.1\%$ success and 4.6% hidden-scene failures. Re-observing after every primitive reduces hidden-scene failures from 7.6% to 3.3% of all episodes, while episode time increases from 10.4 to 24.2 seconds.

Table 6: Simulation success (%) with initial observations only and re-observation after every primitive.

Method	Initial views only	Re-observation
$\pi_{0.5}$	55.5	63.2 ± 4.1
Long-VLA	52.3	64.0 ± 4.0
ParticleFormer	72.8	82.3 ± 3.9
SCOUT	80.8 ± 3.5	86.3 ± 3.0

Proposal and search ablations. With the same world-model and MCTS budget, geometry-only proposals achieve $74.6 \pm 3.9\%$ success, compared with $80.8 \pm 3.5\%$ for LLM plus geometry. Local search depths 1, 2, 3, and 4 achieve $70.6 \pm 4.2\%$, $80.8 \pm 3.5\%$, $81.5 \pm 3.4\%$, and $81.1 \pm 3.6\%$, respectively, with relative world-model compute of $1.0\times$, $1.8\times$, $2.6\times$, and $3.4\times$.

Predicted versus executed success. Across 500 evaluation episodes, predicted-state and ground-truth success agree in 89.2% of cases; predicted-success precision is 91.7%, and false positives account for 7.0% of all episodes. Visibility and accessibility MAE are 0.078 and 0.094, respectively. Replacing only GHOST's initial state with oracle geometry raises success from $80.8 \pm 3.5\%$ to $87.8 \pm 2.8\%$.

Reconstruction quality. On the held-out reconstruction set using Hungarian matching, multi-view GHOST achieves 72.6% hidden-object recall and 0.81 blocker F1, compared with 58.9% and 0.71 for single-view GHOST. These diagnostics measure reconstruction separately from downstream task success.

A.8 Real-World Hardware and Protocol



Figure 11: Real-world shelf observation setup. Left: Spot observes the cluttered shelf using its wrist-mounted camera. Right: representative robot-captured RGB views and corresponding depth maps used for scene reconstruction.

Robot platform. Real-world experiments in Fig. 9 are conducted on a Boston Dynamics Spot robot equipped with the Spot Arm [5]. The Spot Arm is mounted in the standard configuration and is controlled through the Spot SDK. We use the standard Spot Arm gripper, which provides actuated gripper jaws and integrated sensing rather than a custom parallel-jaw end effector [5].

Observation. The real system uses the wrist-mounted gripper sensors. The Spot Arm gripper includes RGB and ToF depth sensing, which we use for frontal shelf observation [5]. The RGB stream is captured at 3840×2160 resolution and resized for perception. The ToF depth stream is aligned to the RGB frame and filtered to the shelf workspace. We capture $3 - 5$ frontal wrist

observation at the start of each planning episode. Fig. 11 shows the real shelf observation setup and representative robot-captured RGB-D views used for scene reconstruction.

Shelf setup. We construct real cluttered shelf environments using household objects. The real shelf has approximate dimensions 0.36 m depth, 0.90 m width, and 0.40 m height, matching the simulated shelf scale. Each trial contains 6–10 household objects, including bottles, cans, boxes, mugs, and containers.

Task setup. We evaluate Reveal and Place tasks. Reveal requires exposing and accessing a partially occluded target, while Place requires moving a specified object into a marked goal region after clearing nearby clutter. We run 10 trials per task, for 20 total trials. SCOUT succeeds in $12/20$ trials, corresponding to 60% overall success, with $7/10$ Reveal successes and $5/10$ Place successes.

Primitive execution and success. Each primitive is executed as a Cartesian end-effector motion in the shelf frame. Push primitives use a straight-line motion of 0.08 m at approximately 0.05 m/s. Pull primitives establish contact near the front face of the selected object and retract by 0.08 m along the negative depth direction. After each primitive, the end effector retracts upward by 0.04 m. The real-world action budget is $T = 12$, matching simulation. Reveal succeeds if the target is exposed and reachable from the shelf opening; Place succeeds if the specified object is moved into the marked goal region. A trial is marked unsuccessful if the robot exhausts the action budget, executes an invalid primitive, or leaves the target or goal region blocked.

Real-world failures. Among the 8 unsuccessful real-world trials, 3 failures are due to execution errors such as slipping or insufficient object displacement, 3 are due to perception or reconstruction errors, 2 are due to transition-prediction mismatch.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims match the proposed method and are supported by simulation, ablations, failure analysis, and real-robot experiments.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations section discusses reconstruction errors, world-model errors, static-scene assumptions, and the fixed primitive set.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: The paper does not present theoretical results or formal proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper and appendix describe the benchmark, tasks, metrics, baselines, training setup, and real-robot protocol.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

Justification: Due to time limits, the supplemental material includes data-generation code and instructions; we plan to release the full source code for training and evaluation.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper reports task setup, data splits, metrics, optimizer, and evaluation protocol, with full hyperparameters in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Simulation results report uncertainty over evaluation seeds, and real-robot results are reported as success counts over fixed trials.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Compute resources, training hardware, and inference setup are reported in the implementation details and appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses simulated data, public assets, and controlled robot experiments, with no human subjects or private data.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses benefits for reliable cluttered-space manipulation and risks from perception or planning failures in physical deployment.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a high-risk model, scraped dataset, or human-subject dataset.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Existing simulators, object assets, baselines, and open-source models are cited, with license details provided where applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The generated shelf benchmark and transition data are documented in the appendix and supplemental data-generation instructions.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve human subjects or user studies.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: The paper describes the LLM's role as an action-proposal model, including its inputs, outputs, and separation from world-model evaluation and MCTS.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.