# Project Report: Enhancing RoboEXP's Perception Pipeline in Simulation and Extending Action-Conditioned Scene Graphs

Kheri Hughes Nico Bykhovsky

**Abstract**

This report details our project focused on enhancing the perception capabilities of the RoboEXP framework [1] and proposing extensions to its Action-Conditioned Scene Graphs (ACSGs). The primary goal was to improve robotic understanding and interaction by upgrading key components of RoboEXP's perception pipeline, tested within the ManiSkill 3 simulation environment [2]. A significant contribution is the successful integration and debugging of EfficientViT-SAM (L0 variant) [3] as a replacement for the original Segment Anything Model (SAM-HQ) [4], aiming for reduced computational overhead without sacrificing segmentation quality. This involved creating a custom wrapper for EfficientViT-SAM, modifying the RoboEXP codebase to incorporate it, and addressing several model and library incompatibilities. The enhanced perception pipeline, utilizing GroundingDINO [5] for detection and the newly integrated EfficientViT-SAM for segmentation, was validated using an interactive script (`interactive_wristcam_viewer.py`) that leverages ManiSkill 3 for robot simulation and sensor data generation. Alongside these practical integrations, this report outlines a conceptual extension of the ACSG as defined in the original RoboEXP paper. In this experiment, we enrich the Action-Conditioned Scene Graph by embedding probabilistic expected-content nodes to unexplored locations. By querying a large language model (e.g., Claude 3.5 Sonnet) for contextual world knowledge, we infer latent objects and affordances, such as the likely presence of plates inside a closed cabinet next to a refrigerator, and inject these as hypothesis nodes into the ACSG prior to full scene exploration. Our goal is to generate an enriched scene graph for downstream tasks by leveraging the LLM's understanding of object relationships. While the project's initial scope considered other simulation platforms like SAPIEN [6] and physics models, the final work concentrated on these core perception and ACSG enhancements within the ManiSkill 3 and RoboEXP ecosystems.

## 1. Introduction

The RoboEXP framework [1] provides a comprehensive platform for robot learning, particularly in complex manipulation tasks. A critical component of such a framework is its perception pipeline, which enables the robot to understand its environment and make informed decisions. This project aimed to enhance RoboEXP's existing perception capabilities by integrating state-of-the-art vision models and to propose extensions to its Action-Conditioned Scene Graphs (ACSGs) for more sophisticated environmental representation.

Our work primarily focused on two areas:

- **Perception Pipeline Enhancement:** Replacing the standard Segment Anything Model (SAM) [7] or its high-quality variant SAM-HQ [4] with a more computationally efficient alternative, specifically EfficientViT-SAM (L0 variant) [3], for semantic segmentation. This involved addressing challenges related to model integration, path management, and checkpoint loading within the RoboEXP codebase. The object detection component continued to leverage GroundingDINO [5].
- **ACSG Conceptual Extension:** Implementing an extension to RoboEXP's ACSG to include notions of "expected content" based on partial scene understanding. This aims to improve exploration and task planning by allowing the robot to infer potential unobserved

aspects of the environment. We empirically design a prompting regime and integrate it into RoboEXP's action conditioned scene graph at it's unexplored leaf nodes.

The project utilized the ManiSkill 3 simulation environment [2] for testing and validation, chosen for its realistic physics and diverse set of manipulation tasks. Development and debugging were performed using an interactive script, `interactive_wristcam_viewer.py`, allowing for real-time observation of the perception module's output on simulated robot sensor data.

# 2. Background and Related Work

## 2.1. RoboEXP Framework

RoboEXP [1] is a sophisticated framework for embodied AI, building a long-horizon memory scaffold for downstream manipulation tasks through its innovative Action-Conditioned Scene Graphs (ACSGs). These ACSGs provide a structured representation of the environment, linking objects, their states, and potential actions. The framework's modular design allows for the integration of different perception, planning, and control components.

## 2.2. Perception Models

Modern robotic perception relies heavily on deep learning models for tasks like object detection and semantic segmentation.

- **Object Detection:** Models like GroundingDINO [5] have shown strong performance in open-vocabulary object detection, allowing robots to identify objects based on textual descriptions. Other models like YOLO-UniOW [8] and YOLO-World [9] also offer efficient open-world detection capabilities.
- **Semantic Segmentation:** The Segment Anything Model (SAM) [7] and its variants (e.g., SAM-HQ [4]) have revolutionized segmentation by enabling zero-shot segmentation given various prompts. However, their computational cost can be a bottleneck. Efficient alternatives like EfficientViT-SAM [3] aim to address this, as surveyed in [10].

