Appendix

Appendix I
Prompt Example

To incorporate prior knowledge of the robot's operating environment, we utilize GPT-4o to generate a potential list of detection classes. The prompt and the returned class list are as follows:

User Prompt

I am deploying a robot in an indoor house setting and the robot needs a list of object categories to detect. Based on class name formats from the ScanNet and Replica datasets, provide 100 common indoor categories from living rooms, bedrooms, restrooms, kitchens, etc. The last category should be 'null'.

GPT-4o Output

chair, sofa, table, bed, desk, cabinet, shelf, book, lamp, tv, television stand, refrigerator, microwave, oven, sink, washing machine, dryer, stove, dishwasher, toaster, kettle, coffee maker, towel, toilet, shower, bathtub, mirror, picture frame, door, window, floor lamp, ceiling lamp, fan, air conditioner, heater, carpet, rug, trash can, recycling bin, iron, ironing board, broom, vacuum cleaner, trash bag, laundry basket, tissues, soap dispenser, toilet paper, toothbrush, toothpaste, hair dryer, shampoo, conditioner, shampoo dispenser, toilet brush, shampoo bottle, soap bar, candle, plant, pot, vase, clock, speaker, remote control, fan heater, curtains, blinds, curtain rod, frame, desk chair, bed frame, dresser, nightstand, wardrobe, closet, pillows, blanket, mattress, pillowcase, slippers, shoes, hangers, coat rack, wall, floor, ceiling, staircase, hallway, entryway, shelf unit, drawer, light switch, outlet, plug, router, computer, monitor, keyboard, mouse, printer, null.

This output class list, named gpt_indoor_general in the released code, was used for our indoor experiments.

Appendix II
Detailed Experiment Setup

A. Datasets

For ScanNet, we use the following scenes: scene0011_00, scene0050_00, scene0231_00, scene0378_00, and scene0518_00. We adopt the ScanNet200 benchmark, which shares the same geometry as ScanNet20 but provides a broader set of class categories and more detailed semantic annotations for evaluating 3D scene understanding methods. For Replica, the evaluated scenes are: office0-office4 and room0-room2. To evaluate query performance across a wider range of semantic categories, we use HM3DSem, specifically the scenes 00829, 00848, and 00880. To enrich the object categories, we place six objects from the YCB dataset at random positions within each HM3DSem scene. The object models used are: 003_cracker_box, 005_tomato_soup_can, 011_banana, 019_pitcher_base, 024_bowl, 025_mug, 029_plate, and 037_scissors.

TABLE VIII: Queried Objects in HM3D Test

Scene	Objects for Navigation Evaluation
00829	chair, picture, towel, table, ottoman, tap, sofa, bin, tv, cabinet, magazine, washbasin counter, bed, telephone, clothes, bathtub, bag, tv stand, decoration, toilet, cracker box, soup can, pitcher, bowl, plate, scissors
00848	pillow, tap, stool, toilet paper, towel, magazine, vase, bed, armchair, bowl of fruit, bathroom counter, christmas tree, bench, kettle, coffee maker, microwave, cooker, refrigerator, kitchen island, bathtub, cracker box, soup can, pitcher, mug, plate, scissors, banana
00880	shelf, painting, table, cabinet, curtain, container, mirror, hat, tv, washing machine, laptop, clothes, couch, microwave, sink, trashcan, dishwasher, ironing board, printer, desk, cracker box, soup can, pitcher, bowl, plate, scissors

B. Metric Definitions

In semantic segmentation evaluation, three metrics mIoU, FmIoU, mAcc are defined as follows:

$$\text{mIoU} = \frac{1}{C}\sum_{i=1}^{C}\frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i + \text{FP}_i} \tag{8}$$

$$\text{FmIoU} = \sum_{i=1}^{C}\frac{\text{TP}_i + \text{FN}_i}{\sum_{j=1}^{C}(\text{TP}_j + \text{FN}_j)} \times \frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i + \text{FP}_i} \tag{9}$$

$$\text{mAcc} = \frac{1}{C}\sum_{i=1}^{C}\frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i} \tag{10}$$

