LEG-SLAM: Language-Enhanced Gaussian Splatting for Real-Time SLAM

Anonymous ICCV submission

Paper ID 6330

Abstract

Modern Gaussian Splatting methods have proven highly effective for real-time photorealistic rendering of 3D scenes. However, integrating semantic information into this representation remains a significant challenge, especially in maintaining real-time performance for SLAM (Simultaneous Localization and Mapping) applications. In this work, we introduce LEG-SLAM — a novel approach that fuses an optimized Gaussian Splatting implementation with visual-language feature extraction using DINOv2 followed by learnable feature compressor based on Principal component analysis, while enabling an online dense SLAM. Our method simultaneously generates high-quality photorealistic images and semantically labeled scene maps, achieving real-time scene reconstruction with more than 10 fps on the Replica dataset and 18 fps on ScanNet. Experimental results show that our approach significantly outperforms state-of-the-art methods in reconstruction speed while achieving competitive rendering quality. The proposed system eliminates the need for prior data preparation such as camera's ego motion or pre-computed static semantic maps. With its potential applications in autonomous robotics, augmented reality, and other interactive domains, LEG-SLAM represents a significant step forward in real-time semantic 3D Gaussian-based SLAM. Project page: https://anonymousleg-slam.github.io/LEG-SLAM/

1. Introduction

In recent years, 3D Gaussian Splatting has become a popular method for representing and rendering novel views in 3D scenes, especially in applications that demand high performance and real-time processing. However, traditional approaches, which are primarily focused on photorealistic rendering, often fail short to simultaneously deliver high-quality semantic information—a critical requirement for applications in autonomous driving, augmented reality, and robotics.

By real-time, we mean a delay in the operation of the algorithm that does not exceed the period of receipt of the input sensory data. For example, for indoor robots, the frequency of image reception is typically 10 frames per second.

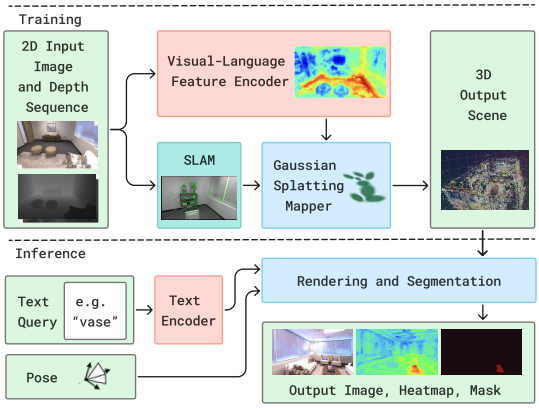


Figure 1. Simplified diagram of the proposed LEG-SLAM approach implementing real-time SLAM with language-enhanced Gaussian Splatting

Following these advantages of the 3D representation, recent works such as LangSplat [21], demonstrate the possibility of integrating language embeddings to support open-vocabulary queries. Despite its high semantic quality, LangSplat suffers from low performance in scene construction, limiting its real-time applicability. A similar approach, presented in [16], integrates semantic features into 3D Gaussian Splatting for accurate mapping and tracking; however, it operates in a closed-vocabulary mode and requires a pre-prepared semantic map, which reduces its adaptability for mapping unseen environments.

In this work, we present the first real-time SLAM (Simultaneous Localization and Mapping) method with visual-language features named **LEG-SLAM** (Language-Enhanced Gaussian Splatting SLAM). Our approach simultaneously renders photorealistic RGB images and creates accurate semantic masks, while executing a dense SLAM

system in real-time. To achieve this, we have developed an efficient implementation of Gaussian Splatting, integrated with a semantic extraction module based on DINOv2 [20] features from RGB images and Principal component analysis for feature compression. This approach significantly reduces processing time, saves memory and provides acceptable segmentation accuracy, making the system suitable for real-time operation.

One of the main advantages of our proposed method is the ability to construct a semantic representation of a 3D scene in real-time, without the need for prior data preparation. Unlike existing solutions that require static semantic maps or extensive pre-processing, our approach processes incoming RGB-D images on the fly, automatically extracting semantic features and building the 3D scene map.

Our main contributions:

- We developed an efficient Gaussian Splatting algorithm with original visual-language feature encoding based on DINOv2 and Principal Component Analysis (PCA).
- We implemented the first real-time SLAM with language features in an unified manner, enabling open-vocabulary segmentation without the need for a pre-prepared static semantic map. It allowed us to achieve up to 30 times faster performance than that of the analogues.
- We achieved real-time map construction for input images frame rate of 10 fps and competitive results in semantic segmentation quality on popular ScanNet and Replica datasets.

2. Related work

2.1. 3D Scene Representation Techniques

Historically, 3D scenes have been represented using point clouds, meshes, and voxels. Point clouds are collections of points with XYZ coordinates and optional attributes like color. Meshes use polygons, usually triangles, to model object surfaces. Voxels divide 3D space into cubic elements analogous to 2D pixels.

Recently, neural implicit representations such as Neural Radiance Fields (NeRF) [19] have gained popularity. NeRF models a continuous 3D radiance field using neural networks, enabling highly photorealistic scene reconstructions. However, it is computationally intensive. Extensions like Semantic-NeRF [29] and EditNeRF [17] add semantic labels and editing capabilities but remain impractical for real-time use.

On the other hand, 3D Gaussian Splatting (3DGS) [12] is a promising new representation that uses 3D Gaussians. It achieves photorealistic 1080p rendering at 60 FPS by leveraging point-based alpha blending and a differentiable tiled rasterizer. The explicit Gaussian parameterization also facilitates efficient scene editing. Dynamic 3D Gaussians [18] extends 3DGS to handle dynamic scenes. Other works [9, 28] generate 3DGS scenes from text and images.

2.2. Semantic 3D Scene Understanding

Large annotated datasets and multimodal models like CLIP [22] have advanced open-vocabulary semantic segmentation methods such as OpenSeg [4] and LSeg [14]. However, most operate on single images without ensuring multi-view consistency, limiting their utility for 3D scene analysis.

NeRF-based approaches like Semantic-NeRF [29] and Panoptic Lifting [23] incorporate semantic embeddings for enhanced 3D scene understanding but suffer from high computational complexity. Recent works [5, 16] extend 3DGS with semantic embeddings for open-vocabulary segmentation and reconstruction, but remain computationally demanding, which prevents the real-time use.

