A SCENE LANGUAGE MODEL FOR OPEN-VOCABULARY SCENE MAPPING

Anonymous authors
Paper under double-blind review

ABSTRACT

Open-vocabulary 3D scene mapping aims to build a persistent representation of the objects in an environment. Existing systems typically rely on engineered mapping pipelines to associate observations, merge information across views, and maintain a consistent scene representation over time. Many additionally store feature-rich object representations, such as embeddings or image crops, increasing the size and complexity of the persistent memory. We introduce SceneLM, a **Scene**–**L**anguage **M**odel that directly maintains a textual scene map. The full scene is represented as a structured text list of objects, which serves as the model's only persistent memory. For each input image, the model reads the current scene state and updates the map by adding, editing, and removing objects. To learn this behavior, we introduce supervision tasks for iterative scene map maintenance together with an automatic annotation pipeline that generates training data from images without human labels. We evaluate SceneLM on both a language-grounded retrieval benchmark and a localization benchmark. Across both benchmarks, the model produces a scene map that achieves competitive performance with complete mapping systems built from dedicated perception and geometric modules while producing a scene representation that is 6–12$\times$ more compact. We further show that SceneLM can be run online on an edge device through experiments on a quadruped. These results show that a persistent open-vocabulary 3D scene map can be maintained directly by a single vision–language model using only a lightweight text representation. Training and inference code is available on the Project page.

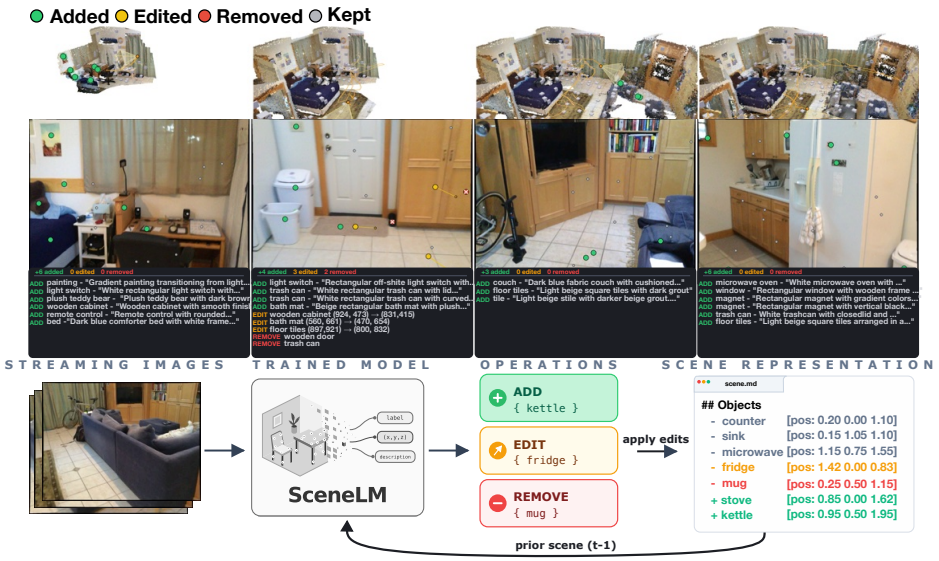


Figure 1: Overview of SceneLM. Given an RGB observation and the current textual scene map, SceneLM predicts `ADD`, `EDIT`, and `REMOVE` operations that update the persistent scene map. The colored point clouds in the top row are for illustration and are obtained by aggregating the posed RGB-D observations. The four example observations in the middle row illustrate how the textual scene map is incrementally updated as objects are added, edited, and removed.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

A robot must maintain a persistent representation of its environment to remember which objects have been observed, where they are located, and how they relate across observations. In open-vocabulary settings, language provides a natural interface for referring to and querying objects in this memory. Such persistent scene understanding is important for navigation, manipulation, and long-horizon embodied interaction. Current open-vocabulary mapping systems typically construct this memory using explicit pipelines that project detections into 3D, associate observations across views, and maintain object tracks or scene graphs (He et al., 2026; Gorlo et al., 2025). Many additionally store visual embeddings, image crops, or segmentation masks, resulting in increasingly complex and memory-intensive scene representations. Rather than extending these pipelines with further specialized components, we ask whether scene-map maintenance itself can be learned by a single vision–language model using only a lightweight textual representation as persistent memory.

We investigate this question with SceneLM, an autoregressive vision–language model that maintains a structured text list of objects, locations, and short descriptions. We call this formulation a *Scene Language Model*: a vision–language model that maintains and updates a persistent scene state through language. Given a posed RGB-D observation and the current scene map, SceneLM predicts `ADD`, `EDIT`, and `REMOVE` operations that update this state (Fig. 1). The model therefore learns operations normally handled by an explicit mapping backend, including discovering objects, associating observations across views, correcting existing entries, and removing false or duplicate objects. No visual embeddings, image crops, or feature banks are retained between observations.

Learning these behaviors requires supervision for how a scene map changes as new observations arrive. We introduce training tasks for object insertion, association, correction, and removal, together with OVAL (Open-Vocabulary Automatic Labeling), an automatic annotation pipeline that generates object locations, noun phrases, and short descriptions from images and filters low-quality annotations. This allows map-maintenance supervision to be generated from individual images rather than manually annotated 3D scene maps, while the trained model is applied iteratively to streaming observations.

We evaluate SceneLM on complementary open-vocabulary mapping benchmarks for language-grounded retrieval (He et al., 2026) and 3D object localization for navigation (Zhang et al., 2024). Across both benchmarks, SceneLM matches or exceeds several complete mapping systems built from dedicated perception and geometric modules, while using a 6–12$\times$ smaller scene representation. In summary, our contributions are:

1. We introduce SceneLM, a Scene Language Model maintaining a persistent open-vocabulary scene map through `ADD`, `EDIT`, and `REMOVE` operations over a structured textual state.
2. We introduce map-maintenance supervision tasks and the OVAL automatic annotation pipeline, enabling these behaviors to be learned from automatically labeled individual images rather than manually annotated 3D scene maps.
3. We show that the resulting text-based scene memory achieves competitive retrieval and localization performance while being more compact than feature-rich mapping representations.

2 RELATED WORK

Open-vocabulary 3D scene mapping typically combines learned perception with an explicit geometric backend that maintains a persistent scene representation. ConceptFusion (Jatavallabhula et al., 2023) fuses dense visual features into a point cloud, while ConceptGraphs (Gu et al., 2024) constructs object-centric scene graphs through cross-view geometric association. Later work adds hierarchical representations (Werby et al., 2024), task-aware abstractions (Maggio et al., 2024), and long-term temporal consistency (Gorlo et al., 2025); TagMap (Zhang et al., 2024), OpenScene (Peng et al., 2023), FARM (He et al., 2026), and BBQ (Linok et al., 2025) follow the same general pattern of open-vocabulary perception coupled with an engineered mapping backend. In contrast, SceneLM maintains a structured textual scene state directly with a single vision–language model. Our aim is to replace specialized association and map-maintenance components with a learned process that jointly handles perception, object association, and scene-state updates.

Vision–language models for 3D reasoning such as 3D-LLM (Hong et al., 2023), LLaVA-3D (Zhu et al., 2024), and Video-3D LLM (Zheng et al., 2025) combine visual and geometric information for tasks including grounding, question answering, and instruction following. These methods generally assume that a scene representation has already been constructed, whereas SceneLM incrementally builds and maintains that representation itself.

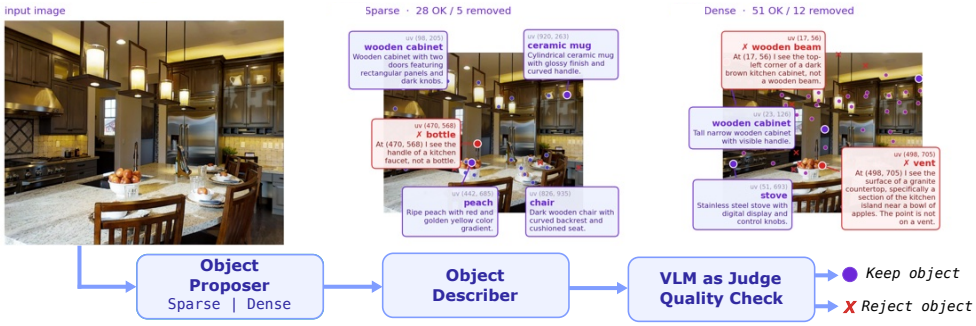


Figure 2: **The OVAL automatic labeling pipeline.** Given an input image, OVAL generates object proposals using a sparse open-vocabulary detector or dense promptable segmentor, then assigns each object a 2D anchor point, noun phrase, and short description. Dots indicate object locations and highlighted examples show their annotations. A VLM verifies each annotation for visual and semantic consistency; rejected objects are marked with a red × and accompanied by the rejection reason. Accepted and rejected annotations are retained to construct training supervision.

Automatic supervision for scene understanding uses foundation models to generate spatial-language supervision without manual 3D annotation. SpatialVLM (Chen et al., 2024), 3D-LLM (Hong et al., 2023), and ReMEmbR (Anwar et al., 2025) construct such supervision from images, videos, or reconstructed environments. Our annotation pipeline follows this direction but targets scene-map maintenance rather than question answering or scene description, producing supervision for object insertion, cross-view association, correction, and removal.

3 The OVAL Data Pipeline

Manual 3D scene-map annotation is costly and hard to scale. We therefore introduce OVAL (**O**pen-**V**ocabulary **A**utomatic **L**abeling), which uses foundation models to generate object-level pseudo-labels from images without human annotation. Each label comprises a 2D point, noun phrase, and short description. Figure 2 shows the pipeline. Next, we describe its stages and assess label quality.

3.1 Labeling Pipeline

(1) Object proposals: We use either a *sparse* open-vocabulary detector or a *dense* promptable segmentor to generate object proposals, denoted OVAL-Sparse and OVAL-Dense. Dense iteratively samples query points in previously uncovered image regions, generates masks, and repeats until a fixed proposal budget is reached, yielding substantially higher object coverage. For each proposal, we retain the most interior mask pixel as a 2D anchor point; masks are used only within the labeling pipeline.

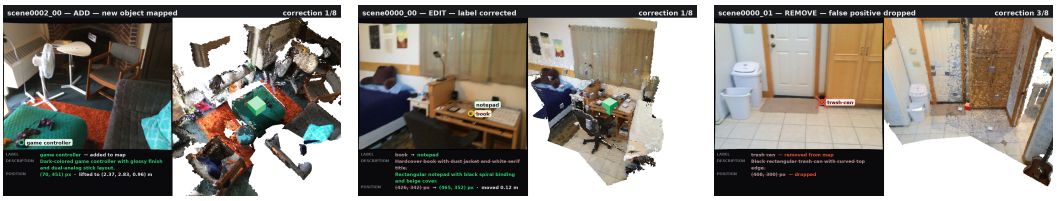
(2) Object descriptions: A mask-conditioned captioning model generates a long-form description for each proposal, which is condensed into a noun phrase and short description. Each annotation is therefore represented as a {2D point, label, short description} triplet.

(3) Quality filtering: A VLM evaluates each annotation for relevance and consistency with the visual evidence, removing fine-grained clutter and incorrect labels or descriptions. Rejected annotations are retained as structured negatives and later used as `REMOVE` training candidates.

(4) Auxiliary training data: To supervise `EDIT` and `REMOVE`, we construct controlled corruption pools from the generated annotations. These include (i) visually similar hard negatives with different semantics, encouraging fine-grained discrimination; (ii) synonym-based variants, exposing the model to alternative valid descriptions of the same object; and (iii) random distractors, creating clearly incorrect object-label pairings. Together, these corruptions provide structured candidates for learning when an existing map entry should be corrected, preserved, or removed.

3.2 Implementation Details

In our experiments, OVAL-Sparse uses YOLOE (Wang et al., 2025) for object proposals, while OVAL-Dense uses SAM (Kirillov et al., 2023). DescribeAnything (Lian et al., 2025) generates mask-conditioned long-form descriptions, which Qwen3 (Qwen Team, 2025) compresses into a noun phrase and short description. For quality filtering, Qwen3-VL-30B evaluates each object against the image evidence and produces an accept/reject decision together with a textual justification. Further implementation details and comparison against human judgments are provided in Sec. B.3.



(a) ADD (b) EDIT (c) REMOVE

Figure 3: **Qualitative map-maintenance examples on a held-out test scene. (a) ADD:** SceneLM recognizes a game controller and adds it with an open-vocabulary label and description. **(b) EDIT:** a closer view refines an existing entry from *book* to *notepad*. **(c) REMOVE:** additional visual evidence reveals a false-positive trash can, which is removed from the map. Colored point clouds are shown only to provide 3D context and are obtained by aggregating the posed RGB-D observations.

3.3 THE DATASET

We apply OVAL with minimal adaptation to 1.3M images spanning indoor navigation and scene understanding (NaVILA, HM3D, ScanNet, SUN RGB-D, and Matterport3D) (Cheng et al., 2024; Ramakrishnan et al., 2021; Dai et al., 2017; Song et al., 2015; Chang et al., 2017), driving (nuScenes and nuImages) (Caesar et al., 2020), and web imagery (COCO) (Lin et al., 2014). Dataset details are provided in Sec. B.1.