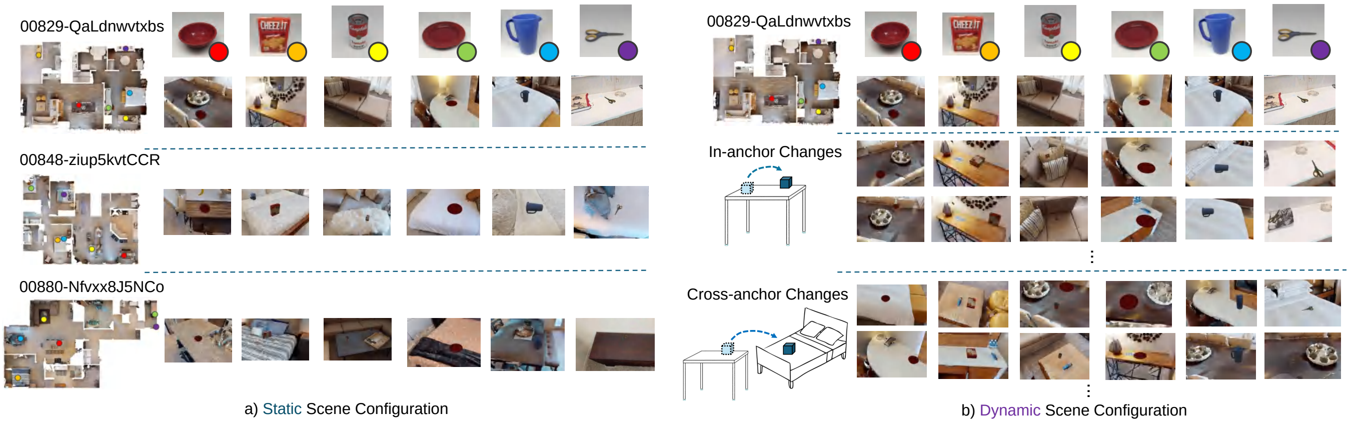
where C is the number of classes, and TP_i, FP_i, and FN_i denote the true positives, false positives, and false negatives for class i, respectively. In FmIoU, the first term represents the class frequency, while the second is the IoU of class i.

C. List of Navigation Tasks

In the HM3DSem object navigation experiments, we selected representative objects from the ground truth labels, covering a variety of common categories found in indoor environments. The complete list of objects used for the navigation tasks is provided in Table VIII. The added YCB objects are marked with underline, and Fig. 7-a illustrates the original placement of these YCB objects across the test scenes.

D. Dynamic Setting

The added YCB dataset objects are also used in dynamic change query experiments. Following the two types of dynamic changes defined in Sec.V-A.1, each object is moved three times for both *in-anchor* and *cross-anchor* changes. Fig. 7 uses scene 00829 as an example to illustrates how these changes are applied.



a) Static Scene Configuration

b) Dynamic Scene Configuration

Fig. 7: Examples of YCB object configurations: a) YCB objects are manually placed across three HM3D scenes. b) The YCB objects are dynamically relocated via two dynamic change types for the navigation test.

TABLE IX: Ablation Study on Feature Weighting

		Replica					Scannet		
$\mathbf{f}_{\text{image}}$	$\mathbf{f}_{\text{text}}$	FmIoU	mAcc	mIoU	$\mathbf{f}_{\text{image}}$	$\mathbf{f}_{\text{text}}$	FmIoU	mAcc	mIoU
0.0	1.0	0.531	0.388	0.237	0.0	1.0	0.284	0.318	0.140
0.1	0.9	0.532	0.384	0.239	0.1	0.9	0.309	0.328	0.150
0.3	0.7	0.545	0.420	0.249	0.3	0.7	0.305	0.350	0.152
0.5	0.5	**0.557**	**0.439**	**0.262**	0.5	0.5	0.321	0.365	0.163
0.7	0.3	0.551	0.425	0.251	0.7	0.3	**0.334**	**0.371**	**0.167**
0.9	0.1	0.498	0.380	0.201	0.9	0.1	0.323	0.368	0.158
1.0	0.0	0.430	0.358	0.174	1.0	0.0	0.301	0.356	0.154

Appendix III
Experiments For Threshold

In this section, we present the experiments conducted to determine the threshold, as discussed in Sec. III-A.2 and Sec. IV-A.