2.3. Neural Representations in SLAM

Neural representations have improved the reconstruction quality, speed, and semantic understanding of SLAM systems. SplaTAM [11] uses silhouette-guided optimization for better reconstruction. GS-SLAM [26] and GS-ICP [6] integrate adaptive Gaussian expansion and G-ICP pose tracking for accuracy and scalability.

Hybrid neural-coordinate representations address efficiency issues. Co-SLAM [25] combines multi-resolution hash grids and one-blob encoding for real-time global bundle adjustment without keyframes. E-SLAM [10] decodes multi-scale axis-aligned feature planes into TSDF and RGB, achieving state-of-the-art accuracy and speed. Photo-SLAM [7] connects explicit geometry and implicit photometric features via hyper primitives and Gaussian-Pyramid training, surpassing existing methods in quality and efficiency.

For semantic mapping, DNS SLAM [15] leverages 2D semantic priors and multi-view constraints. SNI-SLAM [32] refines scene representations through feature collaboration and one-way correlation decoding. 3D Gaussian frameworks like SGS-SLAM [16] incorporate semantic colored labels but may miss higher-level semantics. SemGauss-SLAM [31] and NEDS-SLAM [8] embed low-dimensional semantic features into 3D Gaussians but rely on 2D segmentation, limiting open-set robustness.

In summary, neural representations have revolutionized 3D scene understanding and SLAM. Gaussian-based techniques show great promise for efficient, high-quality, and semantically-aware scene representation. However, further research is needed to develop real-time open-vocabulary understanding while preserving multi-view consistency and scalability.

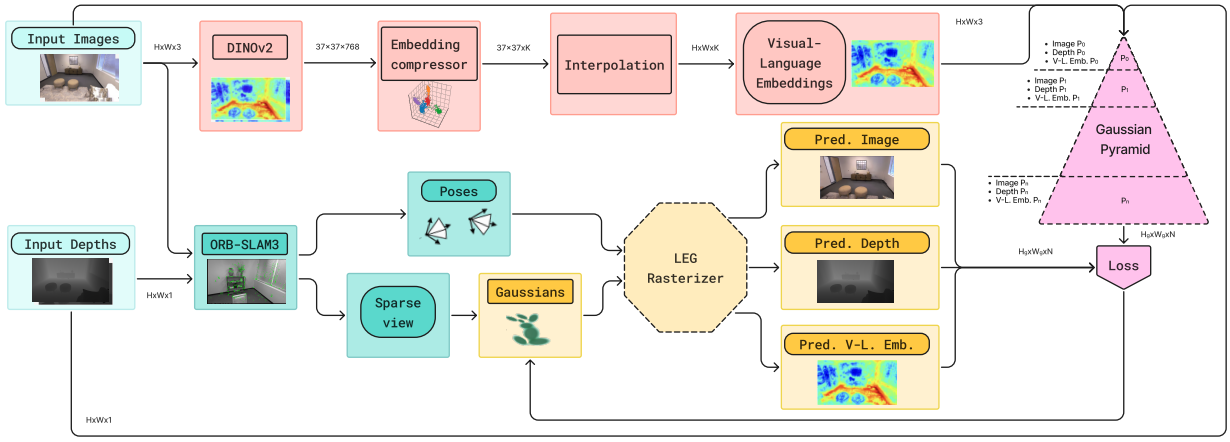


Figure 2. A detailed scheme of the developed LEG-SLAM method. Its distinctive features include a fast approach to obtaining visual-linguistic features, a computationally efficient SLAM implementation, and a learning-based approach to a language-enhanced rasterizer for Gaussian Splatting.

3. Methodology

3.1. Proposed approach

Unlike traditional approaches, where RGB image rendering and semantic segmentation are performed separately, our method integrates these processes into a unified pipeline, ensuring consistency between the visual and semantic representation of the 3D scene. Figure 2 provides an overview of the proposed method. At the core of our approach lies the Gaussian Splatting method, which is used for efficient 3D scene representation and rendering. To extract visual features, we employ the DINOv2 model [20], which generates a compact embedding map characterizing local scene features. These embeddings are then compressed using an autoencoder, significantly reducing data size and accelerating subsequent processing.

The integration of text queries into the semantic segmentation process is performed using the Talk2DINO module [1], which converts text embeddings from CLIP [22] space into the DINOv2 [20] embedding space. This process is applied to an already reconstructed 3D scene, enabling open-vocabulary queries, where the semantic features extracted from the image are matched with textual descriptions in real time.

By combining all these modules into a single rasterization-based pipeline, our method enables the simultaneous output of high-quality RGB images and semantic masks, a crucial feature for real-time SLAM systems where minimizing latency is essential. The computational efficiency of our approach is detailed in Table 1, which presents the average processing time per frame across different resolutions. The results highlight the scalability of our method, demonstrating that even at higher resolutions, the total processing time remains within real-time constraints.

Processing Time Per Frame (ms)			
Processing Stage	**640×480**	**986×728**	**1200×680**
Image preprocessing	8.72	12.36	14.57
Feature extraction	31.91	40.38	48.33
Feature compression	1.57	2.14	2.36
Tracking + Mapping	14.82	26.85	29.95
Average processing time	57.02	78.65	95.21

Table 1. Average processing time for each pipeline stage per frame at different resolutions.

The implementation details of each component are provided in the following sections. Section 3.2 describes the process of obtaining visual features using DINOv2 [20], while Section 3.3 explains the embedding compression step using the autoencoder. The distillation process of semantic information into the scene representation is detailed in Section 3.4, and the procedure for generating semantic queries using Talk2DINO [1] is covered in Section 3.5.

3.2. Visual features generation

To extract visual features, our system utilizes a pretrained DINOv2 [20] model, which takes an input image of size 518×518 pixels and produces an embedding map of size $37 \times 37 \times 768$. In this map, each of the 37×37 cells corresponds to a local scene fragment, while the 768-dimensional vector represents a high-level description of its visual characteristics.

For further processing, the embedding map is passed to the next system module without resizing. This approach

allows working with a compact scene representation, reducing computational costs. To decrease the dimensionality of embeddings, an encoder compresses them from 768 to K dimensions. This significantly reduces computational complexity during subsequent processing while maintaining most of the essential information. The details of this compression are discussed in the next section.

Using DINOv2 enables the formation of an informative semantic representation, which is then distilled into the Gaussian Splatting pipeline. This ensures high segmentation quality while maintaining real-time processing capabilities.

3.3. Embedding Compressor

To optimize the processing of embeddings generated by the DINOv2 [20] model, our system employs a Principal Component Analysis (PCA)-based compressor. This module reduces the dimensionality of the embedding map from $37 \times 37 \times 768$ to $37 \times 37 \times K$, significantly lowering computational costs while preserving essential semantic information.