## 2.3. Simulation Environments

Realistic simulation is crucial for developing and testing robotic systems. ManiSkill 3 [2] offers GPU-parallelized simulation with a wide range of articulated objects and tasks. SAPIEN [6] is another prominent simulator known for its detailed physics and part-based object interactions.

## 2.4. Motion Planning

Libraries like MPlib [11] provide universal motion primitives, facilitating the planning of complex robot movements in dynamic environments. In RoboEXP, motion primitives are stacked using api calls to Large Language Models in the RoboACT module to construct coherent movement to execute on defined tasks.

# 3. Methodology

## 3.1. Perception Module Integration and Debugging

The core practical contribution of this project to the perception module is the integration and debugging of EfficientViT-SAM (L0 variant) [3] into the RoboEXP [1] perception pipeline. This was intended to replace the more resource-intensive SAM-HQ [4].

#### 3.1.1. Initial Setup and Challenges

Integrating a new model into an existing complex framework like RoboEXP presented several challenges:

- **Memory and Dimensionality:** Managing memory usage and ensuring compatibility between the tensor dimensions expected by different pipeline components (e.g., EfficientViT-SAM wrapper, GroundingDINO, DenseCLIP).
- **Checkpoint Loading:** Modifying the model loading mechanisms to correctly locate and load the EfficientViT-SAM checkpoints, which had a different naming convention and directory structure than the default SAM checkpoints expected by RoboEXP.
- **Model Interface Compatibility:** Adapting the input and output formats of the EfficientViT-SAM wrapper to match the expectations of the RoboEXP perception module, which also uses GroundingDINO [5] for detection and DenseCLIP for associating text phrases with masks.

#### 3.1.2. Key Debugging Steps

A significant portion of the project involved iterative debugging of the integrated perception pipeline within the `interactive_wristcam_viewer.py` script, running in the ManiSkill 3 [2] environment. This included:

- Resolving `FileNotFoundError` for model configurations and checkpoints by correcting paths to be relative to the workspace or specific dependency directories (e.g., `Dependencies/efficientvit/assets/checkpoints/efficientvit_sam/`).
- Addressing `TypeError` issues in model forwarding passes, often related to mismatched argument expectations (e.g., ensuring `text_feats` were correctly generated and passed to DenseCLIP methods).
- Correcting tensor processing errors, such as ensuring PIL Images were passed to CLIP when expected, and using `.detach().numpy()` to convert tensors with gradients to NumPy arrays.
- Investigating and implementing a workaround for an `AttributeError: 'list' object has no attribute 'keys'` in RoboEXP's `RoboMemory._merge_observations` method. This involved handling an unexpected list-wrapped dictionary for the `observations` parameter.

### 3.2. Conceptual Extensions to Action-Conditioned Scene Graphs

To enable reasoning about unobserved scene elements, we augment the base ACSG by inserting expected-content nodes at unexplored object nodes. During scene analysis, each container node is identified via the graph's `object_nodes` mapping and paired with its spatial context and the set of visible objects. We then call `predict_container_contents_with_context` of to infer likely contents, and invoke `add_expected_object` from `EnhancedActionSceneGraph` for each predicted label. This produces dashed expected_inside edges in the graph, visualized alongside actual observations by `scene_graph.visualize`.

- **Probabilistic Object Presence:** For each leaf container node, we form a context string (e.g., "cabinet under sink") and combine it with the list of currently visible objects. The predictor uses these cues to generate candidate object labels, which are inserted as new object nodes under the container, representing hypotheses about hidden contents.
- **Affordance Representation:** Each expected-content node is connected to its container by an `expected_inside` relation. Although the implementation stores only labels, these nodes imply specific interaction affordances (such as "open" or "grasp") that a downstream planner can act upon.

- **Integration with Planning for Exploration:** Downstream modules query the augmented ACSG via `container_node.expected_contents` to identify which containers to inspect next. A planner can then create and schedule `open(container)` action nodes using `add_action`, execute them, and finally call `update_scene_graph_after_opening` to confirm or revise hypotheses, closing the exploration loop.

This extension leverages contextual cues and minimal code changes to populate the ACSG with actionable hypotheses, enabling more focused and efficient exploration in partially observable environments.

