

Hier-SLAM++: Neuro-Symbolic Semantic SLAM with a Hierarchically Categorical Gaussian Splatting

Boying Li^{1*}, Vuong Chi Hao², Peter J. Stuckey¹, Ian Reid³, and Hamid Rezatofighi¹

Abstract—We propose Hier-SLAM++, a comprehensive Neuro-Symbolic semantic 3D Gaussian Splattering SLAM method with both RGB-D and monocular input featuring an advanced hierarchical categorical representation, which enables accurate pose estimation as well as global 3D semantic mapping. The parameter usage in semantic SLAM systems increases significantly with the growing complexity of the environment, making scene understanding particularly challenging and costly. To address this problem, we introduce a novel hierarchical representation that encodes both semantic and geometric information in a compact form into 3D Gaussian Splatting, leveraging the capabilities of large language models (LLMs) as well as the 3D generative model. By utilizing the proposed hierarchical tree structure, semantic information is symbolically represented and learned in an end-to-end manner. We further introduce an advanced semantic loss designed to optimize hierarchical semantic information through both Intra-level and Inter-level optimizations. Additionally, we propose an improved SLAM system to support both RGB-D and monocular inputs using a feed-forward model. To the best of our knowledge, this is the first semantic monocular Gaussian Splattering SLAM system, significantly reducing sensor requirements for 3D semantic understanding and broadening the applicability of semantic Gaussian SLAM system. We conduct experiments on both synthetic and real-world datasets, demonstrating superior or on-par performance with state-of-the-art methods, while significantly reducing storage and training time requirements. Our project page is available at: <https://hierslampp.github.io/>.

Index Terms—Semantic SLAM, hierarchical category, gaussian splatting, RGB-D, monocular.

I. INTRODUCTION

VISUAL Simultaneous Localization and Mapping (SLAM) is a key technology for ego-motion estimation and scene perception, and is widely employed across various domains such as autonomous drones [1], self-driving vehicles [2], and interactive experiences in Augmented Reality (AR) and Virtual Reality (VR) [3]. Semantic information, which provides high-level knowledge about the environment, is indispensable for comprehensive scene understanding. It plays a crucial role in enabling intelligent robots to perform complex tasks. In recent decades, advances in image segmentation and enriched map representations have significantly accelerated progress in semantic visual SLAM [4]–[12].

Recently, 3D Gaussian Splatting has emerged as a powerful representation for 3D scenes [13]–[15], owing to its fast

¹ Faculty of Information Technology, Monash University, Australia.² VinUniversity, Vietnam. ³ Mohamed bin Zayed University of Artificial Intelligence, United Arab Emirates. * Corresponding author: Boying Li (boying.li@monash.edu). This work is supported by the DARPA Assured Neuro Symbolic Learning and Reasoning (ANSR) program under award number FA8750-23-2-1016. The work has received partial funding from The Australian Research Council Discovery Project ARC DP2020102427.

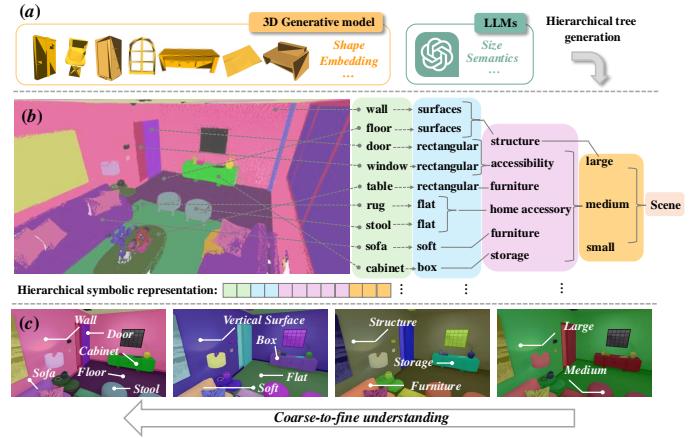


Fig. 1. (a). The hierarchical tree is generated by integrating geometric information and semantic messages, utilizing 3D generative models and Large Language Models (LLMs). (b). The global 3D Gaussian map generated by Hier-SLAM++ with learned semantic labels is shown on the left. The established hierarchical tree of the semantic information is organized on the right. Based on the established tree, the hierarchical symbolic representation for semantic information is shown at the bottom of the block, which compresses semantic data, reducing both memory usage and training time of the semantic SLAM. (c). The rendered semantic map at different levels shows a coarse-to-fine understanding, beneficial for real-world scenarios with shifting perspectives from distant to close.

rendering and optimization efficiency enabled by the highly parallelized rasterization of 3D primitives. Specifically, it models the *continuous* distribution of geometric attributes using Gaussian functions. This formulation not only enhances the 3D representation fidelity but also facilitates efficient optimization, making it particularly suitable for SLAM tasks. While promising, current SLAM systems based on 3D Gaussian Splatting [16]–[20] primarily focus on geometric reconstruction. However, the absence of semantic integration limits their applicability in complex downstream tasks that require high-level scene understanding.

