VLM-Focus: Task-Relevant Scene Reduction for Planning and Control in Clutter

Aileen Liao, Rachel Holladay, Dinesh Jayaraman, and Michael Posa
University of Pennsylvania
Email: {aliao, rhollada, dineshj, posa}@seas.upenn.edu

***Abstract*—Despite recent advances in general-purpose robotic manipulation, real-world multi-object clutter remains challenging to handle for today's prevalent approaches. The problem scales in complexity due to more objects and collisions, harder dynamics, distractors, and task ambiguity. Bridging this gap to real-world deployment requires effective scene abstractions, yet current methods rely heavily on manually engineered representations, an approach that does not scale. As a result, these abstractions are costly to generate and difficult to adjust or fine-tune. We instead propose to automatically generate scene-specific, task-specific, adaptive abstractions, without manual intervention. VLM-Focus produces a de-cluttered abstracted scene representation by merging (eg., stacked objects) or pruning (eg., distant objects) scene entities in a closed loop in response to task progress. These resulting dynamic task-relevant abstractions are versatile and easy to use. In our experiments, we show that VLM-Focus improves classical planning, model-based control, and a vision-language-action model, across a diverse set of highly cluttered manipulation scenes.**

I. Introduction

A messy home is a happy home, but a robot's nightmare. As scene clutter grows in unstructured real-world environments, the number of objects, contacts, decision branches, and interaction modes become limiting factors for solvability and real-time performance in planning and model-based control. Learning offers no silver bullet here: learned policies also face increasing difficulty in cluttered scenes due to semantic and spatial ambiguity, as well as distribution shifts.

Due to these challenges in scaling scene complexity, research in robotic planning and control often focuses on scenes constructed specifically for demonstration: curated environments with relatively few objects, almost all of which are directly relevant to the task. These scenes bear only a passing resemblance to real-world environments, which are typically dense, unstructured, and filled with irrelevant distractors.

To handle information-dense scenes in complex settings, modern planners and controllers often operate with fixed, manually crafted abstractions to simplify the scene.

For example, most task and motion planning (TAMP) frameworks use a fixed, object-centric world representation: objects, their relations, and possible abstractions are predefined, usually by the user [34, 7, 12, 11]. This makes it difficult to handle variable clutter, where only a subset of objects are relevant to the task. As illustrated in Fig 1, classical planners must reason over all objects, including irrelevant distractors (grey), and small changes in the scene or goal often require re-engineering the domain.

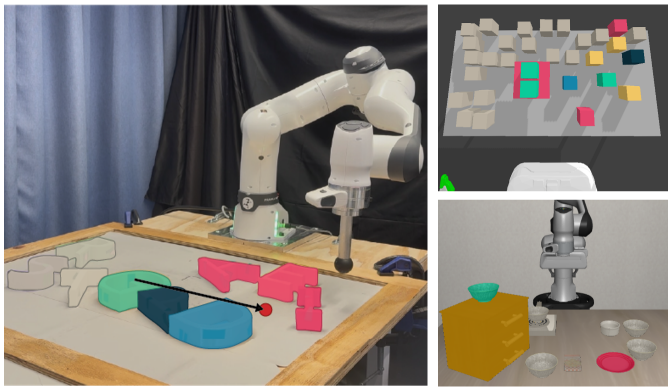

Fig. 1. Scene complexity is reduced via pruning and merging. (Left) The control task is to move the green "G" (green) to a target location. VLM-Focus identifies that the "S", "T", and "R" (gray) can be ignored; the "A", "E", and "I" (pink) are relevant, but can be thought of as a single object (merged); the "P" (dark blue) and "D" (light blue) are relevant and must be modeled. (Top right) for a TAMP problem, the robot must pick and move the three green cubes. VLM-Focus identifies nearby objects as relevant, but prunes the vast majority (gray). (Bottom right) Using a VLA, the green bowl must be moved to the red plate. Other bowls clutter the scene, but VLM-Focus identifies that they can be pruned from the observation before calling the VLA.

Similarly, model-based control methods for manipulation must tackle the fundamental challenge of needing to reason over a combinatorial number of possible contact mode sequences [6, 35, 18, 20]. While recent progress in contact-implicit control has shown the ability to manipulate a few objects at a time [5], these frameworks still struggle as the number of objects in the scene grows. As scene complexity increases to real-world scales, explicitly representing every potential contact modes and conVLM-Focus-C3straints becomes computationally prohibitive.

Learned controllers, including vision-language action (VLA) models, face a related problem [30, 8, 41]: they are typically trained on scenes with few objects and limited clutter. When deployed in dense, unstructured environments, these models encounter semantic and geometric distribution shifts—novel object arrangements, occlusions, or functional couplings not seen during training—leading to large prediction errors, incorrect affordance estimates, or unrealistic contact interactions. Without mechanisms to focus on task-relevant objects, both optimization-based and learned controllers degrade rapidly in performance as scene complexity increases.

To address this, we introduce VLM-Focus, a general, scalable, task-agnostic method for determining and updating task relevancy throughout task execution using vision language

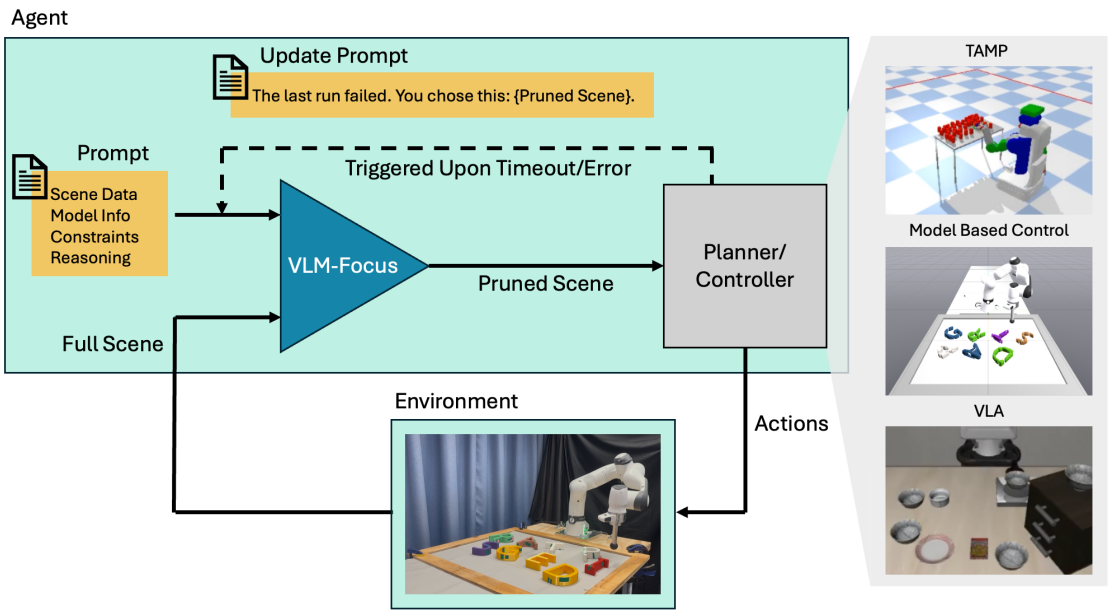

Fig. 2. System overview of VLM-Focus. Given a task description via the full scene and prompt information, VLM-Focus (blue) prunes and merges objects to produce an abstracted scene, which is passed to the planner or controller (grey). When triggered, VLM-Focus revises its scene pruning. This approach generalizes to TAMP, model-based controllers, and VLAs.

models (VLMs) with task-success feedback. We consider various kinds of task relevance: semantic (what is contextually relevant to the goal?), geometric (what blocks a desired motion?), and dynamic (what will move or be affected?). VLM-Focus captures object affordances, functional roles, and semantic importance. We show how this automatically-assigned task relevance can be used in versatile ways across multiple classes of planners and controllers, shaping their state and action space to enable them to function in clutter.