The PCA model was trained using 1,000 ImageNet text classes. For each class, a text embedding was obtained via CLIP [22], projected into the DINOv2 [20] latent space using Talk2DINO [1], and augmented with noise while maintaining a cosine similarity of at least 0.95 with the original. This strategy allows PCA to efficiently compress embeddings beyond ImageNet classes by learning a general feature projection rather than specific class representations.

After compression, the embedding map is upsampled to the original image resolution using bilinear interpolation, ensuring spatial consistency with the input image. The resulting $H \times W \times K$ representation is then used in the Gaussian Splatting pipeline for joint rendering of RGB images and semantic maps. This approach enables real-time performance while preserving segmentation accuracy.

3.4. Integration of Semantic Features and Gaussian Splatting

In our approach, semantic features are incorporated to the Gaussian splatting pipeline for simultaneous rendering of RGB images and semantic masks. This allows to effectively combine color and semantic information in a scene, ensuring consistency of representations and high performance.

Semantic vector representing local features from the semantic embedding map is added to the initial parameters of each Gaussian. This semantic vector is optimized simultaneously with gaussian parameters (such as position, scale, orientation and color), which allows to obtain an accurate match between visual and semantic data.

The optimization of semantic features vector includes the following steps:

1. **Initialization of semantic vectors:** During the initial initialization of the gaussian, the semantic vector $H \times W \times K$ is assigned zero values. If the gaussian is obtained by splitting of an existing gaussian, then the semantic features are copied.
2. **Parameter optimization:** During the training process, the gaussian parameters are optimized by minimizing the error between the rendered and the reference 2d representations. Simultaneous training on a single gaussian cloud ensures consistency of visual and semantic data.
3. **Simultaneous rendering of RGB and semantic masks:** The resulting gaussian parameters are used to simultaneously form a photorealistic image of a scene and corresponding semantic masks, where each gaussian contributes to both representations.

Using a single pipeline for visual and semantic data can significantly reduce computational costs, while ensuring high accuracy and speed of reconstruction.

The loss function includes three terms. The first of them $\mathcal{L}_{color}$ is the standard loss function for color features from the original 3DGS method. The second term $\mathcal{L}_{cos_sim}$ is responsible for calculating the cosine similarity between the rendered semantic feature map and the one obtained from DINOv2. The third term $\mathcal{L}_1$ optimizes the depth value. The inputs of the function are the rendered image I_{pred}, the language feature map L_{pred} and the depth D_{pred}, as well as the gt image I_{gt}, the depth D_{gt} and the compressed features DinoV2 L_{gt}.

$$\begin{aligned}\mathcal{L}(I_{pred}, L_{pred}, D_{pred}; I_{gt}, L_{gt}, D_{gt}) = \\ = \underbrace{(1-\lambda)\mathcal{L}_1(I_{pred}, I_{gt}) + \lambda\mathcal{L}_{ssim}(I_{pred}, I_{gt})}_{\mathcal{L}_{color}} + \\ + \mathcal{L}_{cos_sim}(L_{pred}, L_{gt}) + \\ + \mathcal{L}_1(D_{pred}, D_{gt}). \quad (1)\end{aligned}$$

To speed up and improve the quality of reconstruction, optimization of gaussians is performed on input images which resolution is consistently increasing ($H_g \times W_g$). The resolution of the input images is reduced by using a Gaussian pyramid. This approach allows to quickly set fairly accurate parameters for gaussians, which ensures fast convergence on large-dimensional images.

3.5. Segmentation by Text Query Using DINOv2 Features

To perform segmentation based on a text query, our system employs the Talk2DINO model, which converts text embeddings from CLIP space into the embedding space of DINOv2. This transformation is necessary because CLIP and DINOv2 embeddings are trained on different tasks and exist in distinct vector spaces. Talk2DINO bridges this gap, allowing textual descriptions to be linked with visual fea-

tures of the scene, enabling accurate open-vocabulary segmentation.

In our approach, an embedding map of size $H \times W \times K$ is created by projecting the reconstructed 3D scene onto the current camera viewpoint. This projection is computed based on the contributions of all Gaussians in the scene to each pixel of the image.

The segmentation process consists of several steps:

1. **Text Embedding Generation:** The text query is converted into an embedding of size 1×768 using the CLIP model.
2. **Feature Space Alignment:** Talk2DINO transforms this embedding into the DINOv2 feature space, producing a vector of size 1×768.
3. **Dimensionality Reduction:** PCA is applied to compress the text embedding to $1 \times K$, aligning it with the visual embeddings from the $H \times W \times K$ map. This reduces computational overhead during similarity comparison.
4. **Semantic Similarity Computation:** The embedding map $H \times W \times K$, generated via scene projection, is used to compute a similarity map with the text query. This is achieved by performing a dot product between each scene embedding and the compressed text embedding. The result is a semantic similarity map of size $H \times W \times 1$.

By projecting the reconstructed scene through Gaussians onto a 2D plane, the system generates an up-to-date semantic feature map for each frame, which is then used to segment objects based on the query.

4. Experiments

In this section, we conduct a series of experiments to evaluate the effectiveness of LEG-SLAM in the task of open-vocabulary 3D scene reconstruction with semantic understanding. We analyze the method's performance on various datasets and compare it with existing approaches.

First, we evaluate the quality of 3D reconstruction on the Replica dataset. Then, we assess semantic segmentation accuracy on ScanNet dataset, demonstrating open-vocabulary scene understanding. Next, we analyze SLAM performance on Replica, examining the accuracy of camera trajectory estimation and tracking speed. Additionally, we investigate the impact of different factors, including embedding compression methods, visual feature extraction architectures, and rendering strategies.

Finally, we compare LEG-SLAM with existing methods in terms of 3D reconstruction quality, semantic segmentation accuracy, and computational efficiency, highlighting its advantages for real-time language-enhanced SLAM.

4.1. Experimental Setup

4.1.1. Datasets

To evaluate the quality of scene reconstruction and semantic segmentation in an open-vocabulary setting, we use several widely adopted datasets in 3D Gaussian Splatting research.

Replica [24] is a high-quality synthetic dataset containing realistic 3D-rendered indoor environments. It is widely used for evaluating 3D reconstruction methods, as it provides scenes with detailed geometry and textures. In our experiments, Replica serves as a benchmark for assessing the accuracy of scene reconstruction.