To enable semantic understanding in SLAM systems, a straightforward approach, analogous to the geometric modeling of 3D Gaussians, is to augment each 3D primitive with a discrete semantic label and represent its distribution using a categorical distribution, typically implemented via a flat Softmax-based representation. However, 3D Gaussian Splatting is already a storage-intensive approach [21], [22], requiring numerous primitives with rich parameters to achieve high-quality rendering. Adding semantic distributions for each primitive further increases memory and computational costs, with complexity growing linearly with the number of semantic classes, making it impractical for complex semantic understanding. Moreover, given a limited number of observations

(e.g., video frames), introducing high-dimensional semantic parameters leads to the curse of dimensionality, hindering optimization performance. This added overhead is especially problematic in robotics, where computational efficiency and resource constraints are critical. Therefore, a key challenge is to design semantic encoding strategies that are both efficient and scalable, enabling robust semantic SLAM without compromising runtime and memory efficiency.

Semantic information often exhibits a natural *hierarchical structure*. As illustrated in Fig. 1, semantic classes can be abstracted into multi-level concepts, such as structures and surfaces, which can be further organized into a hierarchical symbolic tree. With careful design, this structure enables rich semantics to be encoded efficiently using a small number of symbolic nodes, yielding a compact and scalable representation. For instance, a binary tree of depth 10 can represent up to 2^{10} classes using only 20 nodes (i.e., 2×10 nodes through 2-dimensional Softmax at each level), reducing the complexity from $O(N)$ to $O(\log N)$.

Building upon this concept, we propose Hier-SLAM++, a comprehensive neural-symbolic semantic SLAM framework based on 3D Gaussian Splatting, featuring a novel hierarchical representation for semantic information. To design an effective hierarchical structure, we incorporate both semantic and geometric cues, including shape and size information. Specifically, we leverage a text-to-3D generative model [23] to extract shape-aware features for each class and organize them into hierarchical levels. Additionally, we employ Large Language Models (LLMs) to enrich the hierarchy with semantic relations and size priors, compensating for the limitations of generative models. With the constructed tree, semantic information in the environment is represented as symbolic nodes along a hierarchical path and learned end-to-end during SLAM optimization. Specifically, we propose two types of hierarchical semantic representations, one-hot and binary semantic embedding, which offer different degrees of compactness. To enhance semantic understanding, we introduce a novel hierarchical loss function that performs joint optimization at multiple levels, including Intra-level and Inter-level supervision. Furthermore, Hier-SLAM++ supports both RGB-D and monocular settings. We employ the geometric output of a 3D feed-forward model [24] as a depth prior, effectively removing the reliance on dedicated depth sensors. This enables monocular semantic SLAM and broadens the applicability of our system. We further refine the 3D Gaussian Splatting-based SLAM pipeline to improve both tracking performance and runtime efficiency. Experiments on both synthetic and real-world datasets demonstrate that Hier-SLAM++ achieves superior or on-par performance with state-of-the-art methods in localization, mapping, and semantic understanding under the RGB-D setting, while maintaining competitive performance in the monocular setting. The main contributions are as follows:

- 1) We propose a hierarchical tree structure that integrates semantic, shape, and size information, constructed using LLMs and 3D generative models. This representation efficiently encodes semantic messages while preserving its structural hierarchy, leading to reduced memory consumption and faster training.

- 2) Based on the generated hierarchical tree, we propose two types of semantic embeddings that offer different degrees of compactness. Furthermore, we introduce a novel optimization strategy for hierarchical semantic representations, incorporating both Intra-level and Inter-level losses to enable comprehensive refinement of both embeddings.

- 3) We develop an enhanced SLAM framework that supports both RGB-D and monocular settings by leveraging a 3D feed-forward model as a geometric prior. To the best of our knowledge, Hier-SLAM++ is the first Gaussian Splatting-based semantic SLAM system capable of operating with monocular input, significantly lowering sensor requirements and improving deployment flexibility.

This paper builds upon our previous work [25], with the following major extensions: 1) The tree construction process is enhanced by integrating 3D generative models and improving the use of LLMs, resulting in a more effective symbolic tree structure. 2) In addition to the one-hot semantic representation from our earlier work, we propose a binary semantic representation for further compactness. 3) We improve semantic optimization with a new hierarchical loss formulation and introduce a comprehensive hierarchical optimization strategy applicable to both one-hot and binary representations. 4) We propose a monocular pipeline to enable robust monocular SLAM by leveraging 3D feed-forward models, thereby extending the RGB-D semantic SLAM system into a unified framework that supports both RGB-D and monocular inputs. Additionally, we include new real-world datasets and experiments to further validate the proposed approach.

II. RELATED WORK

A. Visual SLAM system

Visual Simultaneous Localization and Mapping (SLAM) is a long-standing topic in computer vision and robotics. It jointly estimates camera poses and reconstructs global maps by optimizing appearance-based objectives [26]–[36]. One classical monocular-based approach aims to estimate accurate camera poses and a sparse global map, by minimizing the reprojection error terms [28]–[30], or by minimizing the photometric losses [31]–[33]. DTAM [34] takes a different direction by performing dense reconstruction and camera tracking with RGB inputs, though real-time capability relies on GPU support. Another category of methods adopts RGB-D input for dense 3D reconstruction, which is able to provide richer information. The works [35], [36] jointly estimate camera poses and dense surfaces from RGB-D input using a volumetric TSDF map. Alongside advances in purely geometric SLAM, semantic SLAM has gained increasing attention for its ability to provide high-level scene understanding [6], [8], [37]–[44]. Sparse semantic SLAM methods rely on pre-scanned models [37], [38] or use bounding boxes [39]–[41] and quadrics [42], though accuracy remains limited. In contrast, semantic understanding in dense SLAM becomes more tractable and effective when combined with dense observations from RGB-D inputs [45]–[47] or dense visual SLAM pipelines [44]. More recently, new paradigms have emerged to enhance dense SLAM with richer scene understanding across both monocular and RGB-D inputs, such as implicit neural fields and 3D Gaussian

Splatting. Furthermore, semantic integration allows current SLAM systems to move beyond geometric reconstruction to deliver high-level scene interpretation, such as class-level mapping, semantic navigation, and task-aware perception. This significantly broadens its applications in real-world fields.