We iteratively refine a scene representation using a VLM, provided with task feedback, to identify task-relevant objects. Refinement occurs via two elementary operations shown in Figure 1: (i) *pruning*, which removes objects that can be ignored by the planner or controller, and (ii) *merging*, which groups objects that are functionally or dynamically coupled into a single composite entity. The resulting reduced scene representation preserves physical grounding while significantly reducing the combinatorial complexity faced by downstream planners and controllers.

We recognize that VLMs are not infallible oracles; zero-shot predictions can suffer from hallucinations or physical reasoning errors, or the environment state may evolve to require a different scene representation. Therefore, we do not treat the initial abstraction as static. Instead, we implement closed-loop re-prompting, triggered by failure. If a downstream planner finds the abstracted scene infeasible, or a controller fails during simulation or execution, that feedback is passed back to the VLM. The model then iteratively corrects the abstraction, restoring over-pruned objects or refining merges.

The algorithmic implementation of this framework is intentionally lightweight, consisting primarily of prompt design and modular integration. The core innovation lies in the observation that modern VLMs offer a path to bridge the gap between control and planning algorithms which function in the highly structured laboratory, but struggle to scale or generalize, and the unbounded complexity of the real world. We shift the burden of abstraction from bespoke human intuition to a generalizable, automated semantic reasoning loop. By coupling this abstraction with an error-correction mechanism, we demonstrate that a simple, task-agnostic system can replace the hand-crafted heuristics typically required for complex contact-rich tasks.

We evaluate our approach in a series of contact-rich manipulation problems, demonstrating that VLM-based scene focusing enables our system to scale to significantly more cluttered environments than prior approaches, while maintaining or improving success rates and substantially reducing computation time. Moreover, we find that merging functionally related objects allows the system to reason more effectively about collective motion and contact interactions that would otherwise be intractable at the object level.

Our contributions are:

- A general, task-agnostic method for estimating task relevance using vision-language models and constructing task-focused scene abstractions. Closed-loop reprompting with feedback from planning or simulation corrects VLM errors and refines scene abstractions.
- An integration of this abstraction layer with three distinct state-of-the-art paradigms: (i) Task and Motion Planning (TAMP) [10], (ii) optimization-based contact-implicit control (C3+) [2, 5], and (iii) the π_0 large-scale Vision-Language-Action (VLA) model [15].
- An experimental evaluation in cluttered tabletop settings demonstrating improved scalability, robustness, and efficiency with increasing object count.

II. Related Work

VLMs have become a versatile tool for enabling robots to reason about and interact with their environment. While many systems combine multiple ideas, most approaches align with one of three recurring patterns: using VLMs to (i) generate or sequence actions and plans, (ii) guide or constrain planning

and search, or (iii) shape intermediate scene representations and abstractions. Our work falls into the third category, but differs from prior approaches by using VLMs to actively reduce the world model before any planning or control is performed.

A. VLMs for Action and Plan Generation

A large body of recent work [13, 39, 9, 22] uses language or vision-language models as high-level planners, program generators, or skill sequencers. Systems such as SayCan [1] and Code-as-Policies [21] leverage large language models to decompose tasks into subgoals, sequence skills, or generate executable programs for long-horizon manipulation. Other works use VLMs to directly output symbolic plans and continuous action parameters (e.g., Wang et al. [36]). As these and similar methods typically operate on a given world representation, they are complementary with our presented approach which focuses on producing that task-relevant representation.

B. VLMs for Guiding and Constraining Planning

A related line of work uses VLMs or LLMs to guide classical planners or search procedures. This includes generating task subgoals [40] (e.g., VLM-TAMP uses a VLM to produce semantically meaningful subgoals that guide a task and motion planner), predicting constraints that prune infeasible task plans (e.g., leveraging VLM reasoning to identify feasible task constraints a priori and reduce search effort)[17, 19], ranking or filtering candidate actions, or serving as critics that score plans [14]. For example, VIZ-COAST uses the common-sense spatial reasoning of a VLM to infer constraints before planning begins, reducing downward refinement failures and planning effort [38]. In these systems, the underlying planner itself remains unchanged, but the language model biases or constrains the search process. In contrast with these methods, which are designed to directly interact with specific planning algorithms, we operate at a more abstract level to adapt the scene itself, given only loose guidance regarding the specific planner.

C. VLM-Generated Scene Representations and Abstractions

Finally, most relevant to our approach, several works use vision-language models or learned perception systems to build richer semantic representations of scenes, such as object-centric scene graphs, relational graphs, or affordance-based abstractions. In task and motion planning, scene graphs have been used to encode spatial and relational structures [28], and decomposition or receding-horizon methods have been proposed to reduce search burden [43, 3]. More recent neuro-symbolic systems, such as Zhu et al. [44], represent the entire scene as geometric and symbolic graphs and use learned models to predict both high-level plans and low-level motions, effectively amortizing planning over a fixed, task-complete world representation.

These approaches improve semantic grounding and structured reasoning, but they typically maintain a complete representation of the scene. Even when relations or affordances are predicted, or object importance is learned to bias planning [32], all objects remain present in the model, and the planner or controller must still contend with the full combinatorial complexity.

A complementary line of work in model-based control addresses complexity through explicit contact-implicit model reduction [4], shape approximation for improved contact tractability [27], or dynamic-resolution and lumped-object models for object piles [37, 29]. These approaches, while promising, are ultimately tied to specific tasks or methods.

Learning-based manipulation systems such as CLIPort [31] use language-conditioned perception to act in cluttered scenes, but likewise operate over the full scene and do not explicitly reduce the underlying world model. Closer to our approach, ARRO [26] and PEEK [42] filter visual observations to improve policy robustness, but remain specific to visuomotor policies, require manually specified labels or fine-tuned models, and do not extend to TAMP or model-based control.

III. Background

As we will demonstrate the broad utility of VLM-Focus, in combination with three distinct planning and control strategies, we provide brief background for each strategy of interest here.

A. Task and Motion Planning (TAMP)

Task and Motion Planning formulates manipulation as a hybrid discrete–continuous problem, jointly solving for the sequence of parameterized actions and the parameters of those actions, which are governed by geometric and physical constraints [11]. Many TAMP frameworks operate on a domain description including: (i) a state space, often defined through a series of lifted predicates [25], (ii) a set of parameterized actions, often characterized by preconditions and effects, and (iii) a set of hybrid constraint satisfaction algorithms that solve for the action parameters, e.g. motion planners, grasp generators, collision checkers, etc.

The primary strength of TAMP lies in its explicit structure: symbolic reasoning enables long-horizon planning with correctness guarantees, while accounting for geometric and physical constraints. However, this structure also imposes strong modeling assumptions. Usually, all objects in the environment are explicitly represented in the state space regardless of their relevance to the current task. Search complexity grows combinatorially with the number of objects and constraint checking, such as collision-checking, is further burden by the need to consider the full scene.

To mitigate this, practical systems could craft domain-specific relevance rules. However, these mechanisms effectively encode a priori assumptions about which objects matter for a given task. While effective in narrow domains, these heuristics are brittle and do not generalize when task goals, object layouts, or scene semantics change. As a result, the performance of TAMP frameworks usually degrades rapidly in unstructured, cluttered environments where relevance cannot be predefined.

B. Contact-Implicit Model Predictive Control

Contact-implicit trajectory optimization and MPC formulate manipulation as a continuous optimization problem that jointly reasons over robot motion, object dynamics, and contact interactions [2]. The advantage of this formulation is expressiveness: controllers can reason about making and breaking contact, pushing multiple objects, and exploiting incidental interactions. However, this comes at a steep computational cost. Each object introduces new decision variables, collision constraints, complementarity conditions, and potential contact pairs. The resulting optimization problems are nonconvex and scale poorly with the number of objects.

In cluttered scenes, this leads to a fundamental trade-off. Controllers may simplify contact modeling (e.g., by limiting contact pairs [4] or softening constraints [18]), which reduces computational burden but sacrifices physical fidelity. Alternatively, they may attempt to model all interactions explicitly, often resulting in infeasible or numerically unstable optimization problems.

C. Vision-Language-Action Models

Vision-Language-Action models combine high-capacity visual encoders and language understanding modules to directly predict actions from perceptual observations and task descriptions [15]. These models excel at capturing semantic relationships, object affordances, and task intent, and can generalize to novel objects and instructions without explicit symbolic modeling.