A. Feature Weights

Regarding the weighting in Equation 1 in Sec. III-A.2, we empirically selected the weights based on an ablation study conducted on both the Replica and ScanNet datasets. As shown in Table IX, assigning weights of 0.7 and 0.3 to $\mathbf{f}_{\text{image}}$ and $\mathbf{f}_{\text{text}}$, respectively, yields the best or near-best performance across all metrics.

B. Detailed Object Classification in Abstraction

we predefine three lists to support anchor object identification and map abstraction in Sec. IV-A:

- **Volatile object examples:** A representative list of 12 categories selected from the GPT generated class list, including [“backpack”, “box”, “clothes”, . . ., “indoor-plant”].
- **Anchor object examples:** Similarly, we define a list of 12 anchor categories, such as [“stool”, “cabinet”, “couch”, . . ., “bathroom-vanity”].
- **Descriptive phrases for anchors:** We use manually defined phrases to describe anchor objects: [“furniture that is not often moved”, “furniture that is used for sitting”, “furniture that is used for placing things”].

All the objects from concrete map are embedded into a shared visual-semantic space using CLIP features, allowing us to classify them based on semantic feature similarity. To distinguish between anchor and volatile objects, we further use a two-step heuristic:

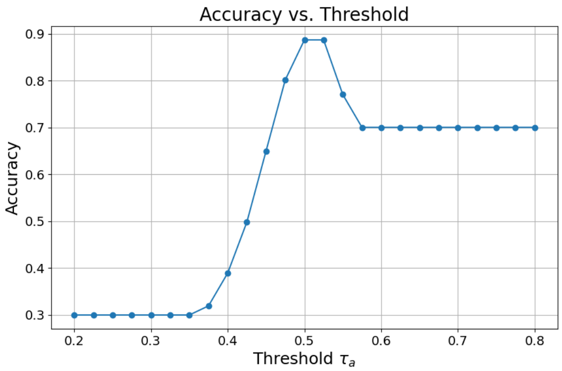


Fig. 8: Binary classification accuracy of anchor vs. volatile objects w.r.t. threshold τ_a across Replica dataset

- First, we compare each object’s CLIP embedding with two representative category example lists—one for volatile objects and one for anchor objects. If the maximum similarity to one list exceeds the other by a margin $\Delta\tau = 0.05$, the object is assigned to the corresponding category.
- Second, for ambiguous cases where the margin condition is not met, we compute the similarity between the object and a list of descriptive phrases for anchors. If the similarity exceeds a threshold τ_a, the object is classified as an anchor.

To determine the optimal threshold τ_a, we conduct a validation experiment. Specifically, we build concrete maps for 8 scenes in the Replica dataset and manually annotate each reconstructed object as either “anchor” or “volatile” to serve as ground truth. To evaluate the classification, we compute accuracy using the standard binary classification formula: Acc $= \frac{TP+TN}{TP+TN+FP+FN}$, where a prediction is considered $True$ if it aligns with the manual annotation. We sweep the threshold τ_a over the range $[0.2, 0.8]$ and plot the resulting accuracy in Figure 8. Based on this curve, we select $\tau_a = 0.5$ as it yields the highest accuracy and thus provides a reasonable trade-off.