OVAL-Sparse and OVAL-Dense produce 10.7M and 41.6M annotations, averaging 8.2 and 32 objects per image, respectively; 81% and 65% pass the VLM quality filter. Each comprises a 2D point, noun phrase, and short description. Unless stated otherwise, we train SceneLM with OVAL-Sparse.

On 100 manually assessed images, 85% of Sparse and 69% of Dense annotations have all three fields correct and visually supported. The filter agrees with human judgments on approximately 80% of annotations (Cohen's $\kappa = 0.47$; balanced accuracy 0.74), and 84% of retained annotations are human-verified as correct. We sample both accepted and rejected annotations from this curated pool to construct the map-maintenance objectives. Further analysis is provided in Sec. B.3.

4 SCENELM: MODEL AND TRAINING

We describe the model architecture and map update mechanism in Sec. 4.1, the supervision task design in Sec. 4.2, and present the training setup in Sec. 4.3.

4.1 SCENE-MAP MAINTENANCE

SceneLM represents the persistent scene map as object records $M_t = \{o_i\}_{i=1}^{N}$, each containing a category label, short description, and 3D world position. Map maintenance is modeled autoregressively as $M_t = f_\theta(M_{t-1}, I_t)$, where I_t is the incoming posed RGB-D observation.

At each timestep, we select the objects in M_{t-1} visible in I_t using camera pose and depth-based visibility filtering, serialize them as structured text, and provide them together with the RGB image to the model. SceneLM predicts structured ADD, EDIT, and REMOVE operations that insert new objects, update existing labels, descriptions, or positions, and remove incorrect or redundant entries. These operations jointly perform cross-view association and scene-map reconciliation within a single forward pass. Predicted objects are localized by a 2D anchor in normalized $[0, 1000]$ image coordinates, which is unprojected to 3D using depth and camera pose. After applying the predicted operations, the updated map M_t becomes the persistent state for the next timestep. No external tracking, optimization, or geometric data association is used.

4.2 TRAINING TASKS

The goal of training is to teach the model to maintain a scene map by predicting the updates required after each new observation, see Fig. 3. OVAL provides object annotations for individual images, but not the scene-map transitions required for this task. We therefore algorithmically construct training examples that mimic required map updates from the pseudo-labels. Each example consists of an input scene-map context, an image, and the corresponding set of update operations. Training is organized into two complementary task families. ADD tasks teach the model to extend the scene map by inserting new objects, while EDIT-REMOVE tasks teach it to correct an existing map through sparse update operations.

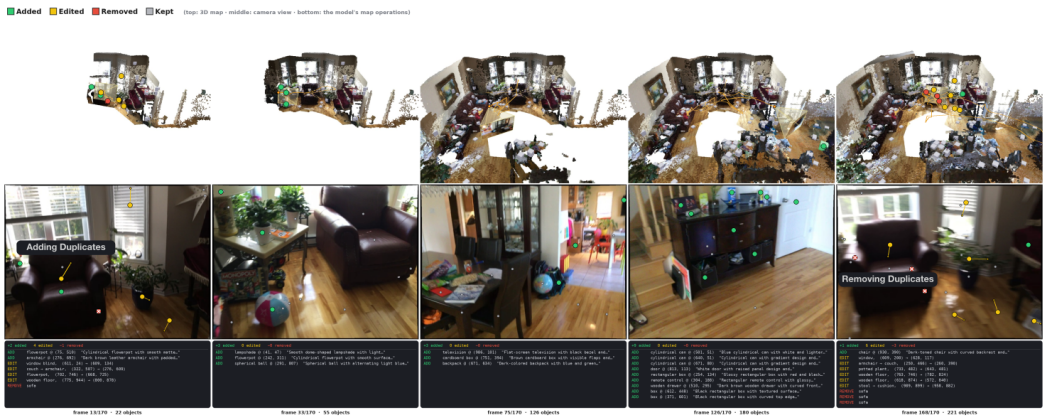


Figure 4: **Scene-map evolution over a complete trajectory.** Each column shows accumulated 3D context (top), the RGB observation (middle), and predicted ADD, EDIT, and REMOVE operations (bottom). SceneLM maps objects with open-vocabulary labels and descriptions, using later observations to correct earlier errors. Here, duplicate objects are added and later removed.

ADD tasks supervise the addition of new objects into the scene map. We instantiate this supervision task through different sampling strategies that control the amount of prior information available: In the *no-context* setting, the model predicts all visible objects from an empty context and learns initial map construction. In the *partial-context* setting, a subset of objects is provided as an existing map, and the model must add only the missing objects without duplicating or reintroducing known ones. Partial context is generated either through spatial cropping or random subsampling of objects, encouraging robustness to incomplete map states and cross-view association. The model receives an image together with a partial or empty scene-map context and is trained to predict all remaining objects that should be inserted to get a complete map.

EDIT-REMOVE tasks supervise correction of an existing scene map. The model receives a corrupted scene map and its image, then predicts a sparse set of EDIT and REMOVE operations required to restore the correct map. Objects omitted from the output remain unchanged, yielding a sparse update script that may be empty. Corruptions include incorrect labels, perturbed object locations, duplicated entries, and hallucinated objects, and may appear individually or in combination. A key design choice is how to generate corruption signals. Sampling a single corruption rate per image leads to strong image-level correlations, encouraging degenerate policies that either modify most objects or none at all. Instead, we corrupt each object independently, removing this global coupling and forcing object-wise decisions grounded in visual evidence. This makes partial corrections the dominant training signal, aligning the learning objective with iterative scene-map maintenance.

4.3 Training Details

We instantiate SceneLM with 2B and 4B Qwen3-VL variants (Qwen Team, 2025) and jointly train the vision encoder, projector, and language model for three epochs on eight NVIDIA A100 80GB GPUs. We use AdamW with a cosine schedule, peak learning rate 2×10^{-5}, effective batch size 128, and maximum sequence length 4096. See Tab. 10 for details.

5 Experiments

Our experiments address four questions. First, *can a vision–language model learn the basic operations required to maintain a scene map?* In Sec. 5.2, we evaluate ADD, EDIT, and REMOVE in controlled single-frame experiments against off-the-shelf VLMs. Second, *can a single VLM maintain a competitive persistent open-vocabulary scene map using only text as memory?* In Secs. 5.3 and 5.4, we evaluate its maps on two complementary benchmarks: language-grounded object retrieval with FARM (He et al., 2026) and 3D localization for navigation with TagMap (Zhang et al., 2024). Qualitative examples are shown in Figs. 4 and 5. Third, *which components enable successful persistent mapping?* Section 5.5 presents controlled ablations of proposal coverage, short object descriptions, quality filtering, and explicit EDIT-REMOVE training. Finally, *what scene-memory and inference-time trade-offs arise from replacing a modular mapping backend with a single VLM?* In Secs. 5.6 and 5.7, we compare memory and inference cost with existing systems and test online operation on embedded hardware through deployment on a quadruped.

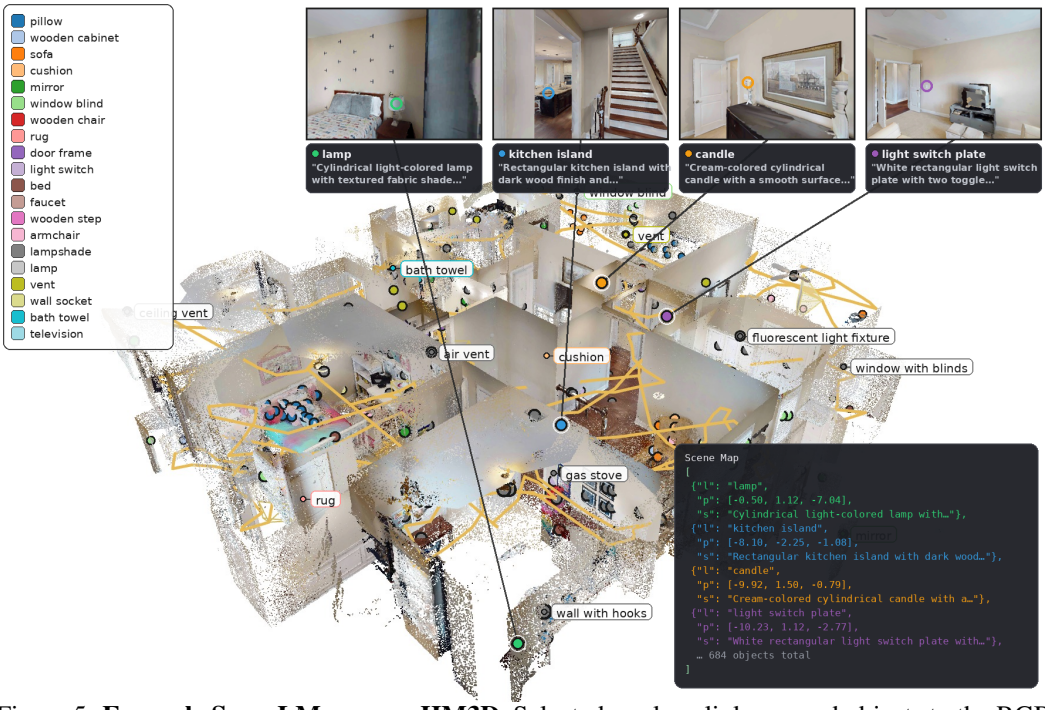


Figure 5: **Example SceneLM map on HM3D.** Selected markers link mapped objects to the RGB observations where they were identified. The inset shows the text-only memory, storing each object as a 3D position, noun phrase, and short description.

5.1 SETUP

Training Tasks: We evaluate whether the model has learned the three core map-maintenance primitives: `ADD`, `EDIT`, and `REMOVE`. For `ADD`, the model receives an image with an empty scene map and predicts all visible objects. Because the dataset vocabularies are incomplete, we report recall based on whether the predicted anchor lies inside the ground-truth object (R_{pos}), and whether both anchor and label match (R_{obj}). These metrics provide a lower bound since spelling variants and synonyms are not resolved. For `EDIT`/`REMOVE`, the model receives a corrupted scene map and image and must keep, edit, or remove each object. We evaluate whether the model identifies objects requiring intervention, selects the correct action, and improves corrupted labels or positions. Full metric definitions and baseline details are provided in Sec. D.

Retrieval: We follow the FARM (He et al., 2026) benchmark and baseline setup, using the 30 largest ReferIt3D (Achlioptas et al., 2020) scenes on ScanNet (Dai et al., 2017) and IRef-VLA (Zhang et al., 2025) on HM3D (Ramakrishnan et al., 2021), and use FARM's implementations and evaluation setup for the mapping baselines. Given a language query, we use the FARM retrieval system, without its visual embedding, to rank objects in the scene map produced by SceneLM. We report top-1 retrieval accuracy (Acc@1), Recall@5, Recall@10, and mean reciprocal rank (MRR), which rewards placing the first correct retrieval high in the ranking. Because SceneLM represents each object by a 3D point, label, and short description rather than an extent, we replace FARM's IoU-based matching criterion ($\tau = 0.1$) with an extent-free evaluation in which a retrieval is correct if the predicted point lies inside the ground-truth 3D bounding box. We compare against BBQ (Linok et al., 2025), DAAAM (Gorlo et al., 2025), FARM (He et al., 2026), and RynnBrain (Dang et al., 2026), a frame-based 30B VLM that localizes queried objects directly from images without building a persistent map, under this point-in-box criterion. Results under the original FARM protocol are provided in Sec. E.2.

Localization: We follow the TagMap (Zhang et al., 2024) benchmark on Matterport3D (Chang et al., 2017). Given a category query, the system retrieves the best matching object and selects the localization node closest to the object. Following prior work, we report proposal-to-entity precision and entity-to-proposal recall at navigation thresholds $\tau \in \{0.1, 0.5, 1.0, 2.0\}$ m, macro-averaged over the 21 Habitat ObjectNav categories. Note that for ScanNet, HM3D, and Matterport3D, all evaluation scenes are excluded from both OVAL pseudo-label generation and SceneLM training.