However, VLAs typically operate over dense, unstructured perceptual representations. Scene understanding is implicit and distributed across learned feature spaces, rather than represented as an explicit object-centric world model. As a consequence, VLAs lack explicit mechanisms for enforcing physical constraints, reasoning about long-horizon interactions, or guaranteeing task feasibility. In cluttered environments, irrelevant objects, occlusions, and novel contact configurations introduce significant distribution shifts, often leading to incorrect affordance predictions or physically implausible actions.

IV. Automatically Generating Task-Relevant Reduced Scene Descriptions

Our approach constructs task-focused scene abstractions using a vision-language model (VLM), and iteratively refines it during task execution by monitoring task success feedback. We first describe the core VLM-Focus framework in Sec IV-A, before discussing its instantiation in various planner and controller architectures in the subsequent sections.

A. VLM-Focus Framework

We now describe the shared core features of VLM-Focus that can be adapted to many downstream architectures. See Figure 2 for a schematic, and Algorithm 1 for pseudocode. Given a natural language task description τ and a scene description $\mathcal{O}$ as input, the VLM (in our implementation, GPT-4o) is prompted to output information to help construct a structured scene abstraction $\tilde{\mathcal{O}}$. The scene descriptions O and $\tilde{\mathcal{O}}$ take a form specific to the particular planning or control framework. For example, in TAMP, $\mathcal{O}$ might consist of object-level features and predicates (e.g., mass, pose, and identity). For a VLA, it might instead be object segmentation masks overlaid on an image observation.

While the full VLM prompt is in the Appendix A, we highlight the salient points here. The VLM must provide two lists to the downstream systems: (i) a minimal set of task-**`Relevant`** objects that are critical for task completion, and therefore important to consider in the downstream planner or controller, and (ii) groups of objects that may be **`Merged`** and treated as a single entity without hurting task performance due to functional or spatial coupling.

Given these lists, we generate the abstracted scene description $\tilde{\mathcal{O}}$ by implementing "pruning" and "merging" operations on the original scene $\mathcal{O}$, as follows.

- Objects excluded in `Relevant` are omitted ("pruned"). The definition of pruning differs for each system and can range from a state space reduction in TAMP to a geometric/collision pairs reduction in C3+. At a high level, this eliminates unnecessary decision variables for TAMP, contact constraints for MPC, and visual distractors for VLA models.
- For each set of objects in `Merged`, VLM-Focus replaces those rigid bodies with a single abstract rigid body whose collision/proximity model is built by aggregating the members' geometry into one co-moving description that is recomputed from the members' poses at run time. This reduces the number of bodies and contact pairs while retaining a conservative (or approximately tight) envelope of the group's physical extent, depending on the geometry representation used by the downstream simulator/controller. Any selected groups in `Merged` are also included in `Relevant`.

Figure 1 illustrates how pruning and merging transform a cluttered scene into a compact, task-focused abstraction.

An important benefit of *autonomously* constructing the reduced scene representation $\tilde{\mathcal{O}}$ is that it can also be autonomously revised when necessary. In VLM-Focus, feedback is triggered by time-outs or error codes (eg. workspace limits, runtime errors) in downstream systems. Then, a framework-specific feedback prompt along with the previous $\tilde{\mathcal{O}}$ is appended to the existing prompt and current scene, and VLM-Focus re-queries the VLM for a revised scene reduction (See Fig. 2).

In this manner, VLM-Focus injects high-level task semantics into a dynamic scene-abstraction process without needing manually specified task-dependent heuristics.To illustrate its versatility, we now discuss how VLM-Focus-generated scene representations $\tilde{\mathcal{O}}$ are integrated into various representative downstream planning and control frameworks.

B. VLM-Focus For Task and Motion Planning (TAMP)

We represent the environment $\mathcal{O}$ as a scene graph of objects with known poses, masses, relations (such as stacked, on) and properties (such as graspable). $\tilde{\mathcal{O}}$ is the subset of $\mathcal{O}$ whose poses are treated as decision variables in the planning problem

Algorithm 1 The VLM-Focus Framework

Require: Task τ, scene objects $\mathcal{O}$, system $\mathcal{S}$, max retries N
Ensure: Task execution using abstracted scene $\tilde{\mathcal{O}}$

for $i = 1$ to N **do**
  Generate abstracted scene $\tilde{\mathcal{O}}$ via VLM
    Construct VLM prompt from τ, object metadata $\mathcal{O}$, and system information $\mathcal{S}$
    Query VLM and parse output into Relevant and Mergings lists
    Prune objects not in Relevant
    Merge object groups from Mergings
    Execute controller/planner using abstract scene $\tilde{\mathcal{O}}$
  if execution succeeds **then**
    return success
  else
    Append execution failure feedback to prompt
  end if
end for
return failure

if PDDLStream [10], allowing actions that explicitly modify their pose (e.g., pick, place, or push). In contrast, static objects excluded from $\mathcal{O}$ have fixed poses and cannot be acted upon, though they still impact feasibility.

To determine $\tilde{\mathcal{O}}$, VLM-Focus is provided with a list of object IDs and their associated mass and pose, along with a goal specification represented as a list of tuples (o_i, g_i), where each tuple corresponds to an object and its desired goal pose.

Groups in `Merged` must be stably movable together (eg. stacks). They adopt a new, unified ID, and the parent object is the lowest supporting object in the stack. Object groups in `Merged` have their geometries fused and maintain the mass of the base object. A set of merged objects cannot be moved separately. Objects which have been pruned or merged are no longer considered for manipulation, which reduces the search space and results in a strictly smaller hybrid planning problem. To maintain feasibility constraints, collision checking over the full scene $\tilde{\mathcal{O}}$ is still required.

In VLM-Focus, feedback is triggered when no solution has been found or when the maximum timeout of 120 seconds has been reached. The feedback includes the previous scene representation and a description of the failure in the form of: "The last run failed. You chose [previous representation]. Choose a larger set of objects from the scene that is not just goal objects." This feedback encourages larger selection sets to expand the plan search. The feedback is appended to the original prompt and new scene representation.

C. VLM-Focus for Model-Based Control

We represent the scene as a set of objects $\mathcal{O}$. Given the set of object's poses and names, VLM-Focus outputs a reduced set of objects via the `Relevant` and `Merged`. This generates $\tilde{\mathcal{O}}$, the reduced set of objects.

Note that sets in `Merged` are not required to be stably movable and are instead clustered if their contact sampling can be reasoned about together. For example, piles of objects are not physically stable but can be pushed aside together. The collective shape of the sets in `Merged` is updated in realtime as the scene changes. Since the controller treats the pile as a single object, it does not model or predict how its shape will shift over time.

We integrate this task-focused scene abstraction $\tilde{\mathcal{O}}$ into the C3+ contact-implicit controller [5] that models manipulation as a linear complementarity system (LCS) and solves a linear complementarity problem (LCP) at each control step.

Following pruning by VLM-Focus, each object in `Relevant` or set in `Merged` contributes a rigid body to the plant, potential contact pairs, and sampled surface regions. Removing objects reduces (i) the number of bodies, (ii) the number of candidate contacts, and (iii) the dimensionality of the LCP. This also reduces the sampling space, since surface samples are generated only for retained objects. Both contact generation and contact resolution therefore operate on a reduced-order system.

Feedback is triggered when 250 controller loop iterations was reached or an error code was received. The feedback includes the previous scene representation and a description of the failure in the form of: "Your previous answer was not solvable. You selected [past object selection]." The feedback is appended to the original prompt and new scene representation.

D. VLM-Focus For Vision Language Action Model

We integrate task-focused scene abstraction into a vision-language-action (VLA) policy by filtering visual inputs using the VLM output. Before querying the VLA (π-0.5 trained on the LIBERO-Spatial dataset [23]), the VLM is provided with $\mathcal{O}$, an overhead scene image and ground-truth object bounding boxes from the simulator, a semantic goal sentence, and the original prompt. Based on the task description, the VLM identifies `Relevant`, $\tilde{\mathcal{O}}$. `Merged` is not used.