TABLE X: Open-vocabulary 3D Semantic Segmentation and Efficiency

Dataset	Method	CLIP-Backbone	mIoU ↑	FmIoU ↑	mAcc ↑	ODR	Avg. Mem ↓	Peak Mem ↓	TPF (s) ↓
Replica	ConceptGraphs	ViT-H/14	0.1483	0.3124	0.3521	2.31	10044.6	23243.0	5.111
		Mobile-CLIP	0.1501	0.3858	0.3559	2.02	7148.9	23551.9	4.188
	HOV-SG	ViT-H/14	0.2129	0.4188	0.3794	3.13	70252.9	163238.5	45.056
		Mobile-CLIP	0.2050	0.4846	0.3835	3.81	73368.0	158126.6	42.005
	DualMap(Ours)	ViT-H/14	0.2323	0.4859	0.3832	**0.974**	3281.5	4688.9	0.458
		Mobile-CLIP	**0.2538**	**0.5207**	**0.4024**	0.967	**3095.2**	**4564.0**	**0.276**
ScanNet	ConceptGraphs	ViT-H/14	0.0646	0.1896	0.2394	7.75	13790.2	27086.0	7.605
		Mobile-CLIP	0.0882	0.3077	0.3538	6.97	9780.3	26155.2	6.301
	HOV-SG	ViT-H/14	0.1229	0.3105	0.3104	18.96	14561.2	44437.2	9.959
		Mobile-CLIP	0.1333	**0.3381**	0.3714	20.34	9223.0	25735.0	8.039
	DualMap(Ours)	ViT-H/14	**0.1611**	0.3179	0.3632	**2.55**	2607.5	4428.6	0.306
		Mobile-CLIP	0.1604	0.3288	**0.3794**	2.56	**2120.9**	**2820.2**	**0.163**

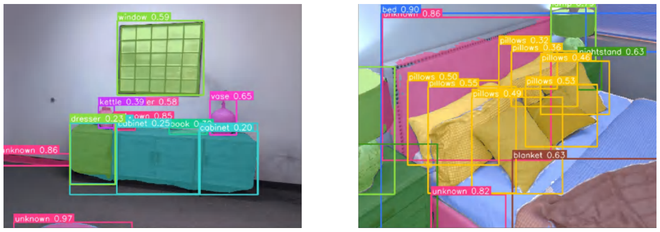
Appendix IV
More Semantic Mapping Experiments

A. Results with Other CLIP Backbone

The results presented in Table X highlight the advantages of using the Mobile-CLIP backbone for semantic segmentation tasks. Notably, methods employing Mobile-CLIP demonstrate superior accuracy and faster performance compared to those utilizing ViT-H/14 backbone. Our proposed method not only exhibits robustness across various CLIP backbones but also consistently outperforms other methods. Additionally, substituting Mobile-CLIP for ViT-H/14 significantly reduces memory usage and time costs, making it more resource-efficient for semantic segmentation applications.

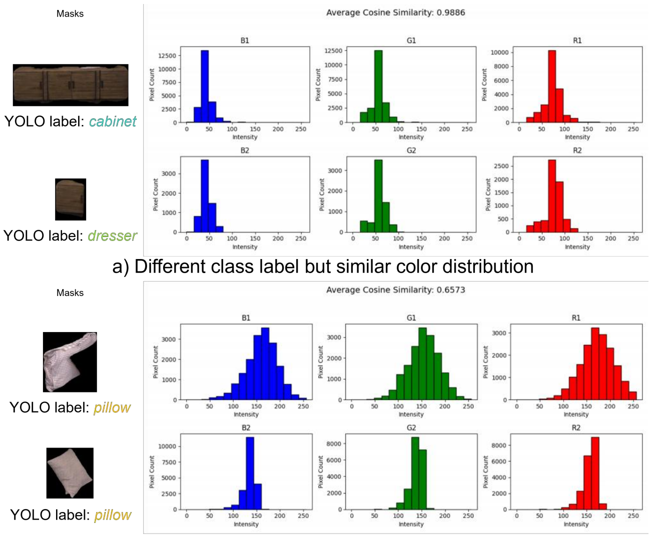
B. Results under Moving Humans

Regarding dynamic entities like humans, **DualMap is capable of generating a complete 3D semantic map without being affected by their movement**. We conduct additional experiments on the *Dynamic Objects* split of the TUM RGBD dataset, widely used in the SLAM field. The qualitative results on `freiburg3_walking_static` sequence are shown in Figure 11. As illustrated in Figures 11-b and 11-c, the mapping results are unaffected by the movements of humans. This robustness is attributed to our hybrid open-vocabulary segmentation pipeline and the object stability check mechanism (detailed in Sec. III-A-1 and Sec. III-B-3). Figure 11-a shows part of the segmentation results, where humans, absent from YOLO's class list, are occasionally segmented by FastSAM. Since human movement leads to unstable segmentation, their related objects in concrete map are easily filtered out by the stability check due to the limited observations.