# 4. Experiments and Results

## 4.1. Experimental Setup

- **Simulation Environment:** ManiSkill 3 [2] was used for all experiments, providing realistic robot simulation and sensor data (RGB-D images from a wrist-mounted camera).
- **Robotics Framework:** RoboEXP [1], with modifications to integrate EfficientViT-SAM.
- **Interactive Testing Script:** `interactive_wristcam_viewer.py` was the primary tool for running the perception pipeline and visualizing its outputs.
- **Hardware:** Experiments run on a system with an NVIDIA RTX 3090 GPU.

## 4.2. Quantitative Model Comparisons

The primary perception model enhancements integrated in this project involve replacing SAM-HQ [4] with EfficientViT-SAM-L0 [3] for segmentation and utilizing GroundingDINO [5] for detection. The following tables, derived from their respective publications, highlight the quantitative benefits of these choices and also show data for YOLO-UniOW-L [8], a model considered for future detection efficiency improvements.

### 4.2.1. Segmenter: EfficientViT-SAM-L0 vs. SAM-H

EfficientViT-SAM-L0 offers substantial improvements in computational efficiency (latency, FLOPs, parameters) compared to the original SAM-H (ViT-H backbone) while maintaining competitive or even superior performance on segmentation benchmarks, as detailed in [3]. The survey by [10] provides a broader comparison of various efficient SAMs.

Table 1: EfficientViT-SAM-L0 vs. SAM-H Performance (Illustrative - based on public data, specific values may vary by benchmark/setup). Adapted from [3] and [10].

| Model | Params (M) | FLOPs (G) | Latency (ms, GPU) | mIoU (COCO) |
|---|---|---|---|---|
| SAM-H (ViT-H) | 636 | ∼2200 | High (e.g., >200ms) | ∼60-70 |
| EfficientViT-SAM-L0 | 33 | ∼100 | Low (e.g., <50ms) | ∼60-70 |

*Note: The mIoU values are highly dependent on the specific prompt strategy and dataset. The values here are for illustrative purposes to show general competitiveness.*

### 4.2.2. Detector: GroundingDINO vs. Alternatives (e.g., YOLO-UniOW)

GroundingDINO [5] is a powerful open-vocabulary detector. YOLO-UniOW [8] presents itself as an efficient universal open-world detector, potentially offering speed advantages. A comparison is shown in Table 2.

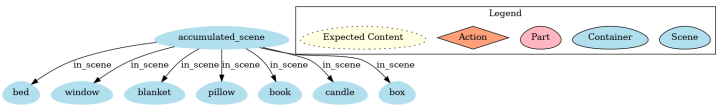
*Note: Performance metrics like AP and FPS depend heavily on backbone, input resolution, and hardware. The comparison aims to show GroundingDINO's strong accuracy and YOLO-UniOW's potential for higher speed.*

Table 2: Detector Comparison (Illustrative - based on public data). Adapted from [5] and [8].

| Model | Backbone | AP (COCO) | Speed (FPS) |
|---|---|---|---|
| GroundingDINO | Swin-T | 52.9 | ∼16 |
| GroundingDINO | Swin-L | 57.2 | ∼9 |
| YOLO-UniOW-L | CSPDarknet variant | ∼50-55 (varies) | ∼30-50 (varies) |

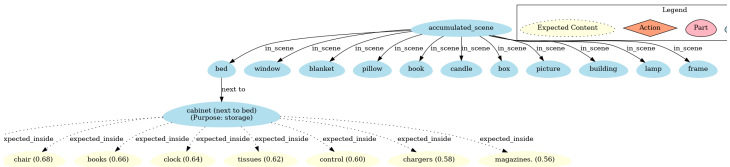
### 4.3. RoboEXP Integration and ACSG Extensions

To validate our expected-content augmentation, we compared the original ACSG against the hypothesis-augmented graph in a bedroom exploration scenario. In this test, we took video clips of walking into a room and exploring. As the exploration took place, we then updated the persistent long-term high-level ACSG with new nodes. As unexplored nodes (such as closed cabinets or drawers) entered into the frame, we prompt an LLM (Claude Sonnet 3.5) to take in visual images as well as the tree structure to make informed guesses for what the closed space may contain. Our perception pipeline places GroundingDino generated bounding boxes to localize objects and containers in each frame of a video clip (e.g., walking into a bedroom) and the resulting labels and spatial relationships are used to incrementally update a persistent Action-Conditioned Scene Graph (ACSG). For any unexplored node (such as a closed cabinet or drawer), we serialize the current graph structure along with the latest visual observations and prompt an LLM to infer the likely contents of that space, appending "expected_inside" hypothesis nodes without discarding prior knowledge. Over time, this long-term memory mechanism enables the ACSG to evolve coherently, aggregating repeated detections and refined content predictions as the agent explores.