	ADD								EDIT-REMOVE							
	COCO		LVIS		nuImages		Average		Decision				Action		Quality	
Model	R_{pos} ↑	R_{obj} ↑	R_{pos} ↑	R_{obj} ↑	R_{pos} ↑	R_{obj} ↑	Loop↓	Obj./Img.	F1↑	P↑	R↑	FT↓	Edit↑	Rem.↑	Label↑	Pos.↑
Ours and ablations																
Sparse w Desc.	65	41	42	21	37	27	0	8.7	65	57	77	15	70	72	87	77
Sparse w/o Desc.	65	41	42	21	37	27	0	9.0	61	48	87	25	72	83	80	78
Dense w Desc.	69	43	53	28	41	26	0	26.9	66	64	69	10	71	57	87	78
ADD-only	65	40	42	20	38	25	0	8.5	–	–	–	–	–	–	–	–
Random Order	64	41	40	20	36	26	1	11.0	63	51	84	22	68	78	82	78
Quantized-40	67	42	43	21	38	27	0	8.6	62	51	84	22	68	81	81	78
Off-the-shelf VLMs																
Qwen3-VL-2B	29	21	15	7	15	12	35	19.2	27	19	49	52	50	0	19	3
Qwen3-VL-4B	51	39	30	15	29	24	40	23.7	29	18	82	91	85	0	21	2
Qwen3-VL-8B	51	38	30	16	31	25	20	31.2	28	18	76	86	81	0	27	4
RynnBrain-30B	37	30	18	8	17	14	20	7.1	31	19	80	83	82	0	17	7

Table 1: Single-frame evaluation of SceneLM-4b. Finetuning eliminates looping and substantially improves scene-map correction, reducing false touches from 83–91% to 10–25% while enabling both EDIT and REMOVE. Dense proposals improve recall at the cost of more objects per image. All metrics are percentages except *Obj./Img.*; arrows indicate whether higher or lower is better.

5.2 Training Tasks Results

ADD: Results are shown on the left of Tab. 1. SceneLM improves single-frame object prediction and removes the looping behavior observed in off-the-shelf VLMs. For instance, Qwen3-VL gets stuck in loops in up to 40% of images for the prompt maximizing baseline recall. Adding short descriptions or jointly training EDIT-REMOVE leaves ADD recall largely unchanged, while coarse coordinate quantization has little effect on SceneLM. Random object ordering slightly reduces recall and can reintroduce looping. Dense proposals increase recall by producing substantially more object candidates, whereas Sparse yields a more compact scene representation.

EDIT-REMOVE: Results are shown on the right of Tab. 1. Off-the-shelf VLMs perform poorly at the decision layer: they modify too many correct objects, yielding false-touch rates of 83–91%, and achieve decision F1 scores around 30%. They also almost never perform REMOVE correctly, with removal recall approximately zero. After finetuning, SceneLM improves decision F1 to above 60%, false touches fall to 10–25%, and both EDIT and REMOVE recall increase substantially. Among correctly action-typed edits, predicted labels are corrected in 81–87% of cases and predicted positions improve over the corrupted locations in 77–78%. These results show that the proposed finetuning changes both *when* the model intervenes and *which* corrective action it selects.

Taken together, these controlled experiments show that a vision–language model can learn the three basic operations required for scene-map maintenance: adding new objects and deciding when existing entries should be edited or removed.

5.3 Retrieval Results

We first evaluate whether the text-only persistent scene maps produced by SceneLM support language-grounded object retrieval. SceneLM improves Recall@5, Recall@10, and mean reciprocal rank over FARM on both ScanNet and HM3D, while achieving similar or better top-1 retrieval accuracy. Adding short object descriptions improves most retrieval metrics, suggesting that object-level semantic attributes provide useful information beyond the noun phrase alone. The proposal strategy introduces a clear trade-off: Dense proposals increase object coverage but also redundancy, while Sparse proposals yield stronger overall retrieval on these benchmarks. Removing EDIT-REMOVE training causes the largest degradation, particularly on ScanNet, indicating that scene-state reconciliation is important for maintaining a consistent persistent map. Removing quality filtering produces smaller but still measurable drops. Overall, SceneLM matches or exceeds prior mapping systems on top-1 accuracy and improves the ranking-based retrieval metrics, while storing only a text-based persistent scene representation.

5.4 Localization Results

We next evaluate whether the resulting maps support category-based object localization for navigation. Tab. 3 reports Matterport3D localization performance under the TagMap benchmark. We report precision (P) and recall (R) at navigation distance thresholds of 0.1, 0.5, 1.0, and 2.0 m. SceneLM improves precision over prior mapping systems at all thresholds, with the largest gains at 0.1 and

Method	Venue	ScanNet				HM3D			
		Acc@1 ↑	R@5 ↑	R@10 ↑	MRR ↑	Acc@1 ↑	R@5 ↑	R@10 ↑	MRR ↑
BBQ	ICRA'25	22.6	34.3	34.9	27.6	4.2	8.6	10.1	6.1
RynnBrain-30B		38.9	—	—	—	4.3	—	—	—
DAAAM	CVPR'26	37.1	—	—	—	5.4	—	—	—
FARM		32.0	63.0	75.0	45.7	6.3	17.2	25.1	11.7
Qwen3-VL-2B †		12.4	20.9	23.1	16.3	2.4	5.6	6.8	3.9
Qwen3-VL-4B †		25.6	44.7	48.5	34.1	4.0	10.1	13.6	6.8
SceneLM-2B		38.5	70.4	78.1	52.2	6.7	18.2	26.5	12.3
SceneLM-4B		39.3	71.3	79.2	52.6	6.6	18.8	26.5	12.5
Ablations									
SceneLM-2B	w/o Desc.	36.7	70.3	78.3	51.0	6.0	17.3	24.8	11.5
	No Qual.	36.8	69.9	79.4	50.9	5.8	18.5	27.1	12.0
	ADD-only	29.7	54.6	61.3	40.7	6.5	17.6	24.2	11.8
	Dense	34.5	62.4	71.2	46.5	5.5	14.0	20.0	9.8

Table 2: Scene-graph retrieval results and ablations. SceneLM achieves strong point-on-box results on both ScanNet and HM3D. † ADD only with ad-hoc rules reducing failure cases and looping behavior.

Method	Venue	P@0.1 ↑	P@0.5 ↑	P@1.0 ↑	P@2.0 ↑	R@0.1 ↑	R@0.5 ↑	R@1.0 ↑	R@2.0 ↑
OpenScene	CVPR'23	31.0	36.0	41.0	46.0	12.0	29.0	61.0	88.0
OpenMask3D	NeurIPS'23	29.0	35.0	41.0	48.0	5.0	19.0	41.0	60.0
TagMap	CoRL'24	28.0	39.0	46.0	53.0	13.0	42.0	65.0	82.0
SceneLM-2B		41.4	48.8	54.4	60.8	28.0	63.9	71.6	78.4
SceneLM-4B		43.2	50.0	55.5	62.1	28.7	66.4	76.6	81.7

Table 3: Matterport3D object localization under the TagMap benchmark. P and R denote precision and recall at the indicated navigation distance thresholds in meters. SceneLM improves precision over prior mapping systems across all thresholds.

0.5 m. The proposal strategy exposes a precision–recall trade-off: Dense proposals increase recall through higher object coverage, while Sparse proposals yield the highest precision. Removing EDIT-REMOVE training reduces both precision and recall across nearly all thresholds, while removing quality filtering causes smaller degradations. Additional ablations are provided in Sec. E.1. Taken together with the retrieval results in Sec. 5.3, these complementary evaluations show that a single vision–language model can maintain a competitive persistent open-vocabulary scene map using only text-based memory, supporting both language-grounded object retrieval and category-based localization for navigation.

5.5 Components Enabling Scene Mapping

We next synthesize the ablations from the training-task, retrieval, and localization evaluations in Secs. 5.2 to 5.4 to identify which design choices enable persistent scene mapping.

Coverage over precision: The proposal source sets the system's coverage–precision operating point. Dense proposals increase object coverage and localization recall, whereas Sparse proposals yield more compact maps and higher precision. For persistent mapping, greater coverage is particularly valuable when missed objects are not observed again, whereas spurious entries can potentially be corrected through subsequent reconciliation.

Textual object representations: Short descriptions have little effect on single-frame localization but improve retrieval, indicating that they contribute primarily to object identity rather than detection. The additional semantic detail helps distinguish objects with similar noun phrases during language-based retrieval. At the same time, a text-only representation necessarily discards fine-grained visual information, which can introduce ambiguity when multiple instances share similar labels or descriptions. The improvement from adding short descriptions suggests that richer textual attributes partially mitigate this ambiguity, but text remains a lossy representation and may not be optimal for all downstream tasks.

Learning EDIT and REMOVE: Removing EDIT-REMOVE training leaves single-frame ADD performance largely unchanged but degrades both retrieval and localization. These results show that maintaining a scene map as new observations arrive requires more than detecting and adding objects. The model must also learn when to edit existing entries and remove stale or duplicate ones.

Quality filtering and serialization choices: Object ordering, coordinate quantization, and quality filtering have smaller effects than proposal coverage, semantic descriptions, and EDIT-REMOVE training. Removing quality filtering causes smaller but measurable degradations, indicating that it contributes to mapping performance without being its primary driver.

Metric	SceneLM		FARM	
	ScanNet	HM3D	ScanNet	HM3D
Scene memory (MiB/scene) ↓	**3.5**	**10.8**	23	125
Mapping time (ms/frame) ↓	965	843	**275**	**258**

Table 4: **Memory–latency trade-off on A100.** SceneLM-4B uses 6–12× less scene memory than FARM, but roughly 3× longer per-frame mapping time.

Precision	RTX 5090		Jetson Thor	
	4B	2B	4B	2B
BF16	1.36	2.59	0.21	0.53
FP8	1.91	3.09	0.33	0.83
NVFP4	**2.36**	**3.54**	**0.51**	**0.86**

Table 5: **Mapping throughput under quantization.** Lower-precision inference increases throughput on both systems.

Overall, the proposal source and EDIT-REMOVE training have the broadest effects on mapping performance, while short descriptions mainly help retrieval and quality filtering mainly improves precision. Object ordering and coordinate quantization have only minor effects. Additional ablations are provided in Sec. E.1.

5.6 Efficiency

We evaluate scene-memory size and per-frame mapping latency on ScanNet and HM3D using an A100. As shown in Tab. 4, SceneLM uses approximately 6–12× less scene memory than FARM: 3.5 MiB vs. 23 MiB on ScanNet and 10.8 MiB vs. 125 MiB on HM3D. This reduction comes from storing only object-level textual state rather than feature-rich multi-view representations. The trade-off is higher latency. SceneLM requires approximately 900 ms per frame, while FARM is about 3× faster. Quantization increases throughput to 3.5 frames/s on an RTX 5090 (Tab. 5). While VLM inference remains fairly slow, improving model efficiency is largely orthogonal to our mapping formulation, and continued advances in VLM inference can be directly inherited by SceneLM.

5.7 Deployment on a quadruped

We deploy SceneLM on a Boston Dynamics Spot quadruped (Boston Dynamics, 2020) equipped with an NVIDIA Jetson and an Odin1 sensor (Manifold Tech Ltd., 2026) providing posed RGB-D observations. We evaluate three previously unseen environments: two indoor scenes and one outdoor scene. Using SceneLM-2B in NVFP4, we process a frame only after the robot has moved at least $0.5\,\mathrm{m}$ from the previous processed pose. This keeps inference in a streaming regime without accumulating a frame backlog on the onboard hardware. Across the three environments, 35 of 44 manually selected target objects are represented in the final maps. These experiments show that the learned mapping behavior transfers beyond benchmark data and remains usable under real sensor noise, viewpoint changes, and scene clutter. Taken together with the efficiency results in Sec. 5.6, they characterize the system-level trade-off: SceneLM substantially reduces persistent scene-memory requirements through its text-only representation, whereas autoregressive VLM inference incurs high per-frame latency. Nevertheless, onboard operation remains feasible when inference is coupled with motion-based frame selection rather than processing every incoming frame. An example map is shown in Fig. 13, and additional implementation details and qualitative results are provided in Sec. G.

6 Conclusions

We introduced SceneLM, a scene–language model that maintains a persistent 3D scene map as structured text from posed RGB-D observations. The scene memory consists only of object locations, labels, and short descriptions, without persistent visual embeddings or feature maps. Across retrieval and localization benchmarks, SceneLM matches or exceeds several modular mapping systems while using a substantially more compact scene representation. Our ablations show that explicit EDIT and REMOVE supervision is important for maintaining scene consistency over time, and OVAL provides the automatic annotations needed to train these behaviors. The main limitation is inference cost, since each processed frame requires a full auto-regressive forward pass. Nevertheless, quantized SceneLM-2B can operate online on a quadruped using motion-based frame selection. These results support a broader direction in which increasingly capable vision–language models absorb scene-map maintenance operations that are currently implemented by specialized mapping components. Future work should improve inference efficiency and extend persistent scene modeling to longer-term and dynamic environments. More broadly, persistent scene-state maintenance may become a useful capability of future vision–language models beyond robotic mapping.

AI USE STATEMENT

Pretrained vision–language models are used as components of the OVAL pseudo-annotation pipeline. They generate textual object descriptions and pseudo-annotations from real images and perform automated quality filtering. These outputs constitute machine-generated training data used to construct supervision for SceneLM, but the input images themselves were not synthetically generated.

We also used generative AI assistants, including coding agents, to assist with implementing training and experimental code and with parsing datasets. In addition, we used these tools to edit author-written text for clarity and readability and to assist in creating and revising scientific figures and visualizations.

We did not use generative AI tools to develop the scientific methodology or conceptual framework, design experiments, formulate mathematical claims, propose or refine hypotheses, translate text, perform qualitative or thematic data analysis, or interpret experimental results. Proof-related uses are not applicable to this empirical work.