Some works, such as Kimera [48], adopt dynamic scene graphs for representing *entities* within the observed environment [49]–[53]. In contrast, we propose a generalizable hierarchical symbolic representation for modeling *semantics* in the world, leveraging large language models and 3D generative priors, along with symbolic capabilities [54], [55] to abstract semantic concepts without relying on scene-specific definitions. Crucially, our symbolic hierarchy is carefully designed to produce compact semantic embeddings and is seamlessly integrated into the neural learning process within the SLAM system, offering both structural abstraction and interpretability for semantic encoding.

B. Neural implicit visual SLAM

Neural Radiance Fields (NeRF) [56] introduce a novel scene representation paradigm, inspiring the integration of NeRF with SLAM pipelines for simultaneous localization and high-quality mapping. Unlike traditional SLAM methods that rely on explicit geometry, NeRF-SLAM models represent scenes implicitly, enabling continuous and photorealistic reconstruction from sequential views. Early work such as iNeRF [57] explored solely pose tracking with a fixed pre-trained neural field. iMap [58] advanced this idea by proposing a keyframe-based RGB-D SLAM that jointly optimizes camera poses and a global implicit map using a single MLP. NICE-SLAM [59] enhances robustness and scalability by introducing a hierarchical neural implicit representation, which is later extended to the monocular setting in NICER-SLAM [60]. Recent works focus on improving efficiency: ESLAM [61] leverages a hybrid representation combining implicit TSDFs with multi-scale feature planes to achieve faster operation; Point-SLAM [62] adopts a point-based neural representation to enable efficient and detail-preserving dense SLAM. To further enhance scene understanding beyond geometry and appearance, researchers also incorporate semantics into NeRF-based SLAM systems [7], [11], [12], [63]. Some works leverage the semantic information extracted from 2D observations to integrate them into the visual SLAM system and generate the global semantic map [7], [63]. SNI-SLAM [11] proposes a semantic RGB-D SLAM which learn semantics by integrating geometry, appearance, and semantic features into a shared feature space. However, these methods remain constrained by the slow convergence and high overhead of neural implicit representations [64], [65], especially when jointly optimizing semantics. The inherent inefficiency of NeRF representations still conflicts with the real-time demands of SLAM systems—let alone the added cost from semantic integration. In contrast, Gaussian Splatting provides significant advantages, including fast rendering performance and high-density reconstruction quality, making it a promising new kind of map representation.

C. 3D Gaussian Splatting SLAM

3D Gaussian Splatting has recently emerged as a promising 3D representation due to its fast rendering and high-fidelity reconstruction. Compared to implicit representations, it models the environment as a collection of explicit Gaussian distributions with geometric and appearance attributes. Several SLAM systems have adopted this representation in their pipelines. SplaTAM [16] proposes a RGB-D SLAM framework with isotropic Gaussians representation using silhouette guidance for pose estimation, while MonoGS [17] presents an efficient solution supporting both monocular and RGB-D inputs. Photo-SLAM [19] uses 3D points from ORB-SLAM to initialize Gaussian positions; GS-SLAM [18] introduces a coarse-to-fine tracking strategy with sparse Gaussian selection; and Gaussian-SLAM [20] utilizes DROID-SLAM poses to manage multiple 3D Gaussian submaps. These approaches showcase the versatility of 3D Gaussian Splatting in various SLAM settings. However, integrating semantics into Gaussian Splatting-based SLAM remains challenging. Jointly optimizing geometry, appearance, and semantics involves heterogeneous value ranges and significantly increases memory and computational demands. GS³LAM [66] fuses 2D semantic labels into an RGB-D SLAM framework, with embeddings sized according to the number of semantic classes. SGS-SLAM [67] appends RGB 3-channels to support semantic visualization. SemGauss-SLAM [68] leverages flat semantic embeddings supervised by foundation models, though this introduces high computational cost and overlooks the hierarchical structure of real-world semantics. To address these limitations, we propose a compact and efficient solution that fully exploits the hierarchical nature of semantic categories. Our method encodes semantic information into a hierarchical symbolic tree, generated through the synergy of large language models and 3D generative priors, and learns this semantic representation end-to-end within the SLAM framework operation.