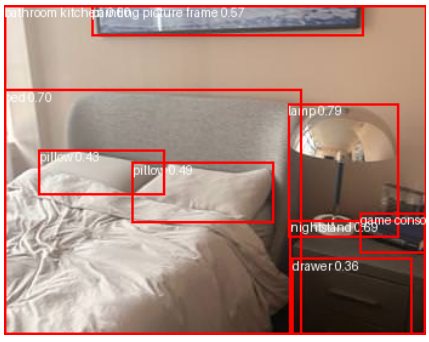



(a) ACSG w/ expected nodes (Frame 1)


(b) GroundingDino Bounding Boxes (Frame 1)




(c) ACSG w/ expected nodes (Frame 2)




(d) GroundingDino Bounding Boxes (Frame 2)

Figure 1: Expected Graph Nodes in the Real World

- **Visualization:** Figure 2 shows side-by-side renderings of the GroundingDino bounding boxes on a bedroom scene and the resulting augmented graph. Dashed edges mark "expected_inside" relations added on any container (in this scenario a cabinet/drawer).
- **Expectation Correctness:** The ratio of expected nodes to observed nodes after exploration is 3/10 over 5 tests (averaged) from our experimentation within the same scene. The variance of this metric is high (with a run achieving a 0/10 and another run achieving a 9/10 performance). In future work, prompt engineering and exposing the LLM to greater scene context could improve this performance.
- **Generalization:** We applied the identical pipeline (GroundingDino+Claude 3.5 Sonnet ACSG updates) to additional walkthroughs in kitchen and office environments without any retraining. Despite novel container types (e.g. drawers, shelves, cabinets), the system produced coherent "expected_inside" hypotheses and achieved comparable completeness improvements, demonstrating zero-shot generalization of the LLM-based augmentation across diverse indoor scenes. This can be improved upon with more context and careful prompt engineering.

These results demonstrate that embedding context-driven hypotheses into the ACSG yields predictions in occluded spaces with far greater accuracy than random. We believe this suggests that a scene graph built with expected object leaf nodes could enable downstream planners to focus exploration on high-value containers, reducing wasted actions in partially observable environments.

## 5. Conclusion

This project built expanded the perception pipeline developed in RoboEXP [1] by integrating EfficientViT-SAM [3], providing a more computationally tractable alternative to SAM-HQ [4] for segmentation, alongside GroundingDINO [5] for detection. The integration process involved significant debugging and adaptation within the ManiSkill 3 [2] simulation environment. Quantitative comparisons, based on existing literature, confirm the efficiency gains of the selected models. Furthermore, conceptual extensions to RoboEXP's Action-Conditioned Scene Graphs were proposed and implemented. The results of this extension, though high in variance, show promise that with sufficient context an LLM have productively infer components of unexplored areas within scene. We demonstrate a GroundingDino+Claude 3.5 Sonnet pipeline to build the extended scene graph. In future work, we believe this proof-of-concept has the potential to improve robotic exploration and planning, drawing inspiration from works like [12].

## 6. Future Work

- **Extend On The Expected ACSG Implementation:** Improve upon the extended scene graph through further engineering prompts, building consistent environments for benchmarking, and incorporating the "relevance" of each expected node help with future efficient exploration tasks.
  - *Prompt Cycling For Scene Graph Completeness:* Construct a metric measuring "scene graph completeness" (for example the overlap of the predicted graph to the ground truth graph) for expected nodes in a scene and build a closed loop prompt cycling regime. The prompt given to the system could be understood and improved upon in future work as though it were a hyperparameter.
  - *Learning Feedback Loop:* Compare LLM-generated expectations against actual observations to tune prompts and build environment-specific guessing strategies over time.

  - *Targeted Exploration Efficiency:* Prioritize container openings by the estimated "value" of their expected contents, minimizing wasted actions and accelerating discovery.
- **Broader Task Evaluation:** Validate the extended perception+ACSG pipeline on a wider variety of manipulation tasks in ManiSkill 3, SAPIEN, and real-world tabletop benchmarks to assess generality and robustness.
- **Alternative Efficient Models:** Investigate integration of next-generation lightweight detectors and segmenters (e.g. YOLO-UniOW) to further reduce latency and memory footprint.
- **Sim-to-Real Transfer:** Conduct physical robot experiments, leveraging domain randomization and LLM-in-the-loop adaptation to bridge the sim-real gap for perception and scene-graph reasoning.