All AI-assisted manuscript text, figures, code, and data-processing outputs were reviewed by the authors. AI-assisted code and dataset parsing were inspected and tested before being used to produce the reported results, and visualizations were checked against their underlying data. We take responsibility for the final content of this work, including all text, claims, code, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The OVAL pseudo-annotation pipeline, its constituent models, and the resulting training data are described in Sec. 3. The complete dataset mixture and the human evaluation of pseudo-label quality are provided in the appendix, and we are only using open-source datasets. The SceneLM architecture, supervision tasks, and training procedure are described in Sec. 4, with the full training configuration provided in Sec. C. The evaluation protocols and additional experimental details are provided in Sec. 5.1 and the corresponding appendix sections.

We, furthermore, release code for pseudo-annotation, dataset preparation, training, inference, and evaluation, together with environment specifications, configuration files, prompts, random seeds, and dataset splits. We additionally provide the model checkpoints used to obtain the reported results through Hugging Face. Because pseudo-annotation and model training contain stochastic components, independent reruns may produce slightly different outputs; the released configurations and checkpoint correspond to the results reported in this paper.

REFERENCES

Panos Achlioptas, Ahmed Abdelreheem, Fei Xia, Mohamed Elhoseiny, and Leonidas J. Guibas. ReferIt3D: Neural listeners for fine-grained 3d object identification in real-world scenes. In *16th European Conference on Computer Vision (ECCV)*, 2020.

Abrar Anwar, John Welsh, Joydeep Biswas, Soha Pouya, and Yan Chang. ReMEmbR: Building and reasoning over long-horizon spatio-temporal memory for robot navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

Boston Dynamics. Spot specifications. `https://bostondynamics.com/wp-content/uploads/2020/05/spot-spec-sheet.pdf`, 2020. Accessed 2026-07-27.

Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. In *International Conference on 3D Vision (3DV)*, 2017.

Boyuan Chen, Zhuo Xu, Sean Kirmani, Brian Ichter, Danny Driess, Pete Florence, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. SpatialVLM: Endowing vision-language models with spatial reasoning capabilities. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

An-Chieh Cheng, Yandong Ji, Zhaojing Yang, Zaitian Gongye, Xueyan Zou, Jan Kautz, Erdem Bıyık, Hongxu Yin, Sifei Liu, and Xiaolong Wang. NaVILA: Legged robot vision-language-action model for navigation. *arXiv preprint arXiv:2412.04453*, 2024.

Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Ronghao Dang, Jiayan Guo, Bohan Hou, Sicong Leng, Kehan Li, Xin Li, Jiangpin Liu, Yunxuan Mao, Zhikai Wang, Yuqian Yuan, Minghao Zhu, Xiao Lin, Yang Bai, Qian Jiang, Yaxi Zhao, Minghua Zeng, Junlong Gao, Yuming Jiang, Jun Cen, Siteng Huang, Liuyi Wang, Wenqiao Zhang, Chengju Liu, Jianfei Yang, Shijian Lu, and Deli Zhao. Rynnbrain: Open embodied foundation models. *arXiv preprint arXiv:2602.14979v1*, 2026. URL `https://arxiv.org/abs/2602.14979v1`.

Nicolas Gorlo, Lukas Schmid, and Luca Carlone. Describe anything anywhere at any moment. *arXiv preprint arXiv:2512.00565*, 2025.

Qiao Gu, Alihusein Kuwajerwala, Sacha Morin, Krishna Murthy Jatavallabhula, Bipasha Sen, Aditya Agarwal, Corban Rivera, William Paul, Kirsty Ellis, Rama Chellappa, et al. ConceptGraphs: Open-vocabulary 3D scene graphs for perception and planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

Siming He, Leo Huang, Adam Lilja, Fabio Hubel, Jonas Frey, Marco Pavone, S Shankar Sastry, Jitendra Malik, and Claire Tomlin. Farm: Find anything using relational spatial memory. *arXiv preprint arXiv:2606.15476*, 2026.

Yining Hong, Haoyu Zhen, Peihao Chen, Shuhong Zheng, Yilun Du, Zhenfang Chen, and Chuang Gan. 3D-LLM: Injecting the 3D world into large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Krishna Murthy Jatavallabhula, Alihusein Kuwajerwala, Qiao Gu, Mohd Omama, Tao Chen, Alaa Maalouf, Shuang Li, Ganesh Iyer, Soroush Saryazdi, Nikhil Keetha, et al. ConceptFusion: Open-set multimodal 3D mapping. In *Robotics: Science and Systems (RSS)*, 2023.

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. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *ACM Symposium on Operating Systems Principles (SOSP)*, 2023.

Long Lian, Yifan Ding, Yunhao Ge, Hongxu Yin, Greg Heinrich, Daniel Smith, et al. Describe anything: Detailed localized image and video captioning. *arXiv preprint arXiv:2504.16072*, 2025.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2014.

Sergey Linok, Tatiana Zemskova, Svetlana Ladanova, Roman Titkov, Dmitry Yudin, Maxim Monastyrny, and Aleksei Valenkov. Beyond bare queries: Open-vocabulary object grounding with 3d scene graph. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13582–13589, 2025. doi: 10.1109/ICRA55743.2025.11128059.

Dominic Maggio, Yun Chang, Nathan Hughes, Matthew Trang, Dan Griffith, Carlyn Dougherty, Eric Cristofalo, Lukas Schmid, and Luca Carlone. Clio: Real-time task-driven open-set 3D scene graphs. *IEEE Robotics and Automation Letters (RA-L)*, 2024.

Manifold Tech Ltd. Odin1 spatial memory module. https://manifoldtechltd.github.io/wiki/odin_series/odin1/, 2026. Accessed 2026-07-31.

Songyou Peng, Kyle Genova, Chiyu Jiang, Andrea Tagliasacchi, Marc Pollefeys, and Thomas Funkhouser. OpenScene: 3D scene understanding with open vocabularies. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Qwen Team. Qwen3-VL. https://github.com/QwenLM/Qwen3-VL, 2025.

Santhosh K. Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X. Chang, Manolis Savva, Yili Zhao, and Dhruv Batra. Habitat-Matterport 3D dataset (HM3D): 1000 large-scale 3D environments for embodied AI. In *Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2021.

Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. SUN RGB-D: A RGB-D scene understanding benchmark suite. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Ao Wang, Lihao Liu, Hui Chen, Zijia Lin, Jungong Han, and Guiguang Ding. YOLOE: Real-time seeing anything. *arXiv preprint arXiv:2503.07465*, 2025.

Abdelrhman Werby, Chenguang Huang, Martin Büchner, Abhinav Valada, and Wolfram Burgard. Hierarchical open-vocabulary 3D scene graphs for language-grounded robot navigation. In *Robotics: Science and Systems (RSS)*, 2024.

Haochen Zhang, Nader Zantout, Pujith Kachana, Ji Zhang, and Wenshan Wang. Iref-vla: A benchmark for interactive referential grounding with imperfect language in 3d scenes. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1677–1683, 2025. doi: 10.1109/ICRA55743.2025.11127464.

Mike Zhang, Kaixian Qu, Vaishakh Patil, Cesar Cadena, and Marco Hutter. Tag map: A text-based map for spatial reasoning and navigation with large language models. In *Conference on Robot Learning (CoRL)*, 2024.

Duo Zheng, Shijia Huang, and Liwei Wang. Video-3D LLM: Learning position-aware video representation for 3D scene understanding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Chenming Zhu, Tai Wang, Wenwei Zhang, Jiangmiao Pang, and Xihui Liu. LLaVA-3D: A simple yet effective pathway to empowering LMMs with 3D-awareness. *arXiv preprint arXiv:2409.18125*, 2024.

A Appendix Content

This appendix provides additional implementation details, experimental results, and qualitative examples that complement the main paper. It contains:

- Details of the training data, including the dataset mixture (Sec. B.1) and an analysis of pseudo-label quality (Sec. B.2).
- The complete training configuration (Sec. C).
- Additional analysis of the per-frame mapping components, including prompt ablations for `ADD` (Sec. D.1) and evaluation metrics for the `EDIT`/`REMOVE` task (Sec. D.2).
- Additional experimental results and ablations for language-guided 3D retrieval (Sec. E).
- Additional qualitative mapping examples (Sec. F).
- Further details on deployment and edge inference (Sec. G).

B TRAINING DATA

This appendix provides additional details on the training data used to train SceneLM. Since the pipeline operates on individual RGB images rather than videos or reconstructed scenes, it can readily incorporate data from a wide range of image sources. We first summarize the datasets used for training in Sec. B.1, followed by an analysis of the pseudo-label quality in Sec. B.2.

B.1 DATASETS

Tab. 6 summarizes the training mixture used throughout the paper. The selected datasets span indoor scans, embodied navigation, autonomous driving, and web imagery, providing broad visual diversity while requiring only single RGB images. Since the pipeline is dataset-agnostic, additional image datasets can be incorporated without modifying the training procedure. Note that we remove the evaluation scenes from HM3D, Matterport3D, and ScanNet from the training data, so that no evaluation scenes or frames entered OVAL or SceneLM training, for a fair comparison.

Dataset	Samples	Good obj.	Rejected obj.	Total obj.
NaVILA Cheng et al. (2024)	433,718	3,146,945	745,264	3,892,209
HM3D Ramakrishnan et al. (2021)	227,248	1,361,945	351,291	1,713,236
COCO 2017 Lin et al. (2014) (incl. unlab.)	174,519	1,640,005	377,571	2,017,576
Matterport3D Chang et al. (2017)	138,280	616,706	132,001	748,707
nuScenes Caesar et al. (2020)	119,783	721,176	128,622	849,798
ScanNet Dai et al. (2017)	95,150	667,574	154,102	821,676
nuImages Caesar et al. (2020)	65,782	413,172	69,779	482,951
ODIN	10,560	91,946	23,388	115,334
SUN RGB-D Song et al. (2015)	2,875	21,039	5,470	26,509
GrandTour (frontier subset)	385	1,314	262	1,576
Total	**1,268,300**	**8,681,822**	**1,987,750**	**10,669,572**

Table 6: **Training mixture, per dataset** (epsilon pack, YOLOE proposals). Samples are RGB frames; objects are per-frame records emitted by the pipeline. "Good" objects (QC-passed) are ADD targets and carry the loss; "rejected" objects supply REMOVE/EDIT supervision. The COCO row folds in the unlabeled split.

B.2 PSEUDO-LABEL QUALITY

The quality check decides *which* proposals to keep; a separate question is how accurate the kept labels are in absolute terms. We answer it with a *golden set*: 100 images (30 COCO, 30 NaVILA, 25 nuScenes, 15 GrandTour) on which a human judged every pseudo-labeled object correct or incorrect at its anchor point. Over $\sim$4,000 judged objects, 71.8% **are correct** (Table 7). Two patterns matter. First, quality tracks the proposal source exactly as the trained map does: Sparse proposals are 84.8% correct against 68.6% for Dense, a $+16$-point precision gap that prefigures the Sparse-precision / Dense-recall split on the cumulative map (Section 5.5). Second, interior anchors are cleaner than center points (71.8% vs. 63.0%), confirming the anchor choice we make. Quality also varies by domain (nuScenes 82%, GrandTour 81%, COCO 72%, NaVILA 68%). The takeaway is that the raw proposals are good but not clean, which is exactly why the quality gate, and the model's learned REMOVE behavior trained on what it rejects, are needed.

Slice	% correct
Overall (interior anchor)	71.8
Sparse proposals	**84.8**
Dense proposals	68.6

Table 7: **Pseudo-label accuracy** on the golden set (100 human-verified images). % of pseudo-labeled objects judged correct, by proposal source.

Judge	MCC	Bal. acc.	Recall-bad
Gemini 3.1 Pro	0.64	0.81	0.74
Grok 4.3	0.48	0.74	0.68
Claude Sonnet 4	0.29	0.66	0.63
Qwen3-VL-30B (used)	0.47	0.74	0.63

Table 8: **Quality-control judge vs. human golden labels**, best configuration per model. MCC and balanced accuracy weight both classes; *recall-bad* is the fraction of human-rejected objects the judge catches, the operative quality-gate metric.

Judge variant	BalAcc	RecBad	Rec	κ	Agree
Base: per-object, annotated, point+label, reason	70.6%	63.9%	77.2%	0.39	73.2%
Query mode: batch (1 call/image)	67.5%	60.6%	74.4%	0.33	70.2%
Image: plain (no point annotation)	70.1%	77.5%	62.8%	0.34	67.2%
Reasoning: none	70.5%	64.7%	76.2%	0.39	72.8%
Reasoning: reason-first	67.0%	42.5%	91.5%	0.38	76.6%
Abstention: allow skip	65.2%	47.1%	83.4%	0.30	74.9%
Zoom-on-object: crop 0.25	68.5%	45.0%	92.0%	0.41	77.8%
Zoom-on-object: crop 0.50	70.8%	52.0%	89.7%	0.45	78.3%
Super-resolution	72.3%	67.1%	77.5%	0.42	74.4%
Pixel grounding	68.9%	50.4%	87.3%	0.40	76.2%

Table 9: Controlled Quality Check prompt ablation on Qwen3-VL-30B: each row flips one design choice off the base configuration.