To construct an input for the VLA using $\tilde{\mathcal{O}}$, a new image is created with only the objects in $\tilde{\mathcal{O}}$ included. In simulation, this is implemented via transparency in the renderer, yielding less cluttered, task-focused observations.[1] The resulting filtered images are then used as input to the VLA policy. This preprocessing step reduces visual clutter and semantic ambiguity by suppressing distractor objects unrelated to task execution. As a result, the VLA receives observations that are more closely aligned with the task-relevant scene structure encountered during training, improving robustness in cluttered environments. Beyond the training distribution, this filtering reduces the number of visually plausible affordances in the scene, simplifying action selection under clutter. Given the short task horizon, the filtered scene remains valid throughout execution, so no additional VLM re-queries are needed.

V. Experiments and Results

We evaluate the effectiveness of VLM-Focus across a set of tabletop manipulation tasks. Our goal is to assess whether

[1]For real scenes, these distraction-free visual observations could be generated through generative modeling-based image editing/inpainting tools.

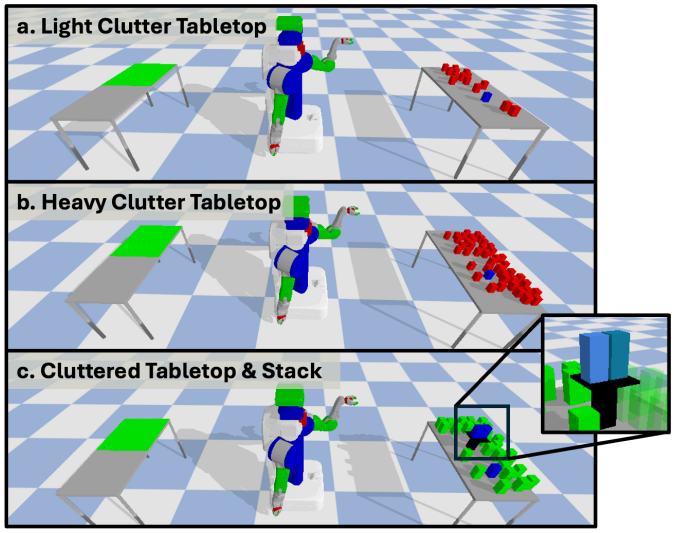

Fig. 3. VLM-Focus was evaluated on three TAMP environments: (a) a lightly cluttered tabletop, (b) a heavily cluttered tabletop, and (c) a cluttered tabletop with a stack. The stack consists of a tray (black) with two goal objects (blue). In each environment, a goal object is placed on the tabletop.

VLM-guided pruning and merging of scene representations improves task success and planning efficiency, especially in cluttered and semantically complex environments. Performance is measured using task success rate and planning/runtime efficiency, and we compare our method against baseline policies that operate on unfiltered visual observations. VLM query time averaged 1.76s over 30 trials in the model-based control setting, with comparable times for TAMP and VLA. This is negligible relative to controller execution time and was excluded from runtime comparisons.

A. Task and Motion Planning

To evaluate VLM-Focus's effect of a TAMP framework's performance, we construct three environments with increasing levels of visual and semantic complexity: (1) light-clutter tabletop scenes, (2) heavy-clutter tabletop scenes, and (3)

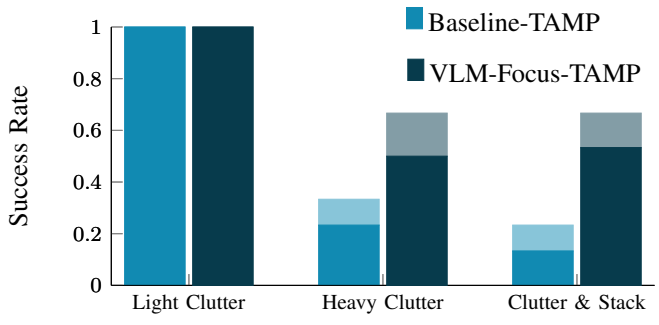

Fig. 4. Baseline-TAMP and VLM-Focus-TAMP success rates in three environments arranged left to right in order of complexity: Light Clutter, Heavy Clutter, and Clutter&Stack. VLM-Focus performs on par with the baseline for simple Light Clutter scenes, but its success rate is higher in the two more complex tasks. In each bar in the top row, Iteration 1 and 2 success rates are dark and light respectively. Iteration timeouts are 120s and baseline is rerun twice for fairness.

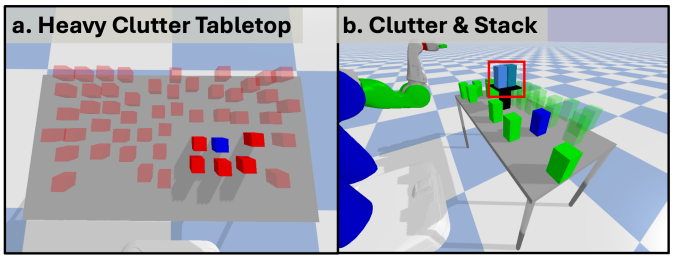

Fig. 5. Examples of scene pruning are shown for (a) Heavy Clutter and (b) Clutter & Stack. Goal objects are shown in shades of blue. Semi-transparent objects are pruned from planning, and the stack is merged and treated as a single composite object.

clutter & stack scenarios. The Light Clutter Tabletop scene (Figure 3a) includes 14 distractor blocks placed near the goal blue block. The Heavy Clutter Tabletop (Figure 3b) includes additional red blocks placed behind the goal object while space allows. Clutter & Stack includes three target objects shown in Fig 3c. Two are placed on a tray, and one is placed directly on a table surrounded by clutter. 29 green distractor objects are randomly populated while space allows.

Across all environments, we compare a policy with VLM-Focus against a baseline operating on unfiltered visual observations. Performance is measured using task success rate over 30 evaluation episodes, as shown in Fig 4. Two iterations of feedback are allowed. Success rates for each iteration are shown stacked, with the lighter region representing 2nd feedback loop.

- **Light Clutter**. In light-clutter, both methods achieve comparable success rates. This is because explicit scene abstraction provides limited benefit at low complexity. So, the best we can do is not *degrade* performance.
- **Heavy Clutter**. In heavily cluttered scenes, VLM-Focus-TAMP significantly outperforms the baseline. Baseline-TAMP frequently times out when searching for a feasible plan due to enumerating a large number of distractor objects. In contrast, our VLM-guided filtering suppresses irrelevant objects, resulting in higher task success rates. Fig 5 shows examples of our VLM-Focus-generated pruned representations. These results demonstrate that targeted scene abstraction improves robustness under high visual and semantic complexity by reducing spurious affordances and perceptual ambiguity.

  An anecdotal example (Figure 6) showcases an iteration of VLM-Focus pruning compared to a distance based metric such as only planning over the closest 4 target objects. While VLM-Focus may not be perfect and my have spurious selections, it is able to maintain the critical objects when multiple blocking objects need to be removed for access.
- **Clutter & Stack**. In this environment, our approach again achieves higher success rates than the baseline. This setting requires reasoning over both spatial relationships and object groupings while ignoring unrelated items. Our filtering mechanism simplifies the scene structure presented to the TAMP algorithm, enabling more reliable action selection in

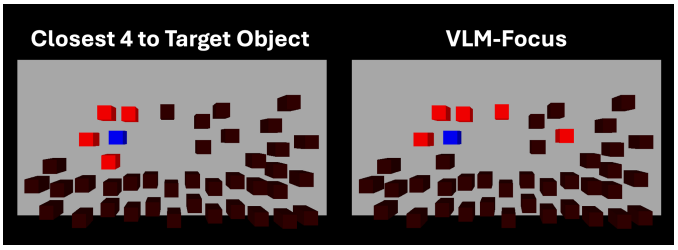

Fig. 6. Example TAMP scene where a distance-based baseline fails. The set of nearest objects to the goal (highlighted) fails to include a critical blocking object further away, which must be removed to enable a successful approach to the target. VLM-Focus identifies the blocker as task-relevant.

the presence of competing visual cues. Fig 5 provides an example of our pruned representation. The baseline policy often fails due to interference from nearby distractor objects, while our method maintains stable performance.