III. METHOD

The overall pipeline of our method is illustrated in Fig. 2. At the core of our approach, we model the entire semantic space as a compact hierarchical structure, where each semantic class is represented by a symbolic hierarchy. To construct a generative hierarchical tree that captures both semantic relations and associated geometric attributes, we leverage Large Language Models (LLMs) and 3D generative models. The hierarchical representation and tree generation process are detailed in Section III-A. Based on the proposed structure, semantic information is encoded into the embedding space in two formats: a *one-hot representation* and a *binary representation*, both of which are hierarchically organized and compact. To support end-to-end training, we introduce hierarchical optimization with two loss functions, Intra-level and Inter-level, which jointly optimize the semantic space based on incoming observations. The two proposed tree encoding schemes and hierarchical optimization are discussed in detail in Section III-B. In the monocular setting, we leverage geometric priors extracted from a feed-forward 3D model to guide learning during SLAM, incorporating online depth

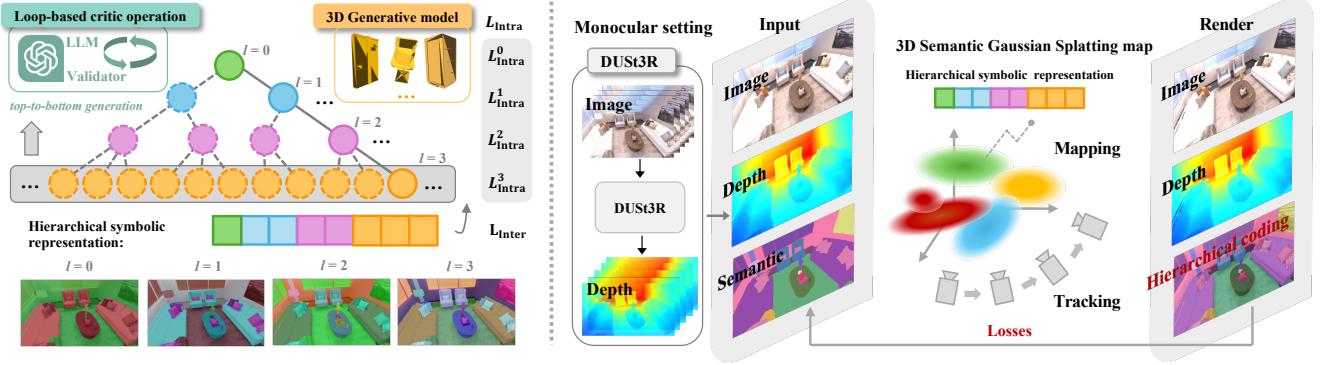


Fig. 2. Overview of the Hier-SLAM++ pipeline. **Top Left:** Hierarchical representation of semantic information. The Tree Generation process leverages the capabilities of both LLMs and 3D Generative Models (see Section III-A). This hierarchical tree is used to establish a symbolic coding for each Gaussian primitive. Additionally, we introduce a novel loss function that combines Intra-level Loss L_{Intra} and Inter-level Loss L_{Inter} to optimize the hierarchical semantic representation (see Section III-B). **Bottom Left:** An example of hierarchical semantic rendering. **Right:** The global 3D Gaussian map is initialized using the first frame of the video stream input. The system then alternates between the *Tracking* and *Mapping* steps as new frames are processed (see Section III-D). In the RGB-D setting, depth is directly obtained from sensor input, whereas in the monocular setting the 3D feed-forward method (DUS3R) is used to generate a geometric prior for depth estimation (see Section III-C).

adjustment and geometric supervisory signals, as described in Section III-C. Finally, the full semantic Gaussian Splatting SLAM system is presented in Section III-D.

A. Hierarchical Representation and Generation

Leveraging the inherent hierarchy of semantic classes, we propose a hierarchical symbolic representation to encode semantic information in a compact and efficient manner, enhancing the scalability of semantic SLAM system. This section presents a detailed description of the proposed hierarchical tree parametrization and its generation process using both LLMs and 3D generative models.

1) Hierarchical Tree Parametrization: The proposed hierarchical representation is abstracted as a multi-level tree, formulated as $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where \mathbf{V} and \mathbf{E} denote the sets of vertices and edges, respectively, as illustrated in Fig. 3. The vertex set is defined as $\mathbf{V} = \{v_l\}_{l=0}^L$, where v_l represents the set of semantic classes at level l of the tree hierarchy, spanning from the root level ($l = 0$) to the deepest level ($l = L$). The edge set $\mathbf{E} = \{e_m\}_{m=0}^{L-1}$ encodes the subordinate relationships between adjacent levels in the hierarchy, capturing semantic attribution, relative scale, and geometric structure. The original semantic classes are represented as leaf nodes located at the deepest level ($l = L$) of the hierarchical tree. For the i -th semantic class g^i , its hierarchical representation is defined as:

$$g^i = \{v_l^i, e_m^i \mid l = 0, 1, \dots, L; m = 0, 1, \dots, L-1\}, \quad (1)$$

which corresponds to a hierarchical path for the semantic meaning: $g^i = v_0^i \xrightarrow{e_0} v_1^i \xrightarrow{e_1} \dots \xrightarrow{e_{L-2}} v_{L-1}^i \xrightarrow{e_{L-1}} v_L^i$. For example, the 5-level hierarchical symbolic representation of the class Bed can be expressed as: $g^{\text{bed}} = \{v_0 : \text{LargeItems}\} \rightarrow \{v_1 : \text{Furnishings}\} \rightarrow \{v_2 : \text{BedroomFurniture}\} \rightarrow \{v_3 : \text{Rectangular}\} \rightarrow \{v_4 : \text{Bed}\}$. In this representation, the original semantic label g^i is abstracted as a hierarchical path from a coarse category to a fine-grained concept. At each level, symbolic nodes v_l^i represent the semantic attributes of the class, while inter-level relationships, such as inclusion

and possession, are described by the edge elements e_m^i . As will be demonstrated in Section III-B, with a carefully structured design, the proposed hierarchical representation yields a compact and computationally efficient representation. It allows each semantic class to be expressed in a progressive, interpretable, and symbolic manner, incorporating both semantic abstraction and geometric structure. Moreover, from a hierarchical perspective, the original flat representation can be regarded as a degenerate case of a single-level tree, enabling a unified formulation. For clarity and notational simplicity, we omit the superscripts (i.e., class indices) of variables in the remainder of this section.