B.3 Quality Check

The per-object keep/reject decision is made by a VLM *quality check*, and its calibration matters: a stricter gate yields cleaner `ADD` targets but more `EDIT-REMOVE` fuel, and it must run over all 10.7M objects, ruling out per-call API pricing. We benchmark candidate quality checks against the human golden labels (Tab. 8). Frontier APIs are the most accurate (Gemini 3.1 Pro reaches MCC 0.64) but carry a per-call cost we cannot pay at this scale. Our self-hosted Qwen3-VL-30B reaches MCC 0.47 and balanced accuracy 0.74—competitive with the best affordable alternative—while agreeing with the human on 78% of objects (Cohen's $\kappa = 0.47$, moderate) and retaining a pool in which 84% of kept objects are human-confirmed correct.

We ablate prompt-level design choices in Tab. 9, flipping one choice at a time off a base configuration. Three findings drive the deployed setup. (i) Dropping the drawn point annotation and grounding the judge on the plain image maximizes recall on incorrect objects (0.78) but over-rejects, dropping agreement to 67%. (ii) Reason-first prompting and explicit pixel grounding each curb this over-rejection, raising agreement to $\approx 77\%$. (iii) Majority voting gives no measurable gain over a single pass, and zooming on the object raises agreement only by catching fewer bad objects. We therefore deploy Qwen3-VL-30B per object on the interior anchor, judging point and label from the plain image, with reason-first prompting, pixel grounding, no voting, and near-duplicate suppression. Relative to the base configuration this combination lifts agreement from 73% to 78% and κ from 0.39 to 0.47, trading peak bad-recall ($0.78 \rightarrow 0.63$) for a better-calibrated gate that does not needlessly inflate the removal pool.

The gate acts as a precision control rather than a recall booster: removing it slightly raises downstream recall but inflates predictions per image to 12.1 (ground truth 7.25) from the gated 10.4.

C TRAINING CONFIGURATION

Objects are represented by a 2D anchor point in normalized 0–1000 image coordinates, a noun-phrase label, and a short description. Training examples are sampled from the pseudo-label and synonym/distractor pools of Sec. 3. We sample `ADD` and `EDIT-REMOVE` tasks with probability 0.7 and 0.3, respectively. Within `EDIT-REMOVE`, 40% of examples are no-ops with an empty target set, while the remaining examples are distributed evenly across attribute edits, removals, and mixed corruptions. Within `ADD`, half of the examples provide a partial scene map and require prediction of the remaining visible objects.

Rather than sampling a single corruption rate per image, we corrupt each object independently with probability $q \sim \mathcal{U}[0.05, 0.20]$. This removes image-level correlations and forces the model to decide independently which objects require modification, while naturally making single-object corrections common during training.

To avoid ambiguity between operations, perturbation magnitudes occupy disjoint ranges in anchor space: no-op jitter ($\approx 1\%$ of the frame), edit displacements (1.5–10%), and removal displacements ($\geq 30\%$). Finally, scene-map entries are serialized in spatial order and re-sorted after corruptions are inserted, preventing the model from exploiting list position to identify edited or hallucinated objects.

Table 10 gives the full training configuration for the main SceneLM runs.

Setting	Value
Base model	Qwen3-VL-(4B/2B)-Instruct Qwen Team (2025)
Trainable parameters	vision encoder + projector + LM (full fine-tune)
Precision	bf16
Optimizer	AdamW
Learning-rate schedule	cosine decay, warmup ratio 0.05
Peak learning rate	2×10^{-5}
Weight decay	0.05
Epochs	3
Per-device train batch size	16
Gradient accumulation steps	1
Effective batch size	128
Sequence length (`model_max_length`)	4096
Dataset Sampling	Uniform
Image-token pixel bounds	$\min = 784,\ \max = 50{,}176$
Distributed strategy	DeepSpeed ZeRO-2
Hardware	$8\times$NVIDIA A100 (80 GB) per run
Object-ordering supervision	`shuffle_object_order=structured`
Loss reduction	`token_mean`
Training-task mixture	ADD + EDIT + `refine` + `dual_crop` (Section 4.2)
Serving	vLLM Kwon et al. (2023)

Table 10: **Training configuration** (SceneLM headline run).

D PER-FRAME COMPONENT DETAILS

This section provides additional details on the per-frame experiments presented in Sec. 5.2. We first analyze the prompts used for the single-frame ADD baselines, followed by a detailed description of the evaluation protocol for the EDIT/REMOVE task.

D.1 ADD- BASELINES

The performance of VLMs is highly dependent on the prompting strategy. To ensure a fair comparison, we evaluate a diverse set of prompts for both Qwen3-VL-4B and Qwen3-VL-8B. The resulting detection performance is summarized in Tab. 11, while the full prompt templates are provided in the accompanying code release.

The experiments demonstrate substantial variation across prompts. Prompts encouraging exhaustive object discovery generally achieve higher localization and labeling accuracy, but also increase the likelihood of repetitive generation (loop behavior) and produce substantially more object predictions per image. The prompt used throughout the paper provides the best trade-off between detection quality and stable generation.

		COCO (%)		LVIS (%)		nuImages (%)		Average	
Model	Prompt	loc	label	loc	label	loc	label	Loop (%)	Pred./img
Qwen3-VL-4B	Exhaustive	51	39	30	15	29	24	40	23.67
	Few-shot	51	40	29	15	31	25	23	17.56
	LU	19	15	12	7	1	1	4	50.93
	LUD	4	3	2	1	0	0	1	47.80
	LUS	26	20	15	8	4	3	1	35.58
	LXY	45	35	25	13	20	16	21	11.94
	LXY3 System	44	35	22	11	26	22	26	9.44
	Expert	47	37	26	13	28	24	34	12.01
	Systematic	49	39	26	14	30	25	28	13.69
	Terse	50	40	28	15	26	21	32	27.16
Qwen3-VL-8B	LU	37	28	20	10	21	17	11	8.14
	LUD	35	27	18	9	21	17	12	7.49
	LUS	44	28	24	11	23	17	5	7.06
	LXY	40	30	24	13	20	16	23	17.44
	Exhaustive	42	29	26	13	24	19	36	35.09
	Few-shot	51	38	30	16	25	20	20	31.20
	LXY3 System	39	30	21	11	25	20	16	9.42
	Expert	45	32	28	14	26	21	37	28.28
	Systematic	35	24	22	11	19	15	29	22.62
	Terse	30	19	18	8	13	10	24	19.86

Table 11: Prompt ablation for single-frame object prediction.

D.2 EDIT/REMOVE

The EDIT/REMOVE task evaluates whether the model can identify incorrect objects in the scene representation and determine the appropriate corrective action. For completeness, Tab. 12 defines all evaluation metrics used in the main paper.

The evaluation is organized into three stages. First, the decision layer measures whether the model correctly determines if an object should be kept or modified. Second, the action layer evaluates whether the model correctly predicts an EDIT or REMOVE operation. Finally, the quality layer measures the correctness of the predicted edits by evaluating both the corrected object label and the updated object position.

Metric	Formula	Definition
Decision layer — for each object, keep vs. address (edit ∪ remove)		
Dec P (precision)	$\mathrm{TP}/(\mathrm{TP}+\mathrm{FP})$	Of the objects the model *touched* (edited or removed), the fraction that genuinely needed fixing. Low ⇒ the model meddles with objects it should have left alone.
Dec R (recall)	$\mathrm{TP}/(\mathrm{TP}+\mathrm{FN})$	Of the objects that *needed* fixing, the fraction the model touched. Low ⇒ the model misses corruptions and leaves them in the scene.
Dec F1	$2PR/(P+R)$	Harmonic mean of decision precision and recall: the headline "did it act on the right objects, and only those?". The action-type and quality metrics are all conditioned on getting the decision right first.
FalseTouch (↓)	$\mathrm{FP}/(\mathrm{FP}+\mathrm{TN})$	Of the objects that should be *left alone*, the fraction the model wrongly touched (lower is better). The clean-object error rate (denominator = all keep objects), companion to decision precision.
Action-type layer — among addressed objects, edit vs. remove		
Edit R (recall)	$\mathrm{TP}_e/(\mathrm{TP}_e+\mathrm{FN}_e)$	Of the objects whose correct action is EDIT, the fraction the model both addressed *and* action-typed as edit (removing or missing it counts against).
Rem R (recall)	$\mathrm{TP}_r/(\mathrm{TP}_r+\mathrm{FN}_r)$	Of the objects whose correct action is REMOVE, the fraction the model both addressed *and* action-typed as remove (hallucinations, duplicates, big shifts, . . .).
Quality layer — among objects correctly action-typed as edit		
Label acc	$n^{\mathrm{ok}}_{\mathrm{label}}/n_{\mathrm{label}}$	Among label-corrupted objects the model correctly chose to edit, the fraction whose predicted label matches the clean ground-truth label (synonym-aware).
Pos ok	$n_{\mathrm{closer}}/n_{\mathrm{pos}}$	Among position-corrupted objects the model correctly chose to edit, the fraction whose predicted point landed *closer* to the true location than the corrupted point was (residual after $<$ residual before); credits any improvement, not pixel-perfect placement.

Table 12: Fix-tasks (`EDIT-REMOVE`) evaluation metrics. The model is shown a context of objects (each with an `id`) in which some are deliberately corrupted, and must decide per object whether to *keep* or *address* it, where addressing is an *edit* (fix the label and/or point) or a *remove*. Scoring proceeds in three layers, each conditioned on the previous. The decision confusion matrix counts objects as **TP** (needed fixing and was touched), **FP** (touched but should have been kept — a "false touch"), **FN** (needed fixing but was kept), and **TN** (correctly left alone); P, R in Dec F1 are Dec P and Dec R. Subscripts e, r denote the edit / remove action-type confusion counters. All metrics are micro-averaged over every object in the evaluation set; an undefined metric (zero denominator) is reported as "—".

E Additional Experiments

This section reports additional ablation studies and supplementary experiments that did not fit in the main paper. We first analyze several training design choices, followed by additional FARM-style experiments that estimate object extent during retrieval.

E.1 Additional Ablations

4B ablation: We perform ablations for the SceneLM-4B when using or not using short descriptions during training and inference, Dense object proposal, `ADD`-only, without the quality check. Results are displayed in Tab. 13 and follow the conclusions drawn in the main paper on the 2B model. On the TagMap benchmark, Tab. 14 show the ablations and importantly show that Dense-SceneLM can be useful in certain scenarios.

Loss reduction: The language modeling loss is naturally accumulated per token, causing training examples with many objects or longer descriptions to contribute more to the optimization. Table 15 compares three reduction strategies: token mean, sample mean, and object mean. The downstream retrieval performance is largely unchanged, indicating that the method is not particularly sensitive to this design choice.

Dataset weighting: The training mixture contains datasets of widely different sizes. We therefore investigate three sampling strategies: the natural sampling distribution, square-root weighting based on dataset size, and uniform dataset sampling with capped upsampling to avoid overfitting the smallest datasets. As shown in Tab. 16, uniform sampling improves performance on ScanNet but slightly reduces performance on HM3D, resulting in only minor differences overall.

Model size: We compare 2B, 4B, and 8B variants of SceneLM in Tab. 17. For practical reasons (fitting training on available GPU) the 8B model is trained without short descriptions. While not perfectly comparable, the 2B model achieves the strongest overall retrieval performance despite having fewer parameters. Increasing the model size does not significantly improve downstream retrieval, suggesting that the bottleneck lies in the dataset, mapping formulation, and/or supervision rather than model capacity.

Image resolution: We evaluate the effect of the maximum image resolution used during both training and inference. Internally, the limit is specified as a maximum number of visual tokens, but for readability we report the equivalent image resolution (e.g., resolution 316 corresponds to a maximum of 100 352 pixels). The results in Tab. 18 show that increasing the resolution generally improves retrieval performance up to a point, after which the gains saturate. The 2B model is more sensitive to this hyperparameter than the 4B model.

	ScanNet				HM3D			
Method	Acc@1	R@5	R@10	MRR	Acc@1	R@5	R@10	MRR
(4B) w Desc.	36.3%	69.9%	77.8%	50.3%	7.1%	19.2%	27.1%	13.0%
(4B) w/o Desc.	36.7%	67.5%	75.4%	49.7%	6.2%	17.0%	25.2%	11.5%
(4B) Dense w Desc.	34.2%	64.2%	74.4%	47.3%	6.0%	15.9%	22.5%	10.8%
(4B) `ADD`-only	30.4%	54.2%	60.9%	41.1%	6.0%	16.9%	24.5%	11.4%
(4B) Quality Check off	35.0%	68.9%	78.3%	49.2%	6.1%	18.3%	26.8%	12.1%

Table 13: 4B SceneLM with natural dataset sampling.