a) A cabinet in Replica *room_0*   b) A cluttered bed in Replica *room_1*

Fig. 9: YOLO and FastSAM detections on two frames.



a) Different class label but similar color distribution

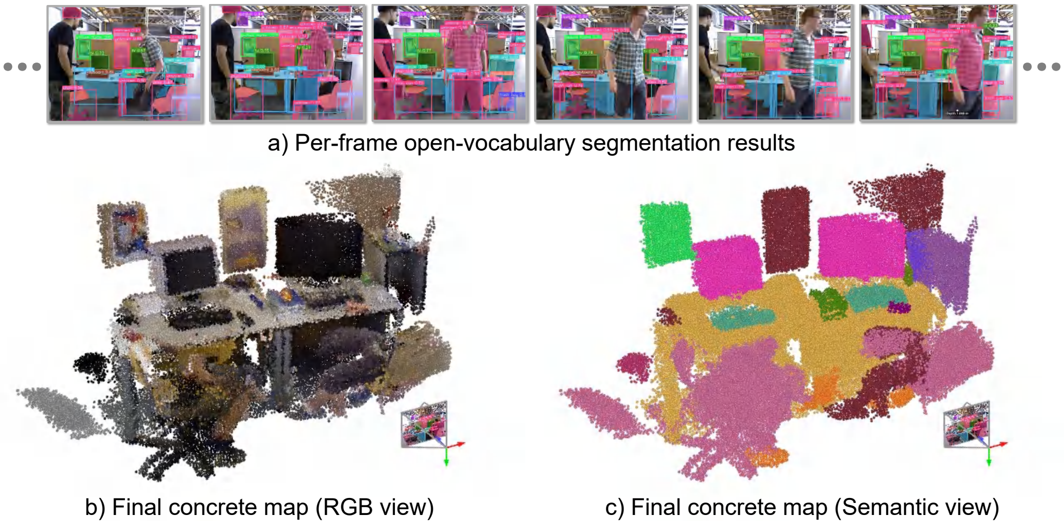
b) Same class label but different color distribution

Fig. 10: Visualization of BGR channel-wise color distributions for two overlapping segment pairs from `Replica`

C. Detection Refinement Details

There are two factors that require refinement to improve the YOLO output. First, YOLO may assign **inconsistent class labels** to different parts of the same object, as shown in Figure 9-a, where a single cabinet is partially labeled as "cabinet" and partially as "dresser." Label-based merging would fail to recognize them as one object. Second, in **cluttered scenes** such as Figure 9-b, multiple overlapping objects (e.g., pillows) often share the same class label, causing label-based methods to incorrectly merge distinct objects into one.

To address these issues, we adopt a color-based merging strategy. We first identify overlapping bounding box pairs. For each pair, we extract the pixel intensity histograms from the B, G, and R channels, dividing each into 16 equal-width bins: $[0, 16), [16, 32), \ldots, [240, 256)$. We then compute the cosine similarity between the corresponding channel histograms (e.g., $\mathbf{v}_{b1}$ and $\mathbf{v}_{b2}$ for blue), and take the average



a) Per-frame open-vocabulary segmentation results

b) Final concrete map (RGB view)

c) Final concrete map (Semantic view)

Fig. 11: concrete mapping results on `freiburg3_walking_static`

of the three channel-wise similarities as the overall color similarity. If the average similarity exceeds a threshold (0.95 in our case), we treat the segments as visually consistent and merge them. This method successfully merges parts of the same object (Figure 10-a) while avoiding incorrect merges in cluttered scenes (Figure 10-b).

Appendix V
More Performance Experiments

A. Runtime Decomposition

We break down the module-wise time cost on RTX 4090 Desktop in Table XI and Figure 12. The results show that over **57%** of the time per frame is spent on model inference (highlighted in grey). This indicates that applying inference optimizations such as TensorRT acceleration or model quantization can significantly reduce runtime with minimal accuracy impact.