B. Model-Based Control

We study a planar pushing task inspired by Push Anything [5], depicted in Fig 10, Fig 11. The goal is for the robot's end-effector to push a goal object ("G") to a target pose (transparent "G"). This requires reasoning over object-object, object-robot, and object-ground contact interactions. The scene is cluttered with many additional objects compared to Push Anything [5], which was limited to four objects at a time.

Simulation Experiments: We evaluate the effect of task-focused scene abstraction in simulation by varying the number of objects present in the environment and comparing against the baseline controller C3+ that operates over the full scene. Baseline-C3+ does not perform any relevance estimation and reasons over all objects uniformly, while VLM-Focus-C3+ prunes task-irrelevant objects as described above. The goal is to push a randomly selected goal object in the scene to the table center position with correct orientation.

Figure 7 reports average execution time as a function of scene clutter. The results for each object are averaged across three trials. VLM-Focus is allowed to run for up to three iterations of scene refinement, with a similar time restriction on the baseline. For all trials, VLM-Focus successfully reached the target. The baseline failed on 5/18 trials.

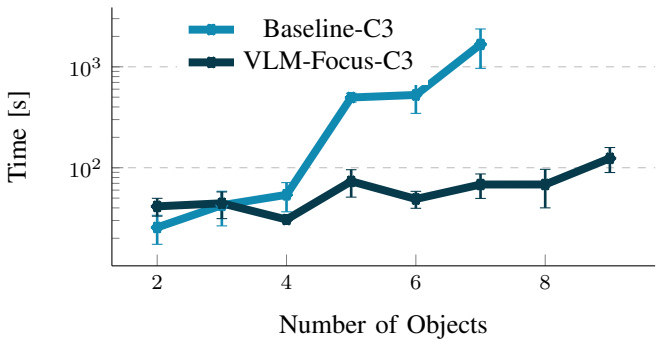

Fig. 7. Object Count Ablation. As the number of objects increases, the baseline execution time increases drastically, while VLM-Focus remains constant.

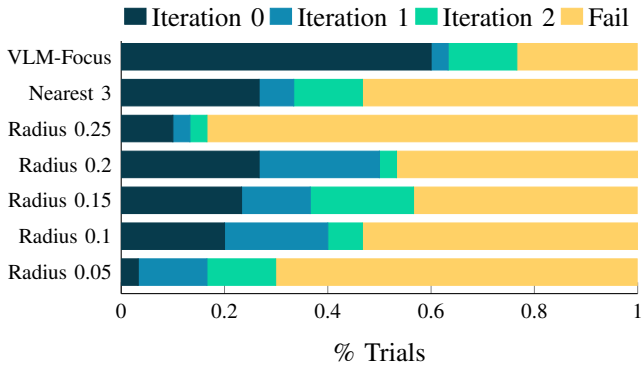

Fig. 8. C3+ distance-based pruning baseline (n=30). Nearest-k objects or nearest objects in r radius are passed to C3+. VLM-Focus outperforms the heuristics by capturing task-relevant objects regardless of proximity.

For small scenes (2–4 objects), the baseline and VLM-based method exhibit comparable performance. In this regime, the overhead of abstraction provides limited benefit, as the full scene remains tractable for the underlying planner and controller. As the number of objects increases, however, baseline-C3+ sharply degrades in performance. At 5 objects, execution time increases by an order of magnitude, and beyond 6 objects, it becomes prohibitively slow, exceeding 500 seconds on average. At 7 objects, it reaches over 1600 seconds.

In contrast, VLM-Focus-C3+ maintains stable performance across all tested clutter levels. Execution time remains within a narrow range (approximately 30–125 seconds) even as the number of objects increases from 4 to 9. Notably, performance does not degrade monotonically with object count, indicating that the effective problem size is determined by task relevance rather than raw scene complexity.

These results demonstrate that the proposed abstraction layer fundamentally alters the scaling behavior of the system. While the baseline suffers from combinatorial growth due to explicit reasoning over all objects and contacts, the VLM-guided pruning reduces the effective state and constraint space seen by the downstream controller. As a result, the system remains tractable in scenes where the baseline approach fails.

Simulation Baselines: We compared VLM-Focus to distance-based geometric pruning baselines for C3+ (Figure 8). These baselines select the k spatially nearest objects to the goal and pass them to the downstream system. Results show that geometric proximity is a reasonable heuristic, but task success may require more distant objects for multi-step planning. This supports the claim that the VLM's task-relevant reasoning adds value beyond simple spatial heuristics, which typically require human-engineering and fine-tuning.

Real Robot Experiments: We evaluate two variations of VLM-Focus, one which utilizes only pruning and one that utilizes only merging. The goal object is always labeled "G." Pruning is best suited for focused tasks where parts of the scene can be safely ignored. In these experiments, success requires the goal object to reach a specified position and orientation, enabling precise evaluation. Merging is suited for gross

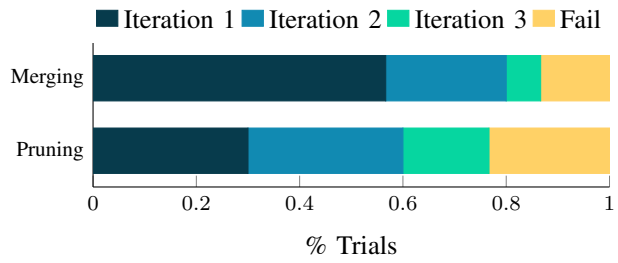

Fig. 9. VLM-Focus's task outcomes are shown for Merging and Pruning experiments on hardware. Task successes are recorded for across iterations of VLM requeries.

movements in cluttered scenes where the environment cannot be fully ignored but can be simplified. In these experiments, the goal object is pushed across the table through clutter, and success is defined using a looser position threshold; orientation is not evaluated.

Fig 9 shows the success rate, broken down by the number of required iterations of scene object merging and pruning. We limit our experiments to at most three iterations of scene reduction. Across both strategies, the cumulative success rate is high, with a 77% success rate for pruning and 87% success rate for merging. The experiments indicate that iterative VLM feedback enables the controller to recover from initial failures. Notably, success is distributed across iterations rather than concentrated in the first loop, highlighting the importance of re-querying the VLM during execution.

For pruning, successes are mostly achieved during the first or second iterations, with some additional recoveries occurring in the third iteration. The experiment was limited to three feedback loops. This suggests that while aggressive object removal may initially omit relevant contacts, subsequent VLM feedback can correct for errors and enable task completion.

Fig. 10. Three iterations of the pruned scene graph at various times during an episode of multi-object planar pushing (Sec V-B). Pruned objects are shown in gray. Successive iterations refine the abstraction to adapt to failures, mistakes, or evolving environment states. Goal location for target "G" is overlayed.

Fig. 11. Three iterations of the merged scene graph at various times during an episode of multi-object planar pushing (Sec V-B). Merged groups, shown with uniform colors, become more focused over iterations.

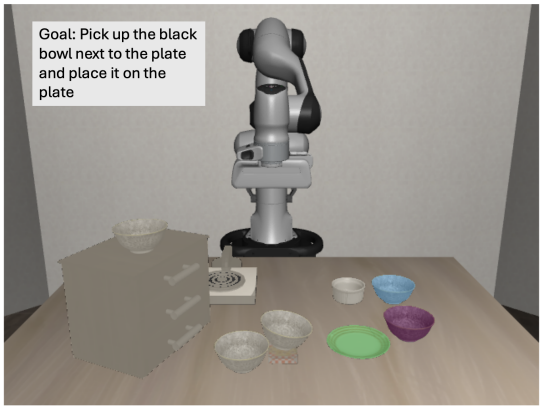

Fig. 12. VLA Pruning Visualization. Grey objects in the scene have been pruned away, while objects with colored mask overlays are identified as semantically and spatially goal relevant. The goal (masked green) and target objects (masked purple) are selected. In addition, a second distractor goal is selected due to ambiguity in the prompt.