Method	P@0.1	P@0.5	P@1.0	P@2.0	R@0.1	R@0.5	R@1.0	R@2.0
(4B) w Desc.	43.2%	50.0%	55.5%	62.1%	28.7%	66.4%	76.6%	81.7%
(4B) w/o Desc.	42.2%	49.4%	54.9%	61.1%	27.7%	64.7%	72.2%	78.4%
(4B) Dense w/o Desc.	40.8%	48.2%	53.8%	60.5%	40.0%	74.2%	82.4%	86.4%
(4B) Dense w Desc.	42.9%	49.9%	55.5%	61.6%	35.3%	70.4%	78.6%	82.5%
(4B) `ADD`-only	38.8%	45.6%	51.5%	58.3%	27.9%	64.1%	72.5%	78.8%
(4B) Quality Check off	38.1%	45.3%	51.1%	57.6%	32.5%	68.1%	77.1%	82.1%

Table 14: Ablation on MP3D dataset, test18 split, macro-averaged over ObjectNav-21 classes.

	ScanNet				HM3D			
Method	Acc@1	R@5	R@10	MRR	Acc@1	R@5	R@10	MRR
(4B) Token Mean	36.3%	69.9%	77.8%	50.3%	7.1%	19.2%	27.1%	13.0%
(4B) Sample Mean	36.1%	67.1%	73.0%	49.2%	6.6%	18.4%	25.9%	12.2%
(4B) Object Mean	36.6%	70.6%	78.1%	50.8%	6.7%	19.0%	27.2%	12.7%

Table 15: Scene-graph retrieval summary from both benchmarks, 3D point-in-box (extent-free). Ablating loss reduction techniques. Natural dataset weighting is used.

	ScanNet				HM3D			
Method	Acc@1	R@5	R@10	MRR	Acc@1	R@5	R@10	MRR
(4B) Natural	36.3%	69.9%	77.8%	50.3%	7.1%	19.2%	27.1%	13.0%
(4B) SQRT.	37.3%	70.2%	78.4%	51.0%	6.5%	18.2%	26.6%	12.2%
(4B) Uniform	39.3%	71.3%	79.2%	52.6%	6.6%	18.8%	26.5%	12.5%

Table 16: Scene-graph retrieval summary from both benchmarks, 3D point-in-box (extent-free). Ablating dataset weighting.

	ScanNet				HM3D			
Method	Acc@1	R@5	R@10	MRR	Acc@1	R@5	R@10	MRR
(2B) Sparse w Desc.	38.3%	70.7%	77.9%	51.8%	7.1%	19.5%	28.1%	13.1%
(4B) Sparse w Desc.	36.3%	69.9%	77.8%	50.3%	7.1%	19.2%	27.1%	13.0%
(8B) Sparse w/o Desc.	36.7%	67.5%	75.4%	49.7%	5.9%	16.5%	23.9%	11.1%

Table 17: Scene-graph retrieval summary from both benchmarks, 3D point-in-box (extent-free). Ablating model size. Natural dataset weighting is used. 8B model is trained without descriptions to fit trainings on the hardware we have available.

	ScanNet				HM3D			
Img. Res.	Acc@1	R@5	R@10	MRR	Acc@1	R@5	R@10	MRR
(2B) 112x112	31.9%	61.5%	69.0%	44.3%	5.0%	14.4%	20.8%	9.5%
(2B) 158x158	34.3%	66.9%	75.0%	47.7%	5.5%	15.6%	22.3%	10.4%
(2B) 224x224	38.3%	71.5%	79.4%	51.9%	6.3%	18.0%	25.9%	11.9%
(2B) 316x316	38.3%	70.7%	77.9%	51.8%	7.1%	19.5%	28.1%	13.1%
(2B) 448x448	36.7%	70.3%	77.1%	50.8%	7.4%	20.1%	28.4%	13.4%
(2B) 512x512	38.0%	70.2%	76.8%	51.3%	6.8%	19.4%	27.8%	12.9%
(4B) 224x224	36.4%	70.1%	78.3%	50.7%	6.6%	19.0%	26.7%	12.5%
(4B) 316x316	36.3%	69.9%	77.8%	50.3%	7.1%	19.2%	27.1%	13.0%
(4B) 448x448	37.8%	70.5%	78.4%	52.0%	6.7%	18.3%	26.2%	12.4%
(4B) 512x512	37.3%	69.3%	76.4%	50.7%	6.4%	18.4%	26.3%	12.1%

Table 18: Scene-graph retrieval summary from both benchmarks, 3D point-in-box (extent-free). Measuring impact of image max resolution during training and evaluation. Natural dataset weighting.

E.2 RETRIEVAL UNDER THE FARM EVALUATION PROTOCOL

SceneLM represents each mapped object by a 3D point and does not directly predict its spatial extent. In contrast, the original FARM evaluation protocol determines whether a retrieved object is correct using 3D axis-aligned bounding-box intersection over union (AABB IoU), and therefore requires an estimate of object size. To evaluate SceneLM under this protocol, we use Qwen3-VL-4B to estimate the size of the queried object and use this estimate to construct the corresponding 3D bounding box. The estimate is based on the predicted general size of the object we are looking for, and not dependent on any visual information, and the box is naively centered around the predicted point. To get an indication how much performance is affected by these object shape estimates, we use the ground truth object sizes as an *Oracle*. The results are reported in Tab. 19.

Under the extent-based protocol, SceneLM achieves slightly higher Acc@1 than FARM on both ScanNet and HM3D, while FARM performs better on Recall@5, Recall@10, and MRR. These results indicate that SceneLM remains competitive under the original FARM protocol, but also show that extent estimation introduces an additional source of error when the underlying scene representation stores only object positions. However, when using the Oracle sizes the SceneLM performance goes up significantly. We thus believe that extending SceneLM to predict object extent directly, removing the dependence on an external size estimator, is a promising direction for future work.

	ScanNet				HM3D			
Method	Acc@1	R@5	R@10	MRR	Acc@1	R@5	R@10	MRR
BBQ	22.7%	35.4%	35.9%	28.1%	3.1%	6.5 %	7.9%	4.6%
FARM	29.4%	60.0%	70.6%	42.8%	4.0%	11.8%	17.9%	8.5%
SceneLM-4B								
Estimated	30.0%	57.1%	64.5%	41.4%	4.4%	11.5%	15.7%	7.7%
Oracle	38.2%	72.5%	80.7%	52.6%	7.6%	20.2%	28.5%	13.7%

Table 19: Language-grounded retrieval under the original FARM 3D AABB IoU protocol. Because SceneLM predicts object positions without extents, Qwen3-VL-4B is used to estimate object size for bounding-box construction. Oracle uses the ground-truth size of the bounding-box and achieves higher performance, indicating that size estimation is an important part of these performance numbers.

F QUALITATIVE EXAMPLES

We provide additional qualitative examples to better understand the behavior of SceneLM during incremental scene mapping. In contrast to methods that construct a static representation from a complete observation sequence, SceneLM continuously updates a structured object-level map through `ADD` and `EDIT` operations. The following examples illustrate both the strengths and limitations of this formulation.

Fig. 6 shows an example of incremental scene expansion. As new regions of the scene become visible, SceneLM successfully adds newly observed objects to the map while preserving previously detected entities. The model is able to recognize a wide range of object categories and attributes, such as chair colors and materials, demonstrating its open-vocabulary mapping capabilities. However, the example also highlights limitations in fine-grained object recognition, where visually similar objects may be assigned an incorrect category.

Beyond recognition errors, maintaining a consistent object-level map over time introduces additional challenges. Figs. 7 to 9 show representative failure cases. A common failure mode occurs when the model incorrectly associates a new observation with an existing object, resulting in duplicate objects or unnecessary edits. Additionally, the model can occasionally apply incorrect removal operations by confusing valid objects with previously introduced errors. Nevertheless, these examples also demonstrate successful correction of earlier mistakes, where redundant or incorrectly positioned objects are removed from the map.

Finally, we visualize complete reconstructed HM3D scene maps in Figs. 10 and 11. These examples demonstrate that SceneLM can construct rich open-vocabulary scene representations containing diverse object categories. At the same time, they reveal remaining challenges in object-level semantic understanding, where visually similar objects may be confused, such as a dartboard being classified as a wall clock.

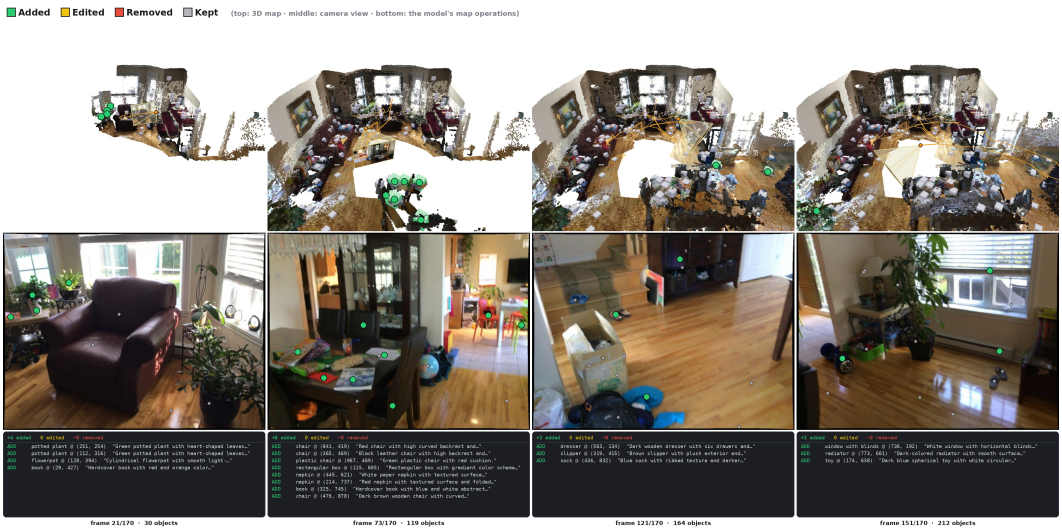


Figure 6: **Incremental scene expansion.** The model successfully adds newly observed objects as they appear in the scene. In frame 21, newly revealed flowers, pots, and books are correctly incorporated into the map. In frame 73, multiple chairs are added with accurate attributes, including color and material, while the paper sheets are incorrectly classified as napkins. In frame 151, the model identifies the hoverboard as a toy but fails to recover the specific category, while correctly adding the window and radiator.

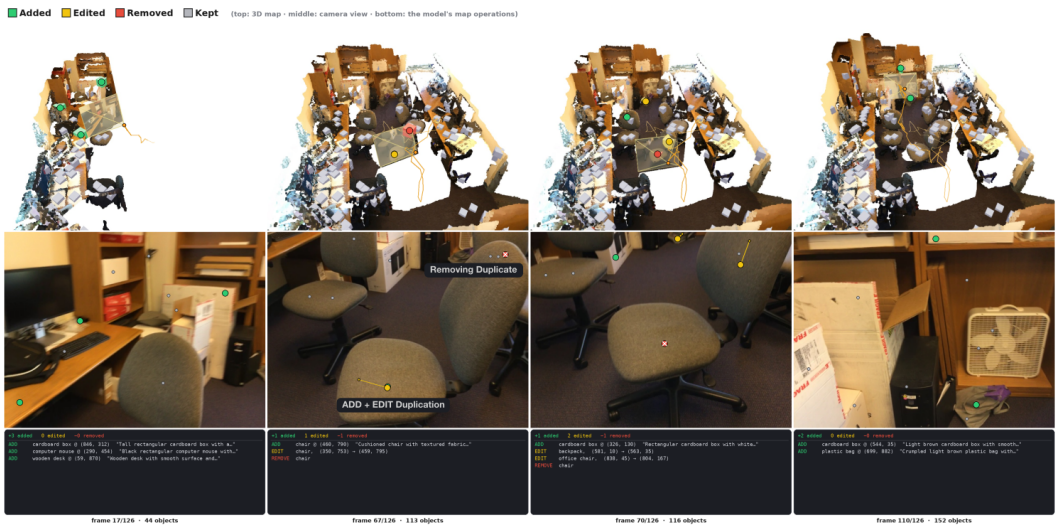


Figure 7: **Failure cases.** The model occasionally produces duplicate objects when an object is both edited and added at the same location. The example also shows successful duplicate removal, where a redundant object introduced in an earlier frame is removed from the map.

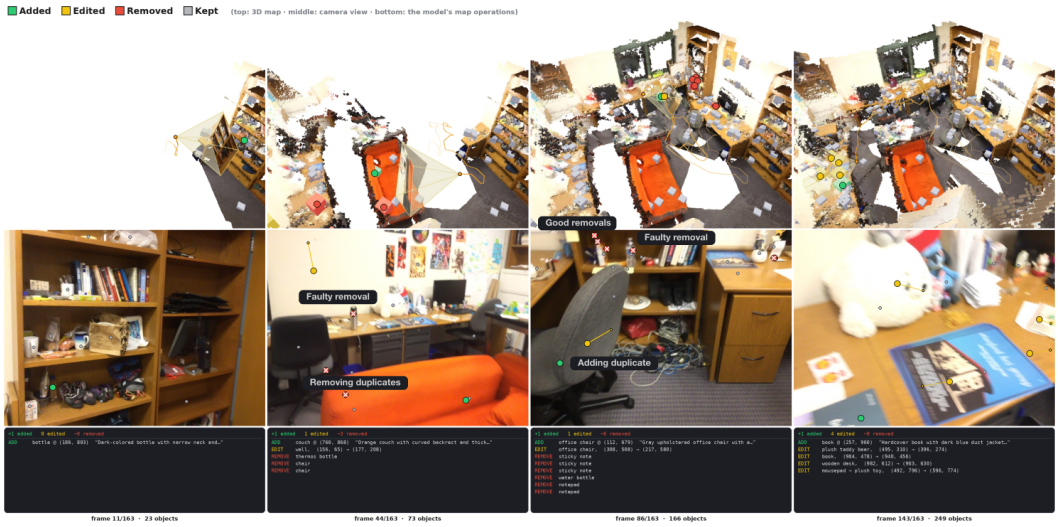


Figure 8: **Failure cases.** The model correctly removes duplicate chairs and incorrectly placed sticky notes, but can also mistakenly remove valid objects, such as the thermos bottle and water bottle.