ScanNet [3] is an RGB-D dataset consisting of real-world scanned indoor scenes with annotated 3D camera poses and 2D object class labels. Due to the presence of semantic annotations, it is well-suited for evaluating semantic segmentation performance. Our experiments are conducted on 12 validation scenes from ScanNet.

Each of these datasets plays a distinct role in our evaluation. While Replica provides a controlled environment for assessing reconstruction quality, ScanNet allows us to analyze both reconstruction and semantic segmentation in real-world indoor scenes. This diverse selection ensures a comprehensive assessment of LEG-SLAM across multiple challenging scenarios, covering both 3D reconstruction and semantic scene understanding.

4.1.2. Implementation Details

All experiments were conducted on an NVIDIA RTX 4090 GPU with 24 GB of video memory. We use DINOv2 ViT-B/14 as the backbone for extracting visual features and CLIP ViT-B/16 for processing text queries. To project textual embeddings into the visual feature space, we employ the Talk2DINO module.

4.2. Quantitative Analysis

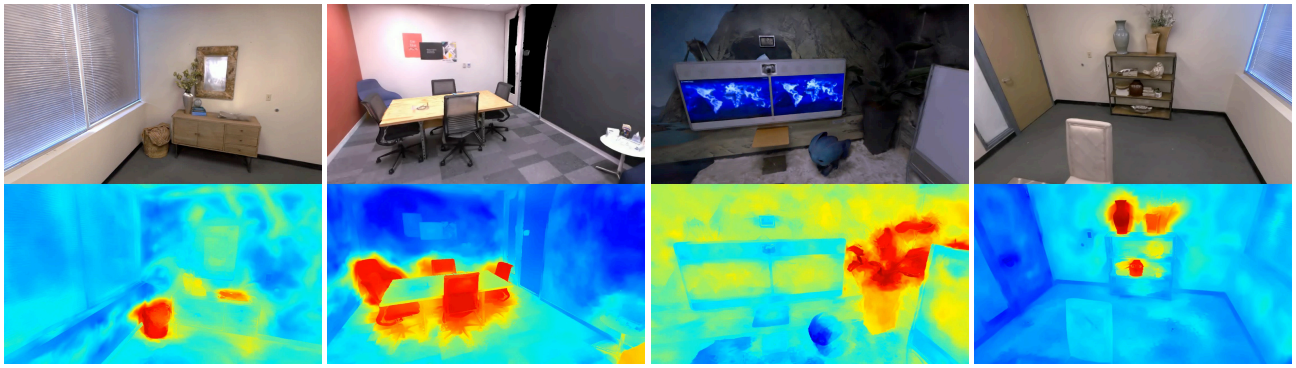
To further illustrate the effectiveness of our method in open-vocabulary semantic understanding, Figure 3 presents RGB renderings and corresponding semantic heatmaps for various text queries across different scenes. Each image showcases how LEG-SLAM successfully localizes and highlights queried objects, even for ambiguous or spatially complex environments.

For instance, Figure 3a demonstrates accurate identification of a "basket", while figure 3b highlights the correct localization of a "chair". Figure 3c showcases the method's ability to recognize a "plant", where fine-grained textures and occlusions make segmentation challenging. Finally, Figure 3d presents a successful retrieval of a "vase", capturing both its geometric structure and material properties.

These qualitative results confirm that LEG-SLAM not only provides high-fidelity 3D reconstructions but also accurately aligns semantic information with the rendered en-

Method	ATE RMSE ↓	FPS ↑	PSNR (dB) ↑	SSIM ↑	LPIPS ↓	Semantics
NICE-SLAM [33]	2.51	0.198	24.42	0.809	0.233	-
Vox-Fusion [27]	1.47	1.07	24.41	0.801	0.236	-
Co-SLAM [25]	1.06	6.41	30.24	0.939	0.252	-
ESLAM [10]	0.62	3.02	29.08	0.929	0.239	-
SplaTAM [11]	**0.41**	0.21	**34.11**	**0.968**	**0.102**	-
Photo-SLAM [7]	1.24	27.35	30.91	0.883	0.143	-
SNI-SLAM [32]	0.46	2.15	29.43	0.935	0.235	Closed-Vocabulary
LEG-SLAM (Ours)	0.94	**10.56**	32.12	0.9151	0.1456	**Open-Vocabulary**

Table 2. Quantitative comparison of SLAM accuracy, speed, rendering quality, and semantic capabilities for LEG-SLAM and other SLAM methods. The results are averaged over 8 scenes from the Replica dataset.



(a) Text query: "basket" (b) Text query: "chair" (c) Text query: "plant" (d) Text query: "vase"

Figure 3. Visualization of RGB and semantic heatmap outputs for different text queries. Each image contains both an RGB rendering (top) and a semantic heatmap (bottom) highlighting regions matching the given query.

vironment, making it well-suited for open-vocabulary object localization in real-time applications.

4.3. Scene Reconstruction

To evaluate the quality of 3D reconstruction, we conducted experiments on the Replica dataset. Table 2 presents the results of our method compared to existing SLAM approaches.

Our method achieves competitive reconstruction quality, slightly trailing behind SplaTAM in terms of PSNR and SSIM. This difference arises from the additional semantic rendering in our approach, which introduces constraints on the optimization process. Despite this, our method maintains strong perceptual quality, as reflected in LPIPS scores, demonstrating its effectiveness in jointly reconstructing geometry and semantics in real-time applications.

4.4. Open-Vocabulary Semantic Segmentation

We compare LEG-SLAM against state-of-the-art methods on the ScanNet dataset. The segmentation quality is evaluated using mIoU and mAcc metrics.

Table 3 presents a comparative analysis of NeRF and Gaussian Splatting-based methods, reporting the average segmentation metrics and training time across 12 ScanNet scenes. Figure 4 visually illustrates the segmentation results for different methods.

Method	Backbone	mIoU, %	mAcc, %	Training Time
LERF [13]	NeRF+CLIP	31.2	61.7	45 min
PVLFF [2]	NeRF+LSeg	52.9	67.0	65 min
LangSplat [21]	3DGS+CLIP	24.7	42.0	180 min
Feature3DGS [30]	3DGS+LSeg	59.2	75.1	150 min
Semantic Gaussians [5]	3DGS+LSeg	**62.0**	**77.0**	90 min
LEG-SLAM	3DGS+DINOv2	41.4	74.3	**1.5 min**

Table 3. Comparison of Open-Vocabulary segmentation methods on 12 Scenes of ScanNet (Average 1800 Frames per Scene)

Results show that while LEG-SLAM does not achieve the highest mIoU, it significantly outperforms competing methods in runtime efficiency, running at 18 FPS on ScanNet. On average, it processes a scene in 1.5 minutes, whereas other methods require tens of minutes or even hours per scene.