Failure cases included timeouts and workspace limits. A visualization of three feedback loops during a real robot experiment for pruning is shown in Fig 10. Over iterations, the Relevant set of objects is adapted to the changing "G" goal object location.

For merging, the majority of successful executions occur in the first iteration, with fewer recoveries in later loops. This indicates that stable object groupings often yield effective abstractions early, reducing the need for further refinement. An example of real hardware experiment with three feedback loops is displayed in Fig 11.

Overall, these results demonstrate that iterative VLM-driven abstraction provides a powerful feedback mechanism for model-based control, enabling recovery from execution errors while maintaining high overall success on hardware.

C. Vision Language Action Models

Simulation Experiments: We next evaluate task-focused scene abstraction with a Vision–Language Action (VLA) using the Spatial LIBERO benchmark with a $\pi_{0.5}$ policy. Spatial LIBERO consists of tabletop manipulation tasks that require reasoning over relative spatial relationships, such as placing, stacking, or aligning a single object type (bowl) in varying positions and arrangements across different scenes. To stress test semantic and spatial reasoning, we introduce additional instances of the goal object beyond the original Spatial LIBERO scenes, increasing visual ambiguity and adding distractions. An example scene and VLM pruning selection is depicted in Figure 12.

Task success rates averaged over 100 evaluation episodes across 10 tasks (Figure 13) show that clutter significantly degrades VLA performance, with success dropping from 0.85 to 0.5, in keeping with prior reports [30, 8, 41]. VLM-Focus recovers a substantial portion of this performance, improving success to 0.70. While scene filtering does not fully match the uncluttered baseline, the remaining gap highlights limitations of both imperfect abstraction and the underlying

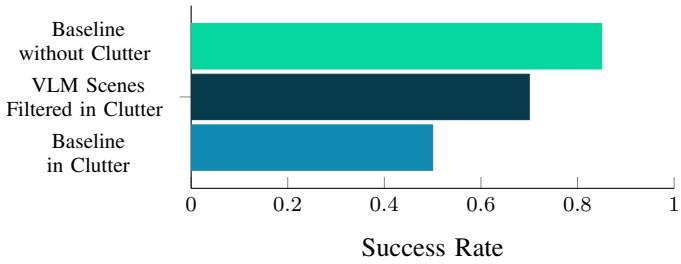

Fig. 13. The VLA is executed directly on the cluttered scene in Baseline in Clutter. In VLM Scene Filtered in Clutter, task-irrelevant objects are removed using our VLM-based scene abstraction prior to querying the VLA. Baseline without Clutter corresponds to the original, uncluttered LIBERO setup.

VLA's robustness to distribution shift. Overall, these results indicate that task-focused scene abstraction is an effective and lightweight mechanism for improving VLA performance in visually cluttered environments.

In simulation, object masking is implemented using ground-truth segmentation rather than learned detectors. This choice is motivated by the observation that open-vocabulary detectors such as GroundingDINO [24] perform unreliably in simulated scenes due to domain mismatch, leading to degraded masking quality unrelated to the abstraction mechanism itself. To verify that performance gains are not dependent on idealized inpainting, we additionally evaluated a pipeline using ground-truth segmentation followed by LaMa-based [33] inpainting and a simplified prompt (Appendix B). This configuration achieved a success rate of 0.69, comparable to 0.70 with the full prompt (Appendix A) and idealized inpainting.

Real Robot Experiments: We now perform a real hardware VLA experiment (Figure 14, Table I) where VLM-Focus uses a perception pipeline without privileged bounding box access. Given the overhead camera and hand images, VLM-Focus selects task-relevant and irrelevant groupings, which are passed to GroundingDINO [24]. GroundingDINO creates masks of task-irrelevant objects via SegmentAnything [16] for the hand image, before the image and masks are passed to LaMa [33] for hand image inpainting. The task setup consists of $\pi_{0.5}$ being queried to pick up one of six diverse fruit plushies and placing it in either the pink or purple bowl. For this setting, we adapt the VLA prompt (Appendix C) to explicitly support both positive and negative object selection (e.g., specifying the target object while excluding distractors) given an image. The performance gains persist, confirming they reflect scene abstraction.

Fig. 14. Real hardware VLA experiment. VLM-Focus is applied directly on hand image (b) without simulator bounding boxes. Foundation models are used to segment and inpaint scenes (c). Performance improvements persist, supporting the validity of the simulation comparison.

Method	Pineapple	Pear	Banana	Apple
VLM Scenes Filtered in Clutter	0.5	0.3	0.3	0.7
Baseline in Clutter	0.2	0.0	0.0	0.0

TABLE I
HARDWARE VLA ($\pi_{0.5}$) RESULTS (N=10 PER FRUIT). VLM-FOCUS CONSISTENTLY IMPROVES OVER THE BASELINE ACROSS TASKS.

Benchmark	Task	VLM-Focus (Clean Prompt)	Baseline
LIBERO	Overall	0.69	0.50
TAMP	Blocked	0.52	0.20
TAMP	Combined	0.58	0.32
C3+	Pruning	0.77	—
C3+	Merging	0.00	—

TABLE II
PERFORMANCE USING A SIMPLIFIED (CLEAN) PROMPT ACROSS BENCHMARKS. RESULTS REMAIN COMPARABLE TO THE ORIGINAL PROMPT, INDICATING THAT GAINS ARE NOT SENSITIVE TO PROMPT ENGINEERING.

VI. Prompt Sensitivity

Our core prompt structure (goal/obstruction identification, output format) is unchanged across tasks and backends; backend-specific constraints are isolated to a dedicated *Model Information* field describing each model's requirements (e.g., C3+'s object budget). A significantly simplified prompt (Appendix B) maintains similar performance (Table II) in simulation, suggesting gains are not prompt-specific.

VII. Limitations

We note a few limitations and corresponding directions for future work. While we provide a feedback mechanism for VLM-Focus, and note that one could potentially validate the abstract representation in simulation, this is computationally prohibitive for some controllers. There remains the possibility that the pruned system leads to irrecoverable failures. Additionally, the feedback mechanism is basic, with substantial opportunity to provide more explicit reprompting (e.g. with the cause of failure). Lastly, VLM-Focus does not learn from its mistakes, and future work could augment it with fine tuning or reinforcement learning.

VIII. Conclusion

We introduced VLM-Focus, a general and task-agnostic framework for constructing task-focused, dynamic scene abstractions using vision-language models and task feedback. Rather than reasoning over complete and static world models, our approach explicitly reshapes the scene representation itself by pruning irrelevant objects and merging functionally coupled ones, producing a compact, physically grounded abstraction that matches the needs of the current task.

We showed that this abstraction layer can be integrated as a front-end to existing planning and control pipelines without retraining or modifying the underlying algorithms. Across a range of contact-rich, highly cluttered manipulation scenarios, VLM-Focus improves scalability, robustness, and computational efficiency for both classical planning and model-based control, while maintaining or improving task success.

In particular, we find that merging enables more effective reasoning about collective motion and contact interactions that are difficult to handle at the level of individual objects.

More broadly, this work argues that as robots are deployed in increasingly complex and unstructured environments, the ability to dynamically construct minimal, task-conditioned world models will be critical for scaling both model-based and learning-based methods. We believe that VLM-Focus represents a step toward this direction, and that future work can extend these ideas to richer abstraction operations, tighter integration with perception, and long-horizon multi-stage tasks where the notion of relevance evolves over time.

Acknowledgments

This work was supported by an NSF CAREER Award under Grant No. FRR-2238480.

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as i can and not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*, 2022.

[2] Alp Aydinoglu, Adam Wei, Wei-Cheng Huang, and Michael Posa. Consensus complementarity control for multi-contact mpc. *IEEE Transactions on Robotics (TRO)*, July 2024. doi: 10.1109/TRO.2024.3435423. URL https://ieeexplore.ieee.org/document/10614849.

[3] Cornelius V Braun, Joaquim Ortiz-Haro, Marc Toussaint, and Ozgur S Oguz. Rhh-lgp: Receding horizon and heuristics-based logic-geometric programming for task and motion planning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13761–13768. IEEE, 2022.