2) Tree Generation via LLMs and 3D Generative Models: To represent semantic information in a reasonable, comprehensive, and compact manner, the tree structure should integrate both semantic and geometric cues, two fundamental sources of information in unknown environments that are essential for achieving a holistic understanding of the global 3D scene. To extract high-level semantic concepts, including functional relationships and abstract descriptors, we employ large language models (LLMs), specifically GPT-4 Turbo [69], which is well known for its strong commonsense reasoning and language understanding abilities. On the other hand, to capture general geometric information beyond dataset-specific constraints, we employ 3D generative models, particularly text-to-3D frameworks. These models are capable of producing diverse and geometrically accurate 3D objects from textual prompts after appropriate training, offering a unified and generalizable geometric representation. Specifically, we adopt MeshGPT [23], a text-to-3D model that learns symbolic 3D embeddings for the local mesh geometry and topology of each object. This property makes it particularly well-suited for our goal of constructing geometry-aware hierarchical semantic structures. However, the text-to-3D method generates 3D objects within a unified coordinate system but omits explicit size information. To address this limitation, we complement the generated geometric shapes with size attributes inferred from large

language models (LLMs), enabling a more comprehensive geometric representation for each semantic class. Leveraging both large language models (LLMs) and 3D generative models enables the construction of an elaborate and well-structured hierarchical tree, where each level captures a distinct aspect of a class's attributes—semantic, functional, or geometric, which further enriches the depth and expressiveness of global 3D scene understanding. The remainder of this section presents a detailed description of the hierarchical tree generation process.

The overall process of hierarchical tree generation is illustrated in Fig. 3. At the top level, the initial grouping divides all semantic classes into several subgroups based on their typical physical sizes in real-world scenes, which are obtained through prompt-based queries to the LLM. The generation of level-0 message through this size-based grouping is formulated as:

$$\{v, e\}_0 = \mathcal{L}^{\text{size}}\{g\} \quad (2)$$

where $\mathcal{L}^{\text{size}}$ denotes the LLM-driven grouping operator. The resulting clusters categorize the semantic classes into groups such as `SmallItems`, `MediumItems`, and `LargeItems`. Details of the specific prompt used to query the LLM are provided in Appendix-I.

Next, the semantic classes within each size-based group are used as input prompts to the LLM for further clustering based on semantic relationships, with an emphasis on functional properties. This step generates the following levels structure, formulated as:

$$\{v, e\}_1 = \mathcal{L}^{\text{func}}\{g\} \quad (3)$$

where $\mathcal{L}^{\text{func}}$ denotes the LLM-driven clustering operator based on functional similarity. It is worth noting that $\{v, e\}_1$ does not refer strictly to a single hierarchical level. Rather, it denotes a generic stage of functional grouping that follows the initial size-based partitioning. Depending on the balance and structural design of the overall tree, the functional clustering process may be applied multiple times to generate several intermediate levels. This hierarchical refinement enables the representation to achieve maximum semantic compactness while maintaining accuracy and interpretability. The specific prompt used for this functional clustering is provided in Appendix-I.

After generating the preceding semantic substructures using LLMs across multiple levels, 3D generative models are employed to further cluster semantic classes based on their geometric characteristics. This geometry-based symbolic representation complements the language-derived hierarchy, enabling a unified structure that incorporates both semantic and geometric information. Specifically, we adopt MeshGPT [23] to capture geometric information for each semantic class. This text-to-3D framework encodes object shapes and internal topological relationships into latent, quantized embeddings, resulting in a symbolic 3D representation within the geometric shape space. Given a text prompt (i.e., the class name) as input to the pre-trained MeshGPT model, we extract the latent quantized embeddings to represent the geometric information of the class:

$$\mathbf{z} = \mathcal{E}\{g\} \in \mathbb{N}^{F \times 2} \quad (4)$$

where \mathcal{E} denotes the pre-trained MeshGPT encoder, and \mathbf{z} is the quantized geometric embedding extracted for class g . Here, F represents the number of triangle faces used in the mesh representation [23]. Since the number of triangle faces F varies across semantic objects, we first compute an average shape embedding $\mathbf{z}_{\text{avg}} \in \mathbb{R}^{1 \times 2}$ for each class. These averaged embeddings are then concatenated and normalized to a standard distribution (i.e., zero mean and unit variance) to ensure consistency across classes:

$$\tilde{\mathbf{z}} = \mathcal{N}\{\mathcal{C}(\mathbf{z}_{\text{avg}})\} \quad (5)$$

where \mathcal{N} and \mathcal{C} denote the normalization and concatenation operations, respectively. The resulting normalized embeddings $\tilde{\mathbf{z}}$ are subsequently clustered using the K-means++ algorithm [70], denoted by the operator \mathcal{K} . To address the lack of descriptive labels for the resulting clusters, we further employ an LLM-based summarization module to interpret and refine the clustering outcomes:

$$\{v, e\}_2 = \mathcal{L}^{\text{sum}}\{\mathcal{K}(\tilde{\mathbf{z}})\} \quad (6)$$

where \mathcal{L}^{sum} denotes the use of an LLM to generate semantic summaries for each geometric cluster. The resulting descriptive labels, such as “*box*”, “*flat*”, or “*soft*”, capture the dominant geometric attributes within each group, thereby enhancing the interpretability of the hierarchical structure. Details of the LLM prompts and summarization process are provided in Appendix-I.