4.5. SLAM Comparison

We evaluate the performance of LEG-SLAM in the SLAM task by comparing it with existing state-of-the-art methods on the Replica dataset. The primary metrics include ATE RMSE for trajectory accuracy and rendering quality. Addi-

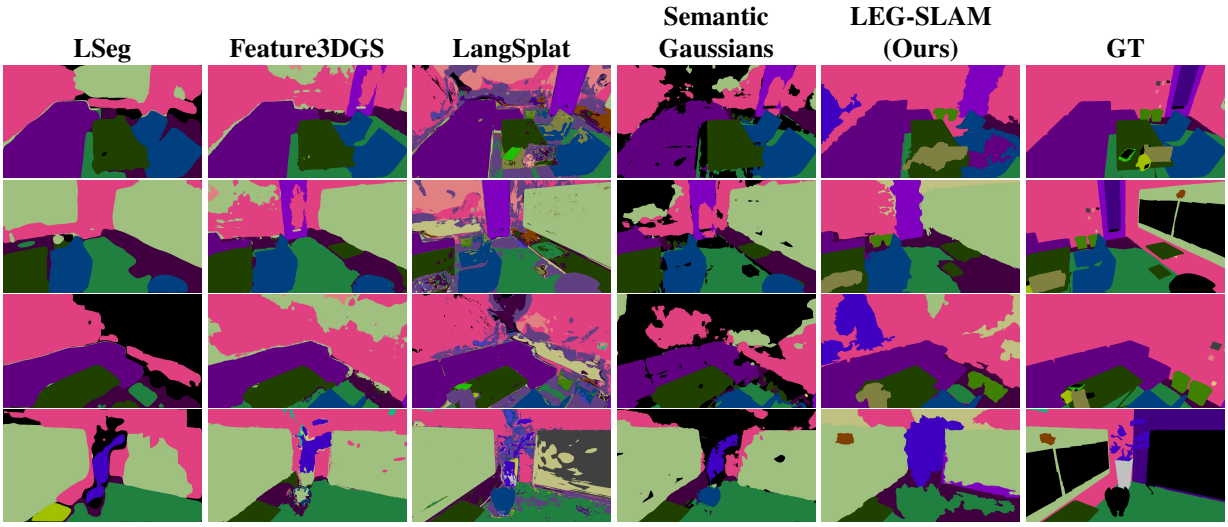


Figure 4. Comparison of semantic segmentation performance across different pipelines on the Replica dataset, including ground truth for reference.

tionally, we emphasize the system's speed, which is critical for real-time applications.

As shown in Table 2, LEG-SLAM achieves competitive reconstruction accuracy while significantly outperforming other methods in terms of speed, processing frames at 10 FPS. While SplaTAM yields slightly better reconstruction and rendering quality, it does not support real-time performance. The added complexity of optimizing 3D Gaussians for both geometric and semantic consistency introduces minor reconstruction trade-offs but enables open-vocabulary semantic segmentation, making LEG-SLAM the only method in this comparison capable of integrating real-time SLAM with open-vocabulary scene understanding.

4.6. Ablation studies

4.6.1. Comparison of Different Feature Extraction Backbones

Selecting an efficient feature extraction method is crucial for real-time performance. We compare two architectures: DINOv2 ViT-B/14 and OpenSeg. Both provide high-quality visual features but differ in computational efficiency and adaptability to open-vocabulary tasks.

DINOv2 is a self-supervised model trained to produce generalizable visual embeddings without relying on predefined categories. This makes it well-suited for diverse environments, where it maintains high feature informativeness. In contrast, OpenSeg is trained on COCO categories and performs well in segmentation but struggles with generalization in open-vocabulary settings.

To evaluate computational efficiency, we measured processing time, including embedding compression using PCA. Results show that OpenSeg + PCA takes 100 ms, while DINOv2 + PCA completes the same operation in 33 ms — a 3× speedup. Given the constraints of real-time operation, this significant reduction in latency makes DINOv2 the preferred choice, offering a balance between feature quality and computational cost.

Method	CosSim ↑	mIoU ↑	Time (ms) ↓
MLP-AE (2-layer)	0.9379	37.3	0.85
MLP-AE (4-layer)	0.9551	34.8	2.96
PCA	**0.9570**	**41.2**	**0.60**

Table 4. Comparison of embedding compression methods (all methods compress to 64 dimensions)

4.6.2. Influence of Embedding Compression Methods

Efficient embedding compression is crucial for real-time applications. Table 4 compares PCA and two MLP-based Autoencoders (MLP-AE), all compressing embeddings from 768 to 64 dimensions.

PCA achieves the best trade-off between accuracy and efficiency, yielding the highest cosine similarity and segmentation quality while maintaining the fastest inference time. The 2-layer MLP-AE shows a noticeable drop in segmentation accuracy, while the deeper 4-layer variant improves reconstruction quality but suffers from significantly higher computational cost. Despite its deeper architecture, the 4-layer model does not outperform PCA, making it less suitable for real-time deployment.

Given these results, PCA emerges as the most effective choice, offering a superior balance between accuracy, computational speed, and implementation simplicity.

Dim	Cos. Sim.↑	mIoU↑	PSNR↑	Train FPS↑
3	0.7423	6.6	24.34	18.38
6	0.8348	20.1	**25.07**	**18.81**
9	0.8814	27.2	24.83	18.72
12	0.9070	32.0	24.53	18.63
16	0.9298	37.1	24.7	18.4
24	0.9532	39.1	24.75	18.22
32	0.9662	40.1	24.48	18.09
48	0.9808	40.4	24.30	17.52
64	0.9880	41.4	24.26	17.55
80	0.9920	42.2	23.96	17.63
96	0.9946	42.3	23.65	17.58
128	**0.9973**	**42.7**	23.57	17.44

Table 5. Results of PCA-based embedding compression experiments

4.6.3. Impact of embedding dimensionality

To determine the optimal dimensionality for PCA-compressed embeddings, we conducted two separate evaluations: compression quality analysis using Cosine Similarity and full pipeline performance measured by mIoU, PSNR, and training speed in FPS.

For compression quality assessment, we randomly selected 10 frames from each of the 12 ScanNet validation scenes and extracted DINOv2 embeddings at the original resolution of $648 \times 484 \times 768$. To prevent redundancy from neighboring pixels, embeddings were sampled with a stride of 4, ensuring a diverse set of semantic features. PCA was then applied with embedding dimensionalities ranging from 3 to 128, and reconstruction quality was assessed using Cosine Similarity.