[4] Hien Bui and Michael Posa. Enhancing task performance of learned simplified models via reinforcement learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9212–9219. IEEE, 2024.

[5] Hien Bui*, Yufeiyang Gao*, Haoran Yang*, Eric Cui, Siddhant Mody, Brian Acosta, Thomas Stephen Felix, Bibit Bianchini, and Michael Posa. Push anything: Single- and multi-object pushing from first sight with contact-implicit mpc. *arXiv preprint arXiv:2510.19974*, 2025.

[6] Xianyi Cheng, Eric Huang, Yifan Hou, and Matthew T Mason. Contact mode guided sampling-based planning for quasistatic dexterous manipulation in 2d. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6520–6526. IEEE, 2021.

[7] Neil T Dantam, Zachary K Kingston, Swarat Chaudhuri, and Lydia E Kavraki. Incremental task and motion planning: A constraint-based approach. In *Robotics: Science and systems*, volume 12, page 00052. Ann Arbor, MI, USA, 2016.

[8] Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

[9] Yunhai Feng, Jiaming Han, Zhuoran Yang, Xiangyu Yue, Sergey Levine, and Jianlan Luo. Reflective planning: Vision-language models for multi-stage long-horizon robotic manipulation. In Joseph Lim, Shuran Song, and Hae-Won Park, editors, *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 2038–2062. PMLR, 27–30 Sep 2025. URL https://proceedings.mlr.press/v305/feng25b.html.

[10] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. In *Proceedings of the international conference on automated planning and scheduling*, volume 30, pages 440–448, 2020.

[11] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4(1):265–293, 2021.

[12] Michael Görner, Robert Haschke, Helge Ritter, and Jianwei Zhang. Moveit! task constructor for task-level motion planning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 190–196. IEEE, 2019.

[13] Chi-Pin Huang, Yueh-Hua Wu, Min-Hung Chen, Yu-Chiang Frank Wang, and Fu-En Yang. Thinkact: Vision-language-action reasoning via reinforced visual latent planning. *arXiv preprint arXiv:2507.16815*, 2025.

[14] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

[15] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[16] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything.

arXiv:2304.02643, 2023.

[17] Nishanth Kumar, William Shen, Fabio Ramos, Dieter Fox, Tomás Lozano-Pérez, Leslie Pack Kaelbling, and Caelan Reed Garrett. Open-world task and motion planning via vision-language model inferred constraints. *arXiv preprint arXiv:2411.08253*, 2024.

[18] Simon Le Cleac'h, Taylor A Howell, Shuo Yang, Chi-Yen Lee, John Zhang, Arun Bishop, Mac Schwager, and Zachary Manchester. Fast contact-implicit model predictive control. *IEEE Transactions on Robotics*, 40: 1617–1629, 2024.

[19] Dongryung Lee, Sejune Joo, Kimin Lee, and Beomjoon Kim. Prime the search: Using large language models for guiding geometric task and motion planning by warm-starting tree search. *The International Journal of Robotics Research*, page 02783649251347307, 2024.

[20] Albert H Li, Preston Culbertson, Vince Kurtz, and Aaron D Ames. Drop: Dexterous reorientation via online planning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14299–14306. IEEE, 2025.

[21] 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 *arXiv preprint arXiv:2209.07753*, 2022.

[22] Fanqi Lin, Ruiqian Nai, Yingdong Hu, Jiacheng You, Junming Zhao, and Yang Gao. Onetwovla: A unified vision-language-action model with adaptive reasoning. *arXiv preprint arXiv:2505.11917*, 2025.

[23] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *arXiv preprint arXiv:2306.03310*, 2023.

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

[25] Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. PDDL-the planning domain definition language, 1998.

[26] Reihaneh Mirjalili, Tobias Jülg, Florian Walter, and Wolfram Burgard. Augmented reality for robots (arro): Pointing visuomotor policies towards visual robustness. *IEEE Robotics and Automation Letters*, 2026.

[27] Nataliya Nechyporenko, Yutong Zhang, Sean Campbell, and Alessandro Roncone. Morphit: Flexible spherical approximation of robot morphology for representation-driven adaptation. *arXiv preprint arXiv:2507.14061*, 2025.

[28] Son Nguyen, Ozgur Oguz, Valentin Hartmann, and Marc Toussaint. Self-supervised learning of scene-graph representations for robotic sequential manipulation planning. In Jens Kober, Fabio Ramos, and Claire Tomlin, editors, *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 2104–2119. PMLR, 16–18 Nov 2021. URL https://proceedings.mlr.press/v155/nguyen21b.html.

[29] Jianing Qian, Yunshuang Li, Bernadette Bucher, and Dinesh Jayaraman. Task-oriented hierarchical object decomposition for visuomotor control. *arXiv preprint arXiv:2411.01284*, 2024.

[30] Amir Rasouli, Montgomery Alban, Sajjad Pakdamansavoji, Zhiyuan Li, Zhanguang Zhang, Aaron Wu, and Xuan Zhao. Distracted robot: How visual clutter undermine robotic manipulation. *arXiv preprint arXiv:2511.22780*, 2025.

[31] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Proceedings of the 5th Conference on Robot Learning (CoRL)*, 2021.

[32] Tom Silver, Rohan Chitnis, Aidan Curtis, Joshua B Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Planning with learned object importance in large problem instances using graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11962–11971, 2021.

[33] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. *arXiv preprint arXiv:2109.07161*, 2021.

[34] Marc Toussaint. Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *IJCAI*, pages 1930–1936, 2015.

[35] Sharanya Venkatesh, Bibit Bianchini, Alp Aydinoglu, William Yang, and Michael Posa. Approximating global contact-implicit mpc via sampling and local complementarity. *arXiv preprint arXiv:2505.13350*, 2025.

[36] Shu Wang, Muzhi Han, Ziyuan Jiao, Zeyu Zhang, Ying Nian Wu, Song-Chun Zhu, and Hangxin Liu. Llmˆ 3: Large language model-based task and motion planning with motion failure reasoning. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12086–12092. IEEE, 2024.

[37] Yixuan Wang, Yunzhu Li, Katherine Driggs-Campbell, Li Fei-Fei, and Jiajun Wu. Dynamic-resolution model learning for object pile manipulation. *arXiv preprint arXiv:2306.16700*, 2023.

[38] Muyang Yan, Miras Mengdibayev, Ardon Floros, Weihang Guo, Lydia E Kavraki, and Zachary Kingston. Using vlm reasoning to constrain task and motion planning. *arXiv preprint arXiv:2510.25548*, 2025.

[39] Yi Yang, Jiaxuan Sun, Siqi Kou, Yihan Wang, and Zhijie Deng. Lohovla: A unified vision-language-action model for long-horizon embodied tasks. *arXiv preprint arXiv:2506.00411*, 2025.

[40] Zhutian Yang, Caelan Garrett, Dieter Fox, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Guiding long-

horizon task and motion planning with vision language models, 2024. URL https://arxiv.org/abs/2410.02193.

[41] Borong Zhang, Jiahao Li, Jiachen Shen, Yishuai Cai, Yuhao Zhang, Yuanpei Chen, Juntao Dai, Jiaming Ji, and Yaodong Yang. Vla-arena: An open-source framework for benchmarking vision-language-action models. *arXiv preprint arXiv:2512.22539*, 2025.

[42] Jesse Zhang, Marius Memmel, Kevin Kim, Dieter Fox, Jesse Thomason, Fabio Ramos, Erdem Bıyık, Abhishek Gupta, and Anqi Li. Peek: Guiding and minimal image representations for zero-shot generalization of robot manipulation policies. *arXiv preprint arXiv:2509.18282*, 2025.

[43] Yan Zhang, Teng Xue, Amirreza Razmjoo, and Sylvain Calinon. Learn2decompose: Learning problem decomposition for efficient sequential multi-object manipulation planning. *arXiv preprint arXiv:2408.06843*, 2024.

[44] Yifeng Zhu, Jonathan Tremblay, Stan Birchfield, and Yuke Zhu. Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graphs. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6541–6548. Ieee, 2021.