Whenever LLMs are employed, we ensure that the outputs are balanced, meaningful, and descriptively coherent through carefully designed prompt inputs, as detailed in Appendix-I. To further ensure the structural correctness within each level of the hierarchical tree, we introduce a *loop-based critic operation* performs multiple iterations. In each iteration, an LLM is followed by a validator module that checks the quality of the generated clusters. Specifically, within each LLM-based layer, the validator assesses the clustering results in terms of completeness and semantic consistency by comparing them against the input prompt. Correctly grouped nodes are retained, while erroneously included classes are removed. Any omitted classes are compiled into a new prompt and fed into the next iteration of LLM-based clustering. This process repeats until no omitted classes remain at the current level, ensuring a complete and semantically accurate hierarchical structure. Finally, a comprehensive validation process is applied to the entire hierarchical structure, including an automated check to detect and resolve duplicates or omissions across levels, followed by a manual inspection to ensure correctness and conceptual consistency. This ensures a fully balanced and semantically coherent tree.

B. Hierarchical Semantic Coding and Optimization

Based on the generated hierarchical tree structure, semantic information is encoded as compact, symbolic hierarchical codes and integrated into the 3D Gaussian primitives, which are learned in an end-to-end fashion from the input image stream. To ensure accurate and structured semantic understanding, we introduce a semantic hierarchical loss, composed

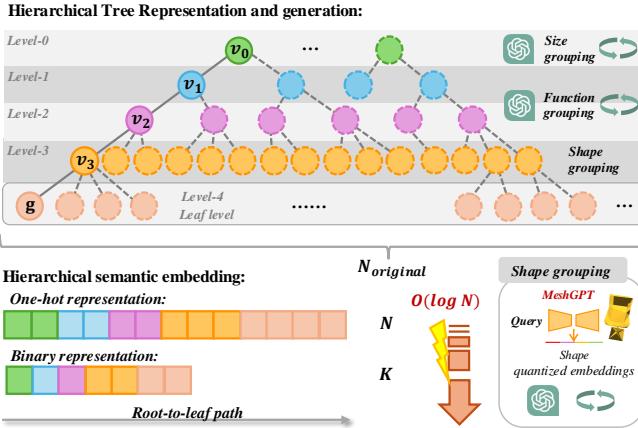


Fig. 3. Visualization of the hierarchical tree structure and semantic embedding. For **tree representation**, each semantic class is expressed hierarchically as $g = \{v_l, e_m\}$, where edges e_m are depicted as lines connecting tree nodes. The nodes within the same level are illustrated via the same color. For **tree generation**, we utilize both LLMs and 3D Generative Models to extract and group messages. For **hierarchical semantic embedding**, we represent semantic information using a hierarchical symbolic path. We propose two types of representations: the one-hot representation and the binary representation, which can reduce the original dimensionality by up to $O(\log N)$.

of both Intra-level and Inter-level components. These losses are incorporated into the overall optimization during the online SLAM process. The following sections provide a detailed description of the hierarchical semantic encoding and the proposed optimization framework.

1) *Hierarchical Semantic Coding*: As introduced in Section III-A, we model the semantic space as a hierarchical tree structure. Within this hierarchical space, each semantic class is represented in a structured and compact form that preserves essential semantic information. The proposed hierarchical semantic coding enhances the joint optimization of the SLAM system and offers valuable scalability for semantic understanding in the complex environments. Specifically, we introduce two types of hierarchical semantic coding strategies, both of which compress the original flat semantic labels while retaining crucial semantic messages.

One-hot representation: As illustrated in Fig. 3, the hierarchical semantic embedding \mathbf{h} is constructed by concatenating one-hot embeddings from all levels of the tree:

$$\mathbf{h} = \mathcal{C}(\mathbf{h}_l) \in \mathbb{B}^N, \quad \mathbf{h}_l \in \mathbb{B}^n \quad (7)$$

where subscript denotes the l -th tree level, and \mathcal{C} represents the concatenation operator. At each level, \mathbf{h}_l is an n -dimensional one-hot vector (as shown in colored boxes in Fig. 3) used to represent symbolic nodes (colored circles) within that level. The dimension n corresponds to the maximum number of nodes at level l , indexing node representations from 0 to $n - 1$. The total dimensionality of the hierarchical semantic embedding is given by $N = \sum n$, i.e., the sum over all levels. Compared to the original flat representation of dimension N_{original} , this hierarchical structure enables an information-preserving reduction of up to $O(\log N)$ in representational complexity. The resulting compact semantic code significantly

reduces memory usage and facilitates efficient joint optimization in the SLAM system.

Binary representation: To further improve compactness, we propose a binary version of the hierarchical semantic representation. The binary embedding \mathbf{b} is constructed by concatenating the binary vectors \mathbf{b}_l across all levels of the hierarchical tree:

$$\mathbf{b} = \mathcal{C}(\mathbf{b}_l) \in \mathbb{B}^K, \quad \mathbf{b}_l \in \mathbb{B}^k, \quad \text{where } k = \lceil \log_2 n \rceil \quad (8)$$

where \mathbf{b}_l is the binary code converted from the one-hot embedding \mathbf{h}_l at level l , i.e., $\mathbf{b}_l = \text{Binary}(\mathbf{h}_l)$, as illustrated in the final row of Fig. 3. For example, if the one-hot vector has dimension $n = 8$ (representing 8 distinct semantic nodes indexed from 0 to 7), its binary representation requires only $k = 3$ bits for $2^3 = 8$. More generally, the binary code length at each level is reduced to $k = \lceil \log_2 n \rceil$, compared to the n -dimensional one-hot encoding.