To evaluate the overall impact on pipeline performance, the full LEG-SLAM pipeline was executed on complete ScanNet scenes, where segmentation quality was measured using mIoU, PSNR, and training FPS.

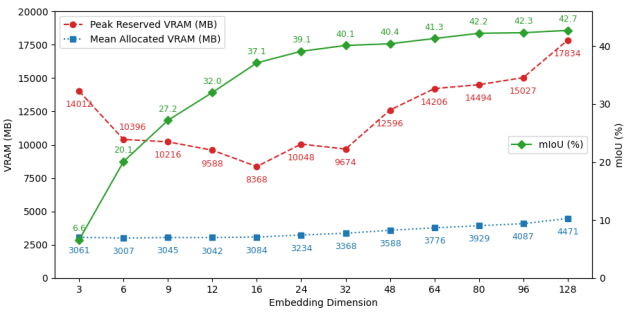


Figure 5. VRAM consumption across different feature compressor embedding dimensions

The results indicate that increasing the embedding dimensionality improves Cosine Similarity. However, after reaching 64 dimensions, these improvements become negligible, while processing time continues to rise. Additionally, larger embedding sizes introduce increased VRAM consumption, complexity in optimizing Gaussian parameters, leading to a decline in PSNR.

Interestingly, at an embedding dimensionality of 3, compression is highly lossy, resulting in poor reconstruction of feature embeddings. This leads to ineffective optimization of Gaussian parameters, causing an overgeneration of Gaussians as the pipeline attempts to compensate for lost information. This behavior significantly increases VRAM consumption (as shown in Figure 5) without yielding meaningful improvements in scene reconstruction or segmentation quality.

The selected dimensionality of 64 provides a favorable balance, preserving essential semantic information while maintaining efficient optimization and reconstruction quality. It ensures that feature embeddings retain sufficient expressiveness without unnecessary computational overhead.

5. Conclusion

We introduced LEG-SLAM, a novel approach that combines 3D Gaussian Splatting with open-vocabulary semantic understanding, enabling real-time 3D reconstruction and interactive scene interpretation based on textual queries. Unlike existing methods, our approach does not rely on predefined object categories and provides efficient integration of semantic and geometric information.

Experiments demonstrate that LEG-SLAM achieves competitive reconstruction quality on the Replica dataset while maintaining high-speed performance for semantic segmentation on ScanNet. The use of DINOv2 for feature extraction and PCA for embedding compression ensures computational efficiency without significant loss of accuracy. However, integrating semantic information during optimization slightly affects reconstruction quality due to additional constraints in Gaussian optimization.

Furthermore, LEG-SLAM significantly accelerates open-vocabulary Gaussian Splatting-based segmentation. Compared to LERF, our method achieves a 30× speedup, making real-time interactive scene understanding feasible for practical applications.

Despite minor trade-offs in reconstruction quality due to semantic constraints, LEG-SLAM offers a unique balance between speed, adaptability, and real-time capability, making it a promising solution for future 3D scene understanding tasks.

References

[1] Luca Barsellotti, Lorenzo Bianchi, Nicola Messina, Fabio Carrara, Marcella Cornia, Lorenzo Baraldi, Fabrizio Falchi, and Rita Cucchiara. Talking to DINO: Bridging Self-Supervised Vision Backbones with Language for Open-Vocabulary Segmentation, 2024. arXiv:2411.19331 [cs]. 3, 4

[2] Haoran Chen, Kenneth Blomqvist, Francesco Milano, and Roland Siegwart. Panoptic Vision-Language Feature Fields. *IEEE Robotics and Automation Letters*, 9(3):2144–2151, 2024. arXiv:2309.05448 [cs]. 6

[3] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017. 5

[4] Golnaz Ghiasi, Xiuye Gu, Yin Cui, and Tsung-Yi Lin. Scaling open-vocabulary image segmentation with image-level labels. In *European conference on computer vision*, pages 540–557. Springer, 2022. 2

[5] Jun Guo, Xiaojian Ma, Yue Fan, Huaping Liu, and Qing Li. Semantic Gaussians: Open-Vocabulary Scene Understanding with 3D Gaussian Splatting, 2024. arXiv:2403.15624 [cs]. 2, 6

[6] Seongbo Ha, Jiung Yeon, and Hyeonwoo Yu. Rgbd gs-icp slam, 2024. 2

[7] Huajian Huang, Longwei Li, Hui Cheng, and Sai-Kit Yeung. Photo-SLAM: Real-time Simultaneous Localization and Photorealistic Mapping for Monocular Stereo and RGB-D Cameras. In *Photo-SLAM: Real-time Simultaneous Localization and Photorealistic Mapping for Monocular Stereo and RGB-D Cameras*, pages 21584–21593, 2024. 2, 6

[8] Yiming Ji, Yang Liu, Guanghu Xie, Boyu Ma, Zongwu Xie, and Hong Liu. Neds-slam: A neural explicit dense semantic slam framework using 3d gaussian splatting. *IEEE Robotics and Automation Letters*, 9(10):8778–8785, 2024. 2

[9] Lutao Jiang, Xu Zheng, Yuanhuiyi Lyu, Jiazhou Zhou, and Lin Wang. Brightdreamer: Generic 3d gaussian generative framework for fast text-to-3d synthesis, 2024. 2

[10] Mohammad Mahdi Johari, Camilla Carta, and François Fleuret. Eslam: Efficient dense slam system based on hybrid representation of signed distance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17408–17419, 2023. 2, 6

[11] Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. Splatam: Splat track & map 3d gaussians for dense rgb-d slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21357–21366, 2024. 2, 6

[12] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics*, 42(4):1–14, 2023. 2

[13] Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. Lerf: Language embedded radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19729–19739, 2023. 6

[14] Boyi Li, Kilian Q Weinberger, Serge Belongie, Vladlen Koltun, and Rene Ranftl. Language-driven semantic segmentation. In *International Conference on Learning Representations*, 2022. 2

[15] Kunyi Li, Michael Niemeyer, Nassir Navab, and Federico Tombari. Dns slam: Dense neural semantic-informed slam, 2023. 2

[16] Mingrui Li, Shuhong Liu, Heng Zhou, Guohao Zhu, Na Cheng, Tianchen Deng, and Hongyu Wang. Sgs-slam: Semantic gaussian splatting for neural dense slam. In *European Conference on Computer Vision*, pages 163–179. Springer, 2024. 1, 2