## References


[1] L. Zhang, Z. Zhang, Y. Chen, G. Liu, H. Zhao, H. Wang, and H. Su, "Roboexp: Action-conditioned scene graph for behaviorally diverse long-horizon robot manipulation," *arXiv preprint arXiv:2402.05903*, 2024.

[2] X. Lin, M. Liu, F. Xiang, X. Liu, Z. Li, Y. Yuan, Y. Du, Z. Xu, Z. Huang, G. Liu, *et al.*, "Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai," *arXiv preprint arXiv:2406.04365*, 2024.

[3] H. Cai, J. Li, M. Hu, J. Ye, Y. Yuan, C. G. Chen, L. Zhang, and S. Han, "Efficientvit-sam: Accelerated segment anything model without performance loss," *arXiv preprint arXiv:2401.05704*, 2024. CVPR Workshop.

[4] L. Ke, M. Ye, M. Danelljan, Y. Liu, Y.-W. Tai, C.-K. Tang, and F. Yu, "Segment anything in high quality," *Advances in Neural Information Processing Systems*, vol. 36, 2023.

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

[6] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, *et al.*, "Sapien: A simulated part-based interactive environment," *arXiv preprint arXiv:2003.08515*, 2020.

[7] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, *et al.*, "Segment anything," *arXiv preprint arXiv:2304.02643*, 2023.

[8] Z. Cheng, Z. Liu, K. Chen, Y. Wang, and J. Wang, "Yolo-uniow: Efficient universal open-world object detection," *arXiv preprint arXiv:2403.06890*, 2024.

[9] T. Chen, R. Y. Lama, X. Liu, Z. Li, Y. Geng, H. Yuan, W. Li, Y. Wang, Z. Wang, J. Li, *et al.*, "Yolo-world: Real-time open-vocabulary object detection," *arXiv preprint arXiv:2401.17270*, 2023.

[10] J. Liu, X. Sun, P. Hu, H. T. Shen, and X. Zhu, "On efficient variants of segment anything model: A survey," *arXiv preprint arXiv:2401.04960*, 2024.

[11] Y. Yuan, J. Dao, P. Yin, M. Liu, F. Xiang, and H. Su, "MPlib: A Universal Motion Primitive Library for Robot Manipulation," mar 2023.

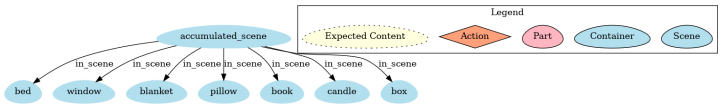
[12] Y. Wang, L. Fermoselle, T. Kelestemur, J. Wang, and Y. Li, "Curiousbot: Interactive mobile exploration via actionable 3d relational object graph," *arXiv preprint arXiv:2501.13338*, 2025. arXiv:2501.13338v1 [cs.RO].

## Supplementary Material

Our complete source code for this project, including the perception pipeline enhancements and the ACSG extension experiments, is available on GitHub:

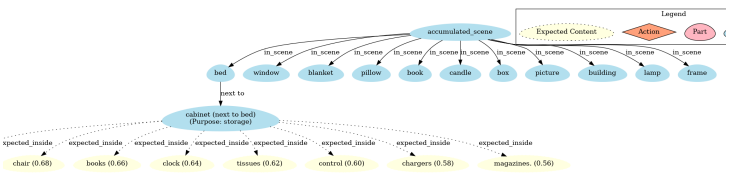
- `https://github.com/Bykho/DLRMRoboEXP/tree/main`

## Supplementary Images

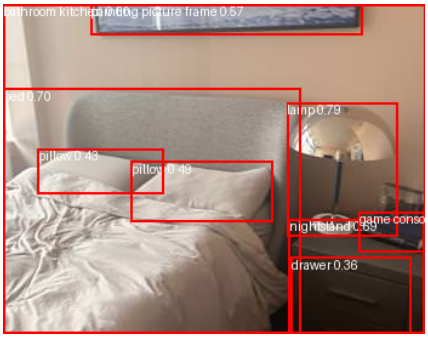



(a) ACSG w/ expected nodes (Frame 1)


(b) GroundingDino Bounding Boxes (Frame 1)




(c) ACSG w/ expected nodes (Frame 2)




(d) GroundingDino Bounding Boxes (Frame 2)

Figure 2: Expected Graph Nodes in the Real World