2) *Loss Calculation*: To effectively and efficiently optimize the proposed hierarchical semantic codings, we introduce a hierarchical loss formulation that supervises semantic embedding both *within each level (intra-level)* and *across levels (inter-level)*. Specifically, the total semantic loss is defined as:

$$L_{\text{Semantic}} = \omega_1 L_{\text{Intra}} + \omega_2 L_{\text{Inter}} \quad (9)$$

where ω_1 and ω_2 are balancing coefficients for each loss term. Specifically, the intra-level loss L_{Intra} is employed within each level's embedding \mathbf{h}^l to ensure the optimization of semantic information within each level:

$$L_{\text{Intra}} = \sum_{l=0}^L L_{\text{ce}}(\mathcal{S}(\mathbf{h}^l), \mathcal{P}^l) \quad (10)$$

where L_{ce} indicates the cross-entropy semantic loss, and \mathcal{P}^l denotes the semantic ground truth at level l . \mathcal{S} stands for the Softmax operation, converting embeddings into probabilities. For our proposed compact version with binary representation \mathbf{b}^l , we employ the binary cross-entropy loss L_{bce} , instead of the standard cross-entropy, for computing the binary intra-level loss:

$$L_{\text{Intra}}^{\text{bin}} = \sum_{l=0}^L L_{\text{bce}}(\mathcal{S}(\mathbf{b}^l), \mathcal{P}^l) \quad (11)$$

To ensure accurate semantic understanding across all levels of the hierarchical structure, we introduce the inter-level loss L_{Inter} , which enforces consistency between the hierarchical representation and the original flat semantic distribution. To this end, we employ a lightweight decoder to transform the holistic hierarchical embedding \mathbf{h} into flat semantic predictions. Specifically, the decoder consists of two stacked 2D convolutional layers, with a ReLU activation applied between them. The output is then passed through a softmax function to produce a probability distribution over the flat semantic classes:

$$\text{decoder}(\mathbf{h}) = \text{Conv}(\text{ReLU}(\text{Conv}(\mathbf{h}))) \quad (12)$$

The inter-level loss is defined as:

$$L_{\text{Inter}} = L_{\text{ce}}(\mathcal{S}(\text{decoder}(\mathbf{h})), \mathcal{P}), \quad (13)$$

where $\mathcal{S}(\cdot)$ denotes the softmax function, \mathcal{P} is the ground truth flat semantic label, and L_{CE} is the standard cross-entropy loss.

C. Monocular setting with geometric priority

Accurate depth information provides valuable geometric cues for global map reconstruction and pose estimation in SLAM systems. While RGB-D methods directly acquire depth from sensors, monocular SLAM approaches [28]–[31] must rely on multi-view triangulation to infer depth and initialize mapping, which increases the difficulty of simultaneously performing localization and mapping. To address this limitation, a number of methods [71]–[76] introduce depth priors from learned depth networks into SLAM systems, achieving promising results. More recently, a new trend has emerged: *3D feed-forward models*, which can generate accurate global 3D representations of previously unseen environments after training. In our work, we adopt a precise and highly generalizable 3D feed-forward method to obtain the geometric prior used in monocular SLAM. Specifically, we employ DUS3R [24], which produces both 3D point clouds and pose estimations from sparse-view image inputs, serving as a strong geometric prior in our system. Details of the monocular SLAM setting in Hier-SLAM++ are provided in the following section.

1) *Geometric prior acquisition*: DUS3R [24] supports sparse view inputs with a minimum of two views. Larger view inputs enhance reconstruction accuracy by providing more information, yet at the cost of increased computational demands. To balance reconstruction accuracy and computational efficiency, we sample every S frames with a skip of δ frames from the input video to form an image set $\{I_i\}$.

$$\{I_i \mid i = i_0 + \delta \cdot (s - 1), s = 1, \dots, S\} \quad (14)$$

Each image set is then fed into DUS3R to generate geometric information, specifically the depth maps. While DUS3R is also capable of estimating camera poses for each input frame, our experiments reveal that its pose predictions exhibit relatively low accuracy, often performing worse than a simple constant-velocity pose model when integrated into a SLAM pipeline. Therefore, we retain only the geometric (depth) information predicted by DUS3R, as it provides more reliable guidance for our SLAM system.

2) *Geometric prior correction*: The depth predicted by DUS3R is scale-ambiguous and lacks metric consistency across image sets. This leads to differences in scale between the predicted geometry and the SLAM system's estimated global map, which evolves over time. We therefore incorporate a geometric prior correction step to resolve this scale inconsistency.