[17] Steven Liu, Xiuming Zhang, Zhoutong Zhang, Richard Zhang, Jun-Yan Zhu, and Bryan Russell. Editing Conditional Radiance Fields. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5753–5763, Montreal, QC, Canada, 2021. IEEE. 2

[18] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *2024 International Conference on 3D Vision (3DV)*, pages 800–809. IEEE, 2024. 2

[19] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 2

[20] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research Journal*, pages 1–31, 2024. 2, 3, 4

[21] Minghan Qin, Wanhua Li, Jiawei Zhou, Haoqian Wang, and Hanspeter Pfister. Langsplat: 3d language gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20051–20060, 2024. 1, 6

[22] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision, 2021. arXiv:2103.00020 [cs]. 2, 3, 4

[23] Yawar Siddiqui, Lorenzo Porzi, Samuel Rota Buló, Norman Müller, Matthias Nießner, Angela Dai, and Peter Kontschieder. Panoptic lifting for 3d scene understanding with neural fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9043–9052, 2023. 2

[24] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica Dataset: A Digital Replica of Indoor Spaces, 2019. arXiv:1906.05797 [cs]. 5

[25] Hengyi Wang, Jingwen Wang, and Lourdes Agapito. Co-slam: Joint coordinate and sparse parametric encodings for

neural real-time slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13293–13302, 2023. 2, 6

[26] Chi Yan, Delin Qu, Dan Xu, Bin Zhao, Zhigang Wang, Dong Wang, and Xuelong Li. Gs-slam: Dense visual slam with 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19595–19604, 2024. 2

[27] Xingrui Yang, Hai Li, Hongjia Zhai, Yuhang Ming, Yuqian Liu, and Guofeng Zhang. Vox-Fusion: Dense Tracking and Mapping with Voxel-based Neural Implicit Representation. In *2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 499–507, 2022. arXiv:2210.15858 [cs]. 6

[28] Taoran Yi, Jiemin Fang, Junjie Wang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. Gaussiandreamer: Fast generation from text to 3d gaussians by bridging 2d and 3d diffusion models, 2024. 2

[29] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew J Davison. In-place scene labelling and understanding with implicit scene representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15838–15847, 2021. 2

[30] Shijie Zhou, Haoran Chang, Sicheng Jiang, Zhiwen Fan, Zehao Zhu, Dejia Xu, Pradyumna Chari, Suya You, Zhangyang Wang, and Achuta Kadambi. Feature 3dgs: Supercharging 3d gaussian splatting to enable distilled feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21676–21685, 2024. 6

[31] Siting Zhu, Renjie Qin, Guangming Wang, Jiuming Liu, and Hesheng Wang. Semgauss-slam: Dense semantic gaussian splatting slam, 2024. 2

[32] Siting Zhu, Guangming Wang, Hermann Blum, Jiuming Liu, Liang Song, Marc Pollefeys, and Hesheng Wang. Sni-slam: Semantic neural implicit slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21167–21177, 2024. 2, 6

[33] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12786–12796, 2022. 6

LEG-SLAM: Language-Enhanced Gaussian Splatting for Real-Time SLAM

Supplementary Material

1. ScanNet dataset

To further analyze the performance of LEG-SLAM, we evaluate the reconstruction and semantic segmentation metrics on individual ScanNet scenes. Table 1 presents per-scene results, where PSNR, SSIM, and LPIPS measure reconstruction quality, while mIoU and mAcc assess semantic segmentation accuracy.

For semantic segmentation, LEG-SLAM operates in an open-vocabulary setting, but to quantitatively evaluate performance, we align with the standard ScanNet benchmark. The evaluation uses 20 semantic categories, which are detailed in Table 2, alongside their per-class IoU scores.

To ensure effective feature compression, the same 20 categories were used to train the PCA encoder for ScanNet dataset evaluation. This alignment allows PCA to learn the optimal projection for retaining key semantic information while reducing computational complexity. However, since PCA learns a generalized compression strategy rather than memorizing specific class distributions, our method remains effective even on previously unseen categories. While segmentation quality is more consistent on trained categories, LEG-SLAM successfully extends to open-vocabulary scenarios, making it applicable across diverse 3D environments.

Scene	PSNR (dB)	SSIM	LPIPS	mIoU	mAcc
0050_02	21.51	0.6963	0.3399	0.3157	0.5474
0144_01	25.23	0.8167	0.1914	0.5388	0.6758
0221_01	23.69	0.7314	0.3868	0.4939	0.6852
0300_01	26.66	0.8410	0.2041	0.5340	0.6204
0354_00	20.98	0.8307	0.2048	0.5554	0.7378
0389_00	26.51	0.8302	0.2183	0.5382	0.8238
0423_02	23.14	0.7491	0.2386	0.4499	0.4633
0427_00	25.41	0.8080	0.2163	0.4985	0.6413
0494_00	24.96	0.8264	0.1902	0.5872	0.7972
0616_00	19.03	0.7386	0.3031	0.5872	0.8007
0645_02	22.03	0.7797	0.2892	0.4229	0.7224
0693_00	27.90	0.8394	0.1870	0.4115	0.7032

Table 1. Quantitative evaluation on ScanNet scenes

2. Replica dataset

To contextualize LEG-SLAM's performance, we compare it against existing SLAM methods on different Replica scenes. Table 3 summarizes the results, providing a detailed breakdown of reconstruction quality across multiple environments. The table reports PSNR, SSIM, and LPIPS metrics, averaged over all scenes and shown individually for each tested environment.

Class	IoU
Wall	0.364
Floor	0.729
Cabinet	0.198
Bed	0.759
Chair	0.497
Sofa	0.275
Table	0.577
Door	0.450
Window	0.441
Shelves	0.521
Counter	not presented
Curtain	0.626
Ceiling	0.338
Refrigerator	0.385
Television	0.394
Person	not presented
Toilet	0.471
Sink	0.034
Lamp	0.219
Bag	0.181
Mean IoU	**0.414**
Mean Accuracy	**0.743**

Table 2. Per-class Intersection over Union (IoU) scores for ScanNet dataset. Some classes (*counter*, *person*) were not evaluated due to annotation errors.