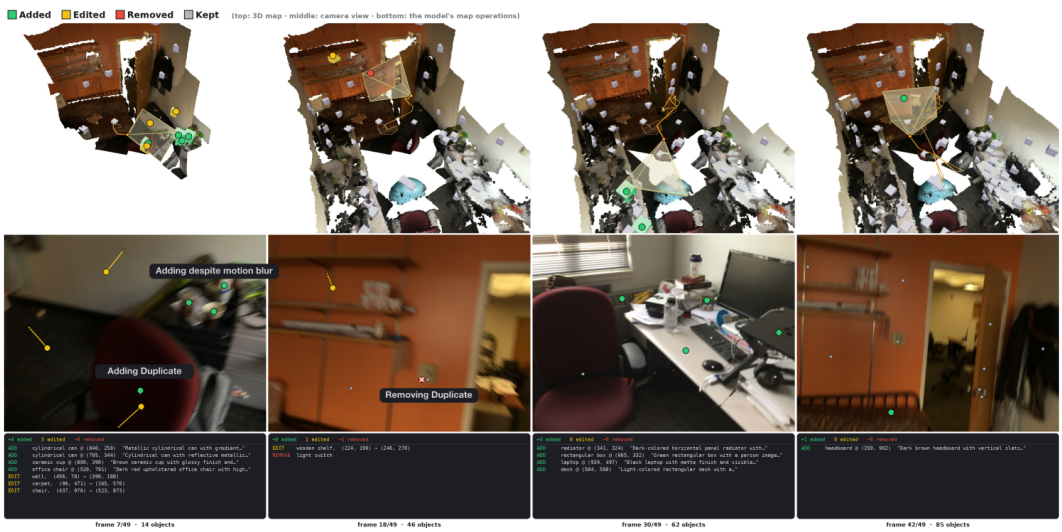


Figure 9: **Failure cases under challenging observations.** Under severe motion blur, the model can still correctly add several objects and remove redundant detections, but may introduce duplicates, such as adding a chair while updating an existing chair position.

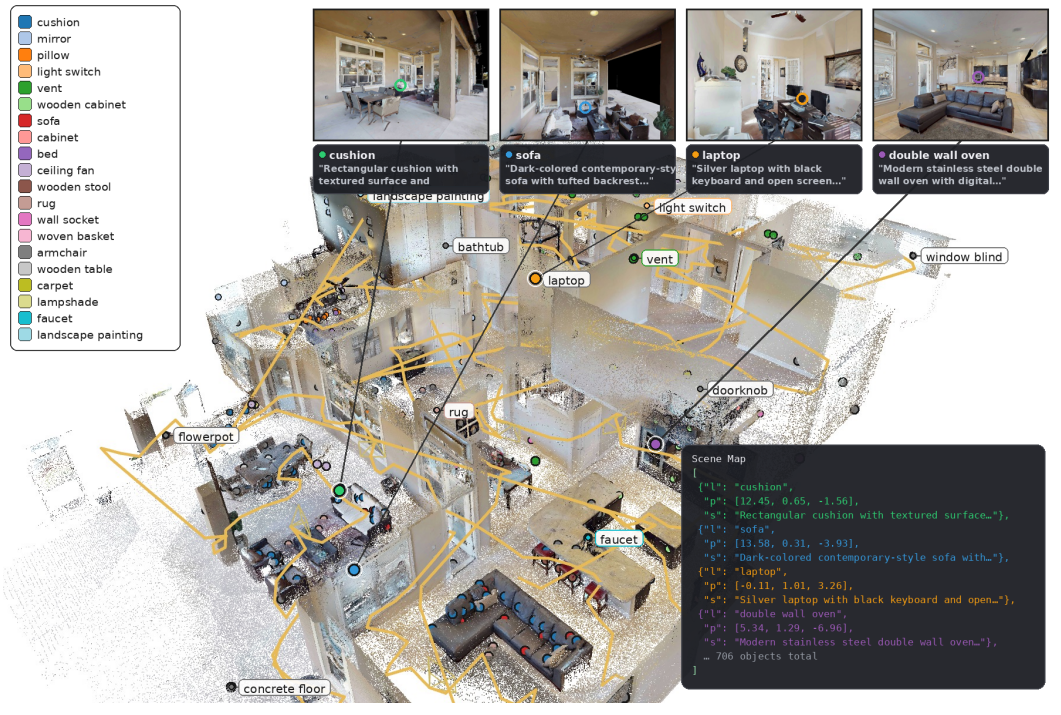


Figure 10: **Open-vocabulary scene mapping.** Visualization of a HM3D scene, demonstrating the ability of the model to represent diverse object categories using open-vocabulary descriptions.

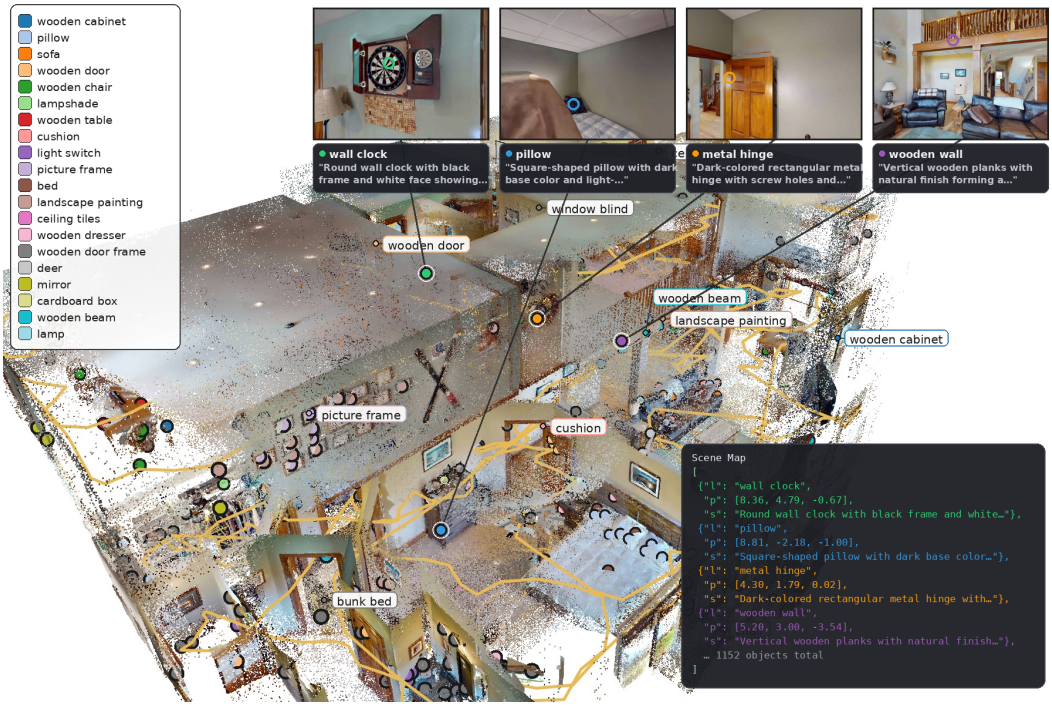


Figure 11: **Failure case in open-vocabulary recognition.** Visualization of a HM3D scene map where the dartboard is incorrectly classified as a wall clock. A subset of other objects are highlighted as dots in the map, and a few as insets with a view from the trajectory through the scene.

G REAL-WORLD DEPLOYMENT

We evaluate whether SceneLM can maintain an open-vocabulary scene map online on a mobile quadruped. We first analyze the deployment trade-offs required for real-time operation on an edge device. We then evaluate the deployed system in several real-world environments. Finally, we provide implementation details of the deployed system.

G.1 DEPLOYMENT TRADE-OFFS

Deploying SceneLM on a mobile robot requires balancing mapping quality against computational throughput. Unlike the offline experiments in the main paper, an online system must continuously process incoming observations without accumulating an ever-growing queue of images. We therefore study two aspects of the deployment. First, we investigate how model quantization affects inference throughput on the robot. Second, we determine how frequently the scene map needs to be updated as the robot moves. Together, these experiments define the deployment configuration used for all real-world experiments.

G.1.1 MODEL QUANTIZATION

We evaluate two model sizes (2B and 4B) using BF16, FP8, and Blackwell-native NVFP4 quantization. Inference is performed using vLLM on the NVIDIA Jetson AGX Thor mounted on the robot and on an RTX 5090 workstation for reference. For each configuration, we report both the decoding throughput in generated tokens per second and the sustained scene-map update rate in completed map updates per second.

Table 20 shows that lower-precision inference substantially improves throughput on both platforms. The gains are particularly pronounced on the Jetson, where inference is increasingly limited by memory bandwidth. Among all evaluated configurations, NVFP4 consistently achieves the highest sustained update rate while also requiring the smallest memory footprint. Unless otherwise stated, all subsequent deployment experiments therefore use the 2B model with NVFP4 quantization.

		Decode speed (tok/s)		**Update rate (keyframes/s)**	
Model	Precision	RTX 5090	Jetson Thor	RTX 5090	Jetson Thor
2B	BF16	315	50.1	2.59	0.53
2B	FP8	382 (1.2×)	79.7 (1.6×)	3.09	0.83
2B	NVFP4	**460** (1.5×)	**100.2** (2.0×)	**3.54**	**0.86**
4B	BF16	162	28.6	1.36	0.21
4B	FP8	232 (1.4×)	45.1 (1.6×)	1.91	0.33
4B	NVFP4	**296** (1.8×)	**68.0** (2.3×)	**2.36**	**0.51**

Table 20: Inference throughput for different model sizes and quantization strategies. Decode speed measures generated tokens per second, while update rate measures completed scene-map updates per second.

G.1.2 DISTANCE-BASED SCENE UPDATES

Rather than processing every incoming camera frame, we trigger a scene-map update once the robot has travelled a fixed distance. This avoids repeatedly processing highly overlapping views while making the update frequency independent of the robot's velocity. The robot therefore performs scene updates only when sufficient new observations are expected.

To determine a suitable distance threshold, we resample ScanNet trajectories at different spatial intervals and reconstruct the complete scene map from each sampled trajectory. Figure 12 shows the resulting mapping performance. Very small spacings introduce highly redundant observations that primarily increase duplicate detections, while large spacings reduce scene coverage and lower recall. Across the evaluated spacings, approximately $0.5\,\mathrm{m}$ provides the best trade-off between precision and recall.

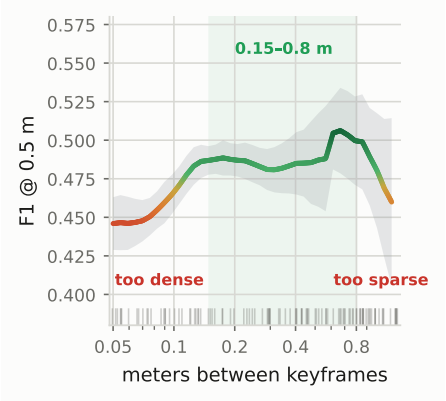


Figure 12: Mapping performance as a function of the travelled distance between processed keyframes. A spacing of approximately $0.5\,\mathrm{m}$ provides the best balance between redundant observations and scene coverage.

The update rate of the deployed model and the selected distance threshold together determine the maximum speed at which the robot can maintain online mapping. If the robot moves faster than the sustained update rate permits, intermediate frames are discarded, increasing the distance between processed keyframes rather than allowing an unbounded processing backlog to accumulate. Since mapping performance degrades only gradually around the optimum in Figure 12, occasional frame drops incur only a modest reduction in mapping quality. Based on these experiments, we deploy the 2B NVFP4 model and trigger scene-map updates every $0.5\,\mathrm{m}$ of robot motion for all real-world experiments. At a spacing of 0.5 m, the 2B NVFP4 model sustains 0.87 updates/s on the Jetson, corresponding to a nominal operating speed of approximately 0.44 m/s.

G.2 REAL-WORLD EXPERIMENTS

We evaluate the deployed system in several real-world environments to assess whether SceneLM can maintain a useful open-vocabulary scene map under realistic operating conditions. Unless otherwise stated, all experiments use the deployment configuration established in the previous section, namely the 2B model with NVFP4 quantization and distance-triggered scene updates every $0.5\,\mathrm{m}$.