B. Experiments on Laptop and Different Resolutions

While our main experiments were conducted on an RTX 4090 GPU Desktop, we conducted additional experiments on a **laptop** equipped with an RTX 3080 Laptop GPU. The additional system evaluation on Replica dataset is shown in Table XII. The results show that, at the same input resolution (1200×680), the RTX 3080 Laptop achieves comparable accuracy with a 1.6× increase in latency (around 0.4s/frame), which remains acceptable for online applications. Furthermore, by reducing the input resolution to 640×360, the system achieves a good trade-off between accuracy and efficiency, with only a minor drop in accuracy (e.g., ˜3% FmIoU loss) and comparable runtime performance to the results on the RTX 4090 Desktop. This suggests that appropriately adjusting input resolution offers a practical path for deployment on lower-cost devices.

C. Resources Usage in Long Operation

TABLE XI: Time Decomposition on Replica with 4090

TPF (s)	Modules	Submodules	Time (s)
0.2524	Observation Generation* 0.2183	FastSAM †	0.0592
		YOLO+MobileSAM †	0.0373
		Detection Filter	0.0372
		Create Obj. Pointcloud‡	0.0866
		CLIP‡	0.0619
		Observation Formatting	0.0187
	Mapping* 0.0485	Concrete Mapping	0.0485
		Abstraction	0.0000
	Visualization 0.0341	–	0.0341

*, †, and ‡ indicate parallel execution.

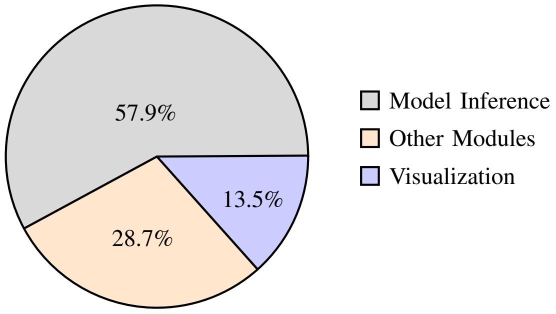


Fig. 12: System Runtime Breakdown on Replica

We additionally conducted long-term mapping experiments in both a structured hallway and a cluttered apartment. The experiments were performed using an RTX 4090 Desktop GPU with an input resolution of 1280×720. As shown in top two rows in Figure 13 and Figure 14, the average runtime per frame remains stable (**0.112s** and **0.156s**, respectively), and memory usage (CPU RAM) grows slowly and stays bounded (**around 3500–4200MB**), even in more complex environments. This is benefit from our dual-map design, the concrete map memory size remains stable, resulting in

TABLE XII: Accuracy and Efficiency Across GPU and Resolution Settings on Replica

GPU	Resolution	FmIoU ↑	mAcc ↑	mIoU ↑	TPF (s) ↓	Rel. FmIoU* ↑	Rel. TPF* ↓
RTX 4090 Desktop	1200×680	0.5508	0.4251	0.2508	0.2524	100.00%	100.00%
RTX 3080 Laptop	1200×680	0.5503	0.4256	0.2502	0.4045	99.92%	160.26%
	960×540	0.5507	0.4259	0.2526	0.3221	99.98%	127.61%
	640×360	0.5341	0.3941	0.2428	0.2717	96.97%	107.65%
	320×180	0.2646	0.1437	0.0801	0.2538	48.04%	100.55%

* Relative FmIOU and Relative TPF (Time Per Frame) are computed with respect to the 4090 Desktop at 1200×680 resolution.