APPENDIX A

VLM Prompt

You are a precise symbolic robot planner assistant. Your task is to categorize objects into lists based on physical properties and spatial relations.

SCENE DATA
- Environment: You are operating in a cluttered tabletop scene.
- Goal: {goal}
- Objects: {objects}
- Stacked: {stacked}

MODEL INFORMATION
{model information}

CONSTRAINTS & DEFINITIONS
- HEAVY, LIGHT objects are determined by masses if given. This is relative to task. Determine which objects are heavy and light.
- RULE A: List [1] must include ALL Goal objects AND TASK RELEVANT MOVABLE OBJECTS BEYOND GOAL OBJECTS. Include objects near the path to the goal or target table (ordered by importance).
Identify non-goal IDs that are numerically closest in (x, y, z) coordinates to a goal object or open a path for robot.
DO NOT INCLUDE FAR OR IRRELEVANT OBJECTS.
- RULE B: For List [1], DO NOT JUST INCLUDE goal and stacked objects. You MUST include other objects relevant to goal.
- RULE C: List [3] Objects whose mass and dynamics allow them to be safely ignored for collision checks. Think about their weights.
- RULE D: HEAVY objects are strictly forbidden from List [3]. If relevant, must go in List [1].
- RULE E: In List [2], every sub-list must start with the BASE supporting object if it is a stack!!!!!

STEP-BY-STEP REASONING PROCESS
1. Goal Identification: Which objects are part of the 'goal'?
2. Mass Analysis: Categorize every object as HEAVY, LIGHT, or MEDIUM if weights given.
3. Obstruction Check: Looking at the positions and the 'stacked' relations, which objects are either:
- Supporting a goal object?
- Clutter obstructing a goal object?
4. List Compilation:
- List [1]: Combine Goal objects + relevant HEAVY, LIGHT, and MEDIUM relevant obstructions.
- List [2]: Group objects based on what objects could be reasoned about together/move together.
If 'stacked' relations, ensuring the BASE is first. If objects have SAME base, COMBINE list. Do not list single object lists.
- List [3]: Include only LIGHT objects that are not vital to move manually but can be pushed aside. HEAVY objects cannot be pushed aside.

OUTPUT FORMAT
1. Reasoning: Output your short reasoning
2. Result: Output a single line of lists separated by semicolons.
[List 1]; [List 2]; [List 3]

FINAL OUTPUT
Reasoning:
Result:

The VLM prompt mentions list 3, which is used for future work. The outputs of list 3 are not utilized in this work. Individual Model Information is listed below. Code will be made available upon acceptance.

C3+ Pruning

You are a model based controller. You are trying to reduce number of objects to reason over to reduce linear complimentarity problem size and contact constraints.
Mergings in this model are object groupings (excluding the goal object) to be reasoned about together for computational efficiency. They are encouraged.
Choose less than 4 objects for list [1]]!

C3+ Merging

You are a model based controller. You are trying to reduce number of objects to reason over to reduce linear complimentarity problem size and contact constraints.
Mergings in this model are object groupings (excluding the goal object) to be reasoned about together for computatinal efficiency.
In this experiment, you should ignore list 1 and only focus on creating list 2 of relevant, goal-oriented mergings.
At most 4 groups in merge, and try not to have large merges!!!!

TAMP Cluttered Tabletop

You are filtering the scene for a TAMP planner. The robot is a PR2 movable base robot located at pos = [2.5, 0, 0]. This is a cluttered tabletop scene. You should reduce the clutter to reason over.

TAMP Combined Tray and Clutter

You are filtering the scene for a TAMP planner. The robot is a PR2 movable base robot located at pos = [2.5, 0, 0]. This is a tabletop scene with mutliple goal objects. List [2] object groupings MUST be stably movable together (cannot just be a cluster). Try to include any (heavy or light) clutter beyond base into list [1]. There is clutter around individual goal object on table.

Libero Spatial

You are filtering a cluttered scene for a pi0.5 VLA. You are trying to SEMANTICALLY/SPATIALLY SIMPLIFY THE SCENE. Focus on what goal semantics say and spatial relevancy of objects. Remove objects that would confuse the goal! Keep in distribution with trained Spatial Libero dataset.

APPENDIX B

Condensed VLM Prompt

You are a precise symbolic robot planner assistant. Your task is to categorize objects into lists based on physical properties and spatial relations.

SCENE DATA
- Environment: You are operating in a cluttered tabletop scene.
- Goal: {goal}
- Objects: {objects}
- Stacked: {stacked}

MODEL INFORMATION
You are filtering the scene for a TAMP planner. This is a tabletop scene with multiple goal objects.
List [2] object groupings MUST be stably movable together (cannot just be a cluster).

RULES
1. Include ALL goal objects and any clutter obstructing or near them. Do NOT omit objects between robot and goal.
2. Each sublist groups objects that can be stably moved together (not just spatial clusters).
3. Objects sharing the same base must be combined into one sublist — base object listed first.
4. Standalone relevant objects with no group are listed as singletons: [obj].
5. Order sublists by task importance.

REASONING STEPS
1. **Goals:** Which objects are part of 'goal'?
2. **Obstructions:** Which objects are stacked on, supporting, or blocking a goal object?
3. **Grouping:** Which objects can move together? Merge any that share a base.
4. **Order:** Base first within each sublist. Most important sublist first overall.

OUTPUT FORMAT
Reasoning: one line, 30 tokens
Result: [[base, obj, ...], [obj], [base, obj, ...], ...]

FINAL OUTPUT
Reasoning:
Result:

APPENDIX C

VLA Hardware

You are helping a robot manipulation pipeline clean up camera images.

The robot's current task is: "{goal}"

Look at the robot workspace image(s) and identify:
1. Objects RELEVANT to the task (to be PROTECTED from removal)
2. DISTRACTOR objects on the workspace (to be REMOVED from the image)

Rules for RELEVANT objects (protect):
- ALWAYS include: the robot arm, gripper, and end effector (if visible).
The gripper/end effector is the most critical — it includes the finger pads, gripper jaws, and anything the robot is currently grasping. NEVER remove it.
- ALWAYS include: any object the robot needs to interact with for the task
- ALWAYS include: any container/bowl/plate the target object is sitting in or on
- ALWAYS include: any goal location (basket, bin, tray, bowl) mentioned in the task
- DO NOT include: background furniture (table, shelf, wall, floor)

Rules for DISTRACTOR objects (remove):
- List EVERY single visible movable object that is NOT the task target or robot
- Include each fruit, toy, container, bowl, plate separately — do NOT skip any
- If an object is inside a bowl/plate, list BOTH the object AND the bowl/plate
- Even if you are unsure whether something is relevant, include it as a distractor unless the task description specifically mentions it
- DO NOT include: the robot arm, gripper, end effector, or task-relevant objects
- DO NOT include: background furniture, table, shelf, wall, floor

Respond with ONLY a JSON object, no explanation:

```
{
  "relevant_objects": [
    {
      "label": "short name",
      "visual_description": "color, shape, material (e.g. orange-red round plush)"
    }
  ],
  "distractors": [
    {
      "label": "specific name",
      "visual_description": "color, shape, material"
    }
  ]
}
```

CRITICAL labeling rules:
- Use the SAME object names as the task description when possible. If the task says "apple", label it "apple" — NOT "orange fruit" or "red ball".
- These may be toy/fake versions of objects. A toy apple is still "apple".
- For distractors, be SPECIFIC: use color + object type (e.g. "yellow banana toy", "purple plum toy", "teal bowl") so each distractor can be individually detected.
- visual_description should describe what the object LOOKS LIKE in the image (color, shape, texture) — this helps the detection model find the right object.
- Only include objects that are EXPLICITLY part of the task as relevant.
- Do NOT include relevant objects in the distractors list.
- It is CRITICAL that you list ALL distractors. Missing even one is a failure.
Example relevant: "robot arm", "apple"
Example distractors: "pineapple toy", "yellow banana toy", "purple plum toy", "orange slice toy", "teal bowl", "gold bowl", "dark blue bowl", "pink bowl"