G.2.1 EXPERIMENTAL SETUP

We evaluate the system in three environments with increasing complexity. The first is a laboratory environment containing a diverse collection of deliberately arranged objects. The second is a cluttered office containing many visually similar objects that challenge long-term object association. The third is an outdoor university campus route spanning several hundred meters and containing buildings, vegetation, street furniture, and construction areas.

Since no ground-truth instance-level 3D annotations are available for these environments, we evaluate the generated maps using a manually constructed object inventory.

G.2.2 QUANTITATIVE EVALUATION

The laboratory environment contains many unique objects and achieves the highest recall.Most missed detections correspond to small or thin objects that are difficult to recognize at the available image resolution.

The office environment is considerably more challenging due to repeated object instances and visually similar furniture. Here, the primary failure mode is not missing objects but reduced semantic specificity and duplicate objects. For example, a leather armchair may be described simply as a *dark chair*, while a server rack is represented by several individual computer towers. Despite these ambiguities, completely hallucinated objects remain rare across all environments.

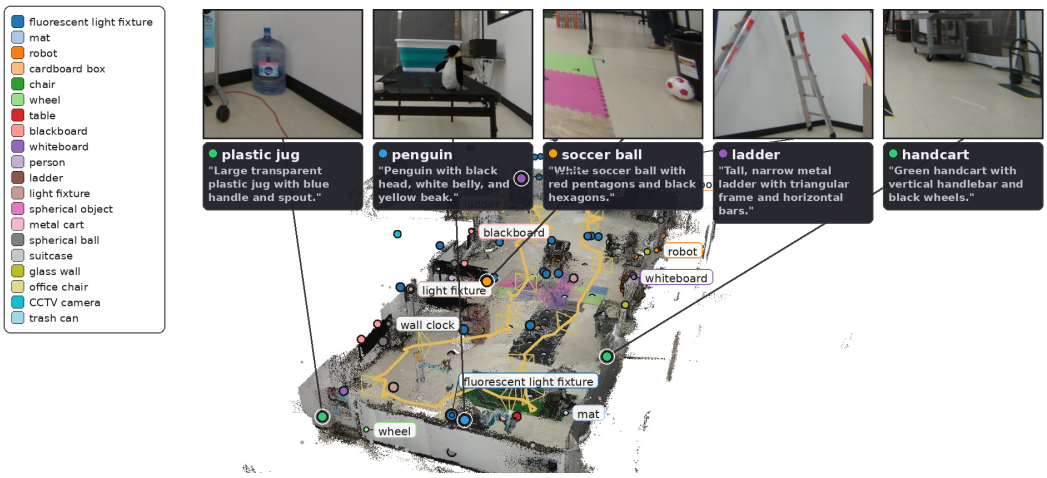


Figure 13: **Real-world deployment of SceneLM-2B-NVFP4 on a Spot quadruped.** Selected mapped objects are shown within the resulting scene map from an unseen environment. SceneLM runs onboard an NVIDIA Jetson and updates the textual map online from posed RGB-D observations.

The outdoor campus demonstrates that SceneLM scales beyond individual rooms to larger environments. Compared to workstation inference, the on-device deployment produces fewer mapped objects due to dropped keyframes when the robot temporarily exceeds the sustained processing rate. Nevertheless, the resulting scene map remains sufficiently complete for open-vocabulary querying and captures the dominant semantic structure of the environment.

G.2.3 Qualitative examples

Figures Fig. 13–Fig. 17 present representative scene maps generated during deployment. The office environment demonstrates that SceneLM successfully accumulates observations across multiple viewpoints to construct a coherent semantic map despite noisy odometry and depth estimates. Compared to the ScanNet and HM3D benchmarks, the real-world deployment exhibits a larger number of duplicate objects, probably due to the increased pose uncertainty of the onboard visual odometry system.

Figures 15–17 show examples from the outdoor campus route. The generated maps capture diverse semantic elements, including buildings, construction equipment, vegetation, benches, bicycles, and other street furniture, while maintaining a compact textual representation throughout the trajectory. However, humans moving around tends to add duplicates instead of edited as a new position is detected. These examples illustrate that the learned mapping policy generalizes beyond indoor environments and can maintain meaningful scene maps over extended real-world trajectories.

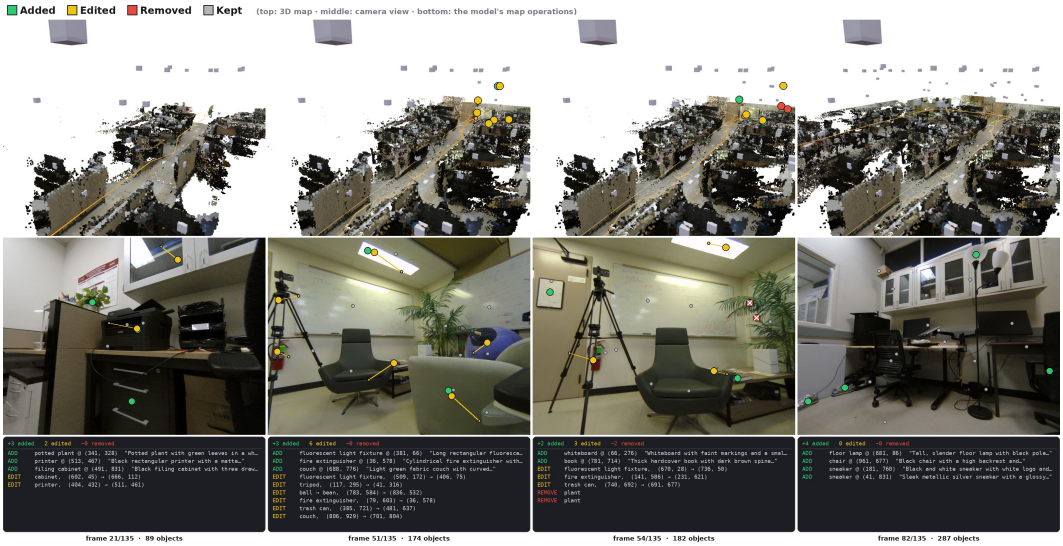


Figure 14: The map generation in the office environment. SceneLM runs completely on the quadruped. Noisier pose and depth estimation makes `EDIT` being used more. However, we note more duplicates compared to the ScanNet and HM3D datasets where poses are more accurate.

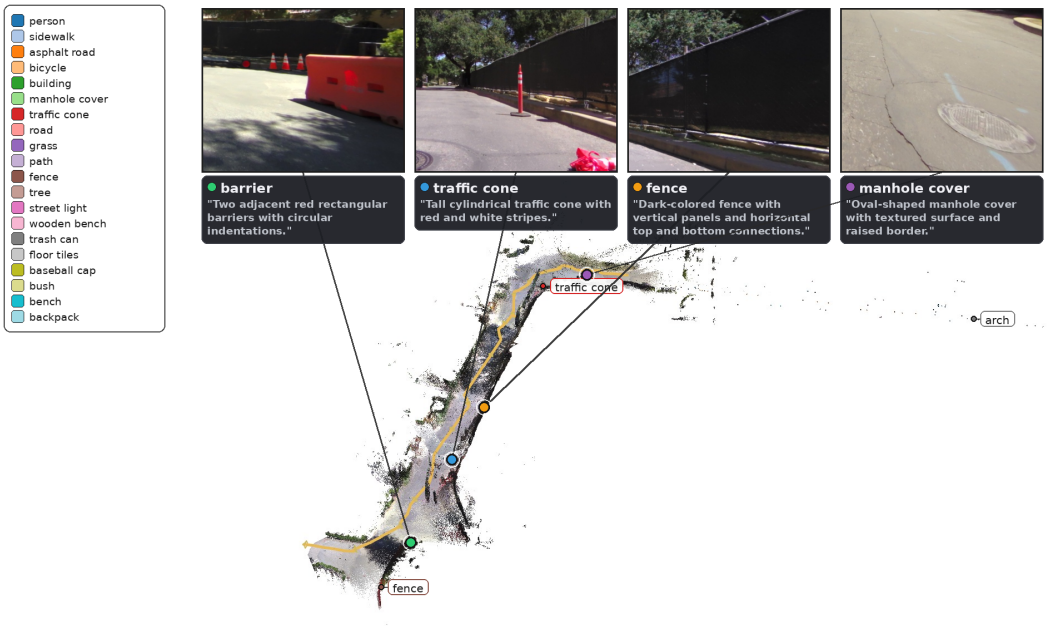


Figure 15: The map generation for a small part of the outdoor campus environment containing a construction zone. SceneLM runs completely on the quadruped.

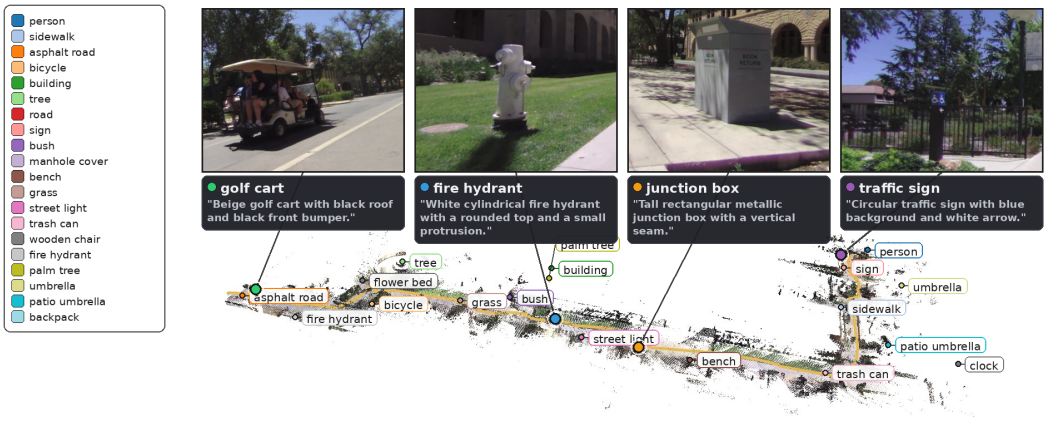


Figure 16: The map generation for a small part of the outdoor campus environment containing a plaza. SceneLM runs completely on the quadruped.

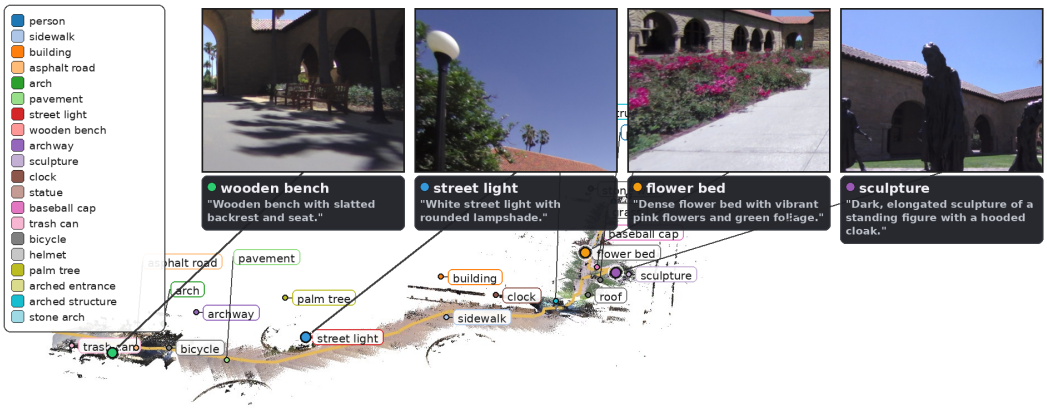


Figure 17: The map generation for a small part of the outdoor campus environment containing a construction zone. SceneLM runs completely on the quadruped.

G.3 Implementation details

The deployment system serves SceneLM using vLLM on the NVIDIA Jetson AGX Thor mounted on the robot. The model runs continuously as part of the robot autonomy stack and receives RGB-D observations together with camera poses estimated by the onboard localization system. Whenever the robot has travelled $0.5\,\mathrm{m}$ since the previous processed keyframe, the current image and the visible portion of the scene map are forwarded to the model to produce the next scene-map update.

To maximize throughput, we enable vLLM prefix caching since consecutive prompts share a large fraction of their context. All deployment experiments use the same inference configuration as the benchmark experiments unless otherwise stated.

The throughput measurements reported in Table 20 are obtained by replaying 36 ScanNet trajectories through the deployed inference pipeline. Reported update rates correspond to completed scene-map updates per second, while decode throughput measures generated tokens per second as reported by the vLLM runtime. Repeating individual trajectories results in only minor variation (1–3%), while substantially larger differences are observed between scenes, indicating that throughput is primarily determined by scene complexity rather than runtime variability.

One practical limitation is that no single version of vLLM currently supports all evaluated precisions on the Jetson platform. BF16 and NVFP4 experiments are therefore performed using vLLM 0.19, while FP8 experiments use vLLM 0.13 due to current kernel compatibility. Consequently, the reported FP8 throughput slightly understates the performance achievable with newer kernels and should therefore be interpreted as a conservative estimate.