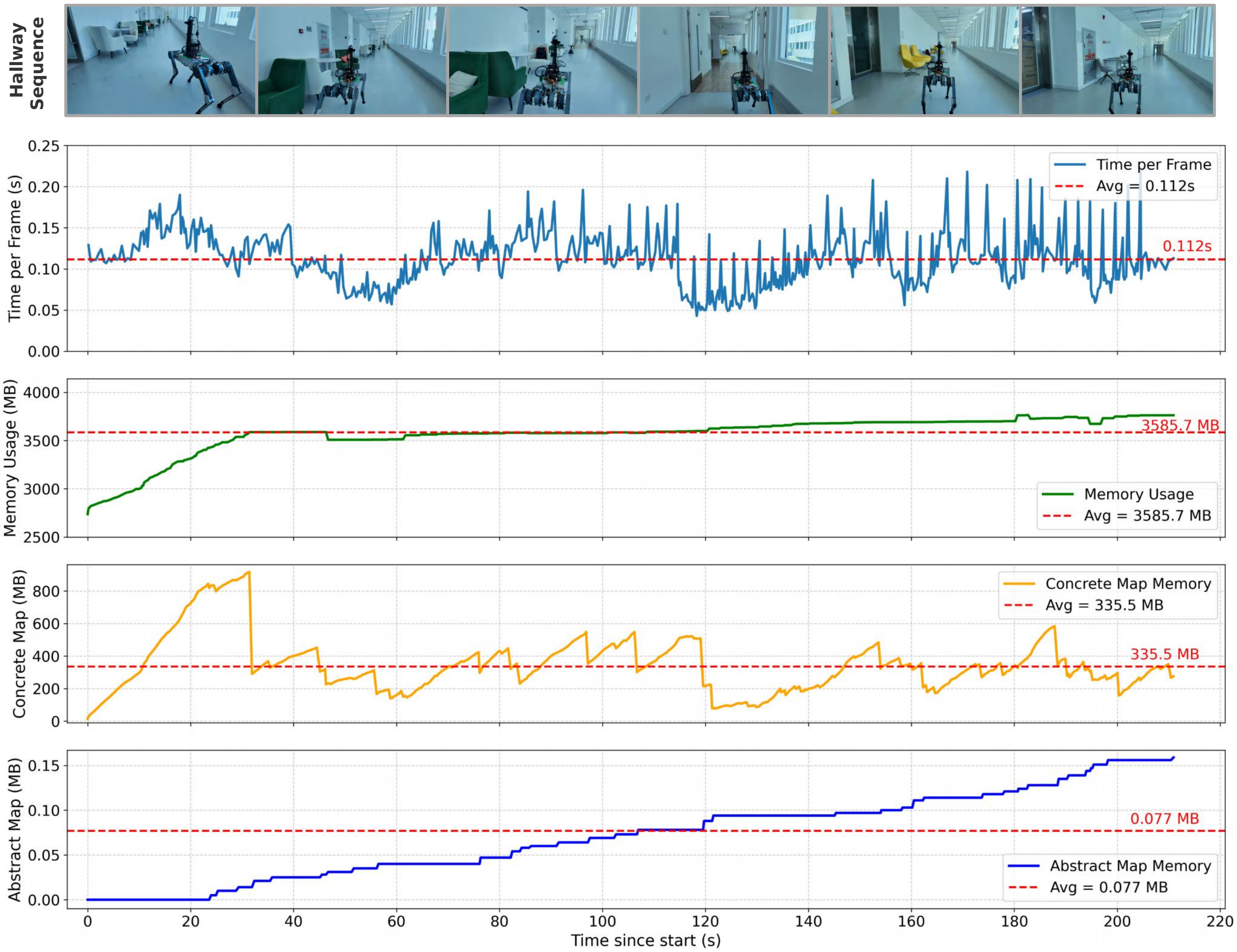


Fig. 13: Memory usage of concrete and abstract maps in a **hallway sequence** (212s).

consistent time cost in matching. Although the abstract map expands, its memory usage is minimal (only a few hundred KB). While the dual map maintains stable memory usage, the overall system memory gradually increases due to the cost of visualization. The detailed memory usage of concrete map and abstract map can be found in bottom two rows in Figure 13 and Figure 14. The testing results demonstrate that **DualMap scales well over time without ballooning in runtime or memory usage**.

Appendix VI
Acknowledgment

We would like to express our sincere gratitude to Pengxu Hou and Runze Yu for generously allowing us to use their experimental platform for the initial system validation. Special thanks to Guowei Huai, Qingyun Wang, Yingxi Lin, and Boyu Zhou for their valuable assistance during the real-world experiments. We also appreciate the insightful suggestions and feedback provided by Bonan Liu, Shibo Wang, Zhengmao He, and Handi Yin, which greatly contributed to this project.