3. Impact of Embedding Dimensionality on Semantic Quality

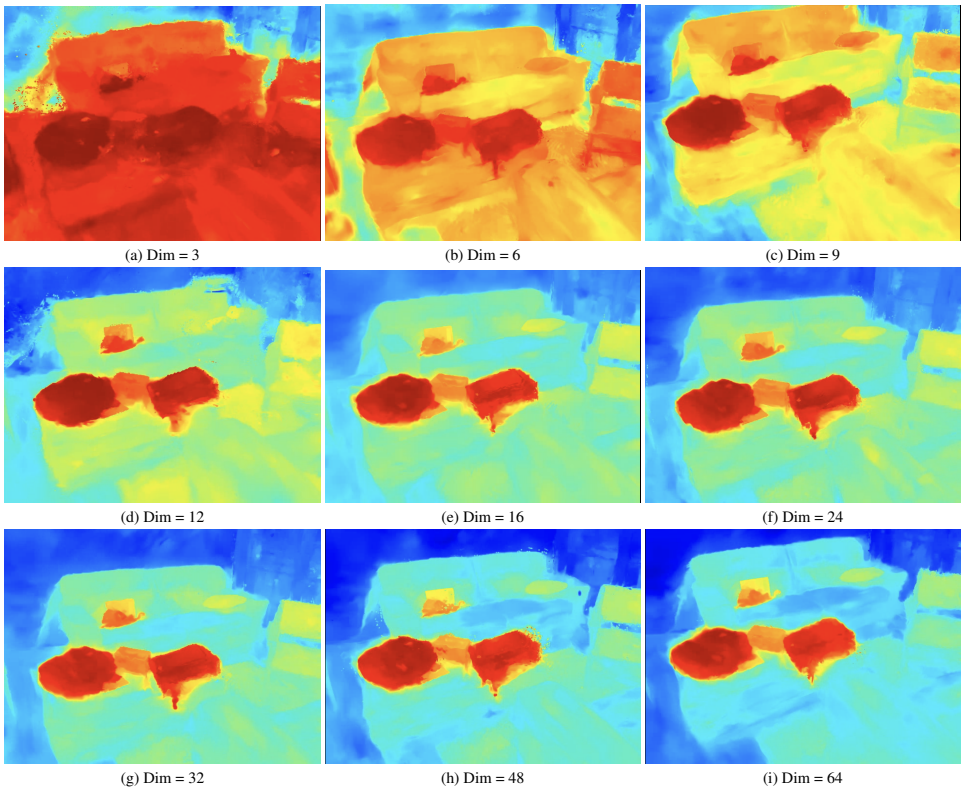
To analyze the effect of embedding compression, we compare the semantic reconstruction quality across different PCA embedding dimensions, as shown in Figure 1. The visualizations illustrate how varying the dimensionality from 3 to 64 affects the ability to preserve fine-grained semantic details while maintaining computational efficiency.

Lower-dimensional embeddings (3–9 dimensions) result in a significant loss of semantic information, leading to blurry segmentations and imprecise object boundaries. The compression at this level fails to retain fine details, causing objects to merge and reducing the distinctiveness of different semantic regions. As the dimensionality increases to 12–24, the quality of segmentation improves considerably. The reconstructed semantic maps become more structured, with clearer object outlines and more accurate spatial distributions. However, some artifacts remain, particularly in complex scene regions where fine-grained features are necessary for precise segmentation.

At higher embedding dimensions (32–64), the semantic reconstructions closely resemble the original feature

Methods	Metrics	Avg.	Room0	Room1	Room2	Office0	Office1	Office2	Office3	Office4
NICE-SLAM	PSNR↑	24.42	22.12	22.47	24.52	29.07	30.34	19.66	22.23	24.94
	SSIM↑	0.809	0.689	0.757	0.814	0.874	0.886	0.797	0.801	0.856
	LPIPS↓	0.233	0.330	0.271	0.208	0.229	0.181	0.235	0.209	0.198
Co-SLAM	PSNR↑	30.24	27.27	28.45	29.06	34.14	34.87	28.43	28.76	30.91
	SSIM↑	0.939	0.910	0.909	0.932	0.961	0.969	0.938	0.941	0.955
	LPIPS↓	0.252	0.324	0.294	0.266	0.209	0.196	0.258	0.229	0.236
ESLAM	PSNR↑	29.08	25.32	27.77	29.08	33.71	30.20	28.09	28.77	29.71
	SSIM↑	0.929	0.875	0.902	0.932	0.960	0.923	0.943	0.948	0.945
	LPIPS↓	0.336	0.313	0.298	0.248	0.184	0.228	0.241	0.196	0.204
SplaTAM	PSNR↑	33.98	32.48	33.72	34.96	38.34	39.04	31.90	29.70	31.68
	SSIM↑	**0.969**	0.975	0.970	**0.982**	0.982	**0.982**	0.965	0.950	0.946
	LPIPS↓	**0.099**	0.072	0.096	0.074	**0.083**	0.093	**0.100**	0.118	0.155
SGS-SLAM	PSNR↑	**34.66**	32.50	34.25	**35.10**	**38.54**	**39.20**	**32.90**	**32.05**	**32.75**
	SSIM↑	**0.973**	**0.976**	**0.978**	0.981	**0.984**	0.980	0.967	0.966	0.949
	LPIPS↓	0.096	**0.070**	0.094	**0.070**	0.086	**0.087**	0.101	**0.115**	**0.148**
Ours	PSNR↑	32.94	27.92	30.89	32.77	36.19	36.99	29.23	31.47	33.57
	SSIM↑	0.914	0.8246	0.9028	0.9312	0.9509	0.9409	0.9229	0.9208	0.9367
	LPIPS↓	0.148	0.2081	0.1409	0.1190	0.1310	0.1439	0.1685	0.1460	0.1269

Table 3. Comparison of reconstruction quality of SLAM methods on different Replica scenes.



(a) Dim = 3 (b) Dim = 6 (c) Dim = 9

(d) Dim = 12 (e) Dim = 16 (f) Dim = 24

(g) Dim = 32 (h) Dim = 48 (i) Dim = 64

Figure 1. Comparison of PCA embedding compression on the **backpack** query across different dimensions. Each row represents different compression levels (3–64), demonstrating how embedding dimensionality affects reconstruction quality and semantic consistency.

space. Objects maintain well-defined contours, and the color-coded segmentations demonstrate a high level of semantic accuracy. The 64-dimensional case provides the most faithful reconstruction, preserving both large-scale structure and fine object details. However, beyond this point, the benefits of increasing dimensionality further become negligible while computational overhead continues to grow.

These results confirm that PCA effectively retains essential semantic information while ensuring computational efficiency. The selection of 64 as the embedding dimension achieves a balance between compression quality, segmentation accuracy, and real-time performance, making it the optimal choice for LEG-SLAM.