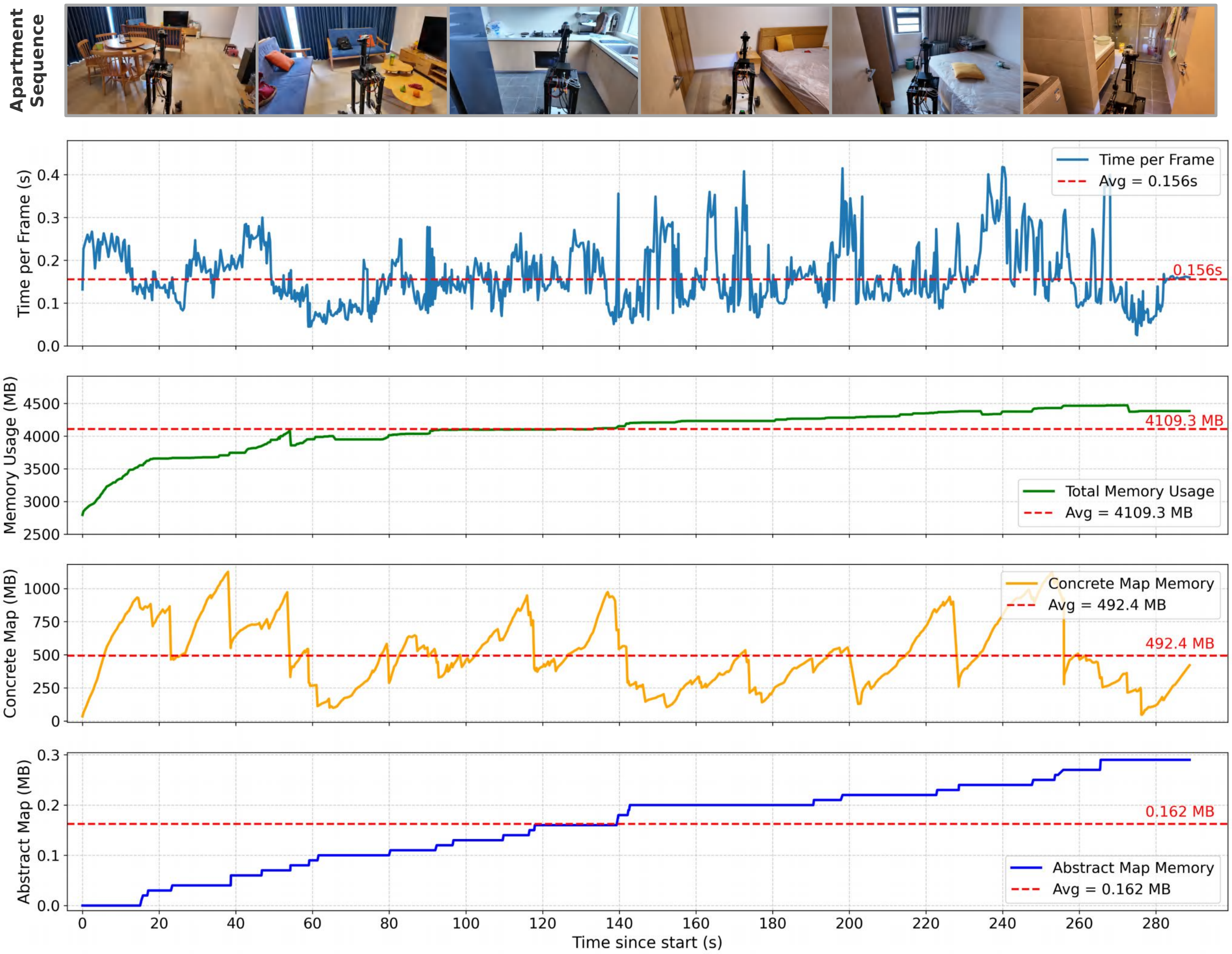


Fig. 14: Memory usage of concrete and abstract maps in an **apartment sequence** (286s).