To align the scale of the depth prior with the current global map, we first estimate the scale and shift parameters by comparing the rendered depth from the SLAM map with the predicted depth from DUS3R. Specifically, for each input frame I_c , we render a depth map D_c and silhouette map S_c from the global 3D Gaussian map G using the current camera pose \mathbf{T}_c :

$$\{D_c, S_c\} = \mathcal{R}(G, \mathbf{T}_c) \quad (15)$$

where \mathcal{R} represents the rasterization procedure for the 3D Gaussian map. We then compute the global scale and shift parameters λ_c and τ_c by aligning the DUS3R-predicted depth \hat{D}_c to the rendered depth D_c using least squares estimation

over the visible regions M_c , as defined by the silhouette map S_c :

$$\{\lambda_c, \tau_c\} = \arg \min \left(\hat{D}_c^{\text{align}} - D_c \right)^2 M_c \quad (16)$$

where $\hat{D}_c^{\text{align}} = \lambda_c \cdot \hat{D}_c + \tau_c$ denotes the aligned depth. This simple affine transformation is sufficient to resolve the scale ambiguity inherent in the geometric priors obtained from DUS3R, while preserving their relative 3D structure for integration into the monocular SLAM pipeline.

Secondly, the calculated scale and shift parameters are used as initial values and are jointly optimized with the 3D Gaussians during the map optimization process, guided by a depth supervision loss:

$$\{\lambda_c, \tau_c, G\} = \arg \min \left(|\hat{D}_c^{\text{align}} - D_c| M_c + L_{\text{reg}} \right) \quad (17)$$

where L_{reg} is a regularization term defined as the absolute deviation between the optimized parameters and their initialization, preventing overfitting or convergence to poor local minima. By incorporating this optimization into the SLAM operation, the depth prior can be continuously refined online as new observations arrive, effectively providing geometric supervision throughout the SLAM process. This not only improves the quality of the global map reconstruction but also enhances the accuracy of camera pose estimation.

D. Semantic Gaussian Splatting SLAM

The global semantic 3D Gaussian Splatting SLAM system is illustrated in Fig. 2. Built upon the semantic 3D Gaussian representation, global reconstruction (Mapping) and camera pose estimation (Tracking) are alternately performed over the input video stream. In the RGB-D setting, depth information for each frame is directly acquired from the depth sensor. In the monocular setting, a feed-forward model is employed to generate geometric priors, as described in the previous section. The construction of the semantic 3D Gaussians and the overall semantic SLAM system are detailed in the following sections.

1) *Semantic 3D Gaussian representation*: Our semantic 3D Gaussian representation encodes both geometric and semantic information using Gaussian primitives augmented with hierarchical semantic embeddings. Motivated by [16], we adopt isotropic, view-independent Gaussians and integrate them with semantic feature vectors. Each semantic Gaussian is parameterized by its color \mathbf{c} , center position $\boldsymbol{\mu}$, radius r , opacity o , and semantic embedding \mathbf{h} . The influence of each Gaussian $G(\mathbf{X})$ [13] at 3D point \mathbf{X} follows the standard Gaussian formulation:

$$G(\mathbf{X}) = o \exp \left(-\frac{\|\mathbf{X} - \boldsymbol{\mu}\|^2}{2r^2} \right) \quad (18)$$

where \mathbf{X} stands for the 3D point.

Following [13], every semantic 3D Gaussian primitive within the current global map is projected onto the image plane using the tile-based differentiable α -compositing rendering. The semantic map H is rendered as follows:

$$H = \sum_{i=1}^n \mathbf{h}_i \alpha_i(\mathbf{X}) T_i \quad \text{with} \quad T_i = \prod_{j=1}^{i-1} (1 - \alpha_j(\mathbf{X})) \quad (19)$$

The rasterized color image C , depth image D , and the silhouette image S are defined as follows:

$$C = \sum_{i=1}^n \mathbf{c}_i \alpha_i(\mathbf{X}) T_i, D = \sum_{i=1}^n \mathbf{d}_i \alpha_i(\mathbf{X}) T_i, S = \sum_{i=1}^n \alpha_i(\mathbf{X}) T_i \quad (20)$$

Different from previous work [16], which uses separate forward and backward Gaussian modules for different parameters, our Hier-SLAM++ system adopts a unified forward and backward modules that processes all parameters, including semantics, color, depth, and silhouette images, which significantly improving overall runtime efficiency.

2) *Tracking*: The tracking step focuses on pose estimation for each frame. For each incoming frame, the constant velocity model is first applied to initialize its pose based on the previous frame's pose estimation. Following that, a pose optimization is performed with the global Gaussian map G fixed:

$$\mathbf{T}_c = \arg \min_{\mathbf{T}_c} L_{\text{Track}}(G, \mathbf{T}_c) \quad (21)$$

where the tracking loss includes the rendering color and depth losses:

$$L_{\text{Track}} = M(w_1 L_{\text{Depth}}(G, \mathbf{T}_c) + w_2 L_{\text{Color}}(G, \mathbf{T}_c)) \quad (22)$$

where L_{Depth} and L_{Color} denote the L1-loss for the rendered depth and color information. The rendering follows the tile-based differentiable α -compositing procedure, utilizing the current pose estimation along with the current global map estimation. Here, weights w_1 and w_2 are introduced to balance the two loss terms, and the optimization is only performed on the silhouette-visible image regions, defined as $M = (S > \delta)$. For the RGB-D setting, depth supervisory signals are directly obtained from depth sensors, whereas for the monocular setting, we leverage the geometric prior derived from the feed-forward method.

3) *Mapping*: The global map information, including semantic information, is optimized in the mapping procedure while keeping camera poses \mathbf{T}_i fixed:

$$G = \arg \min_G L_{\text{Map}}(G, \mathbf{T}_i) \quad (23)$$

The mapping optimization losses include depth, color, and semantic losses:

$$L_{\text{Map}} = w_3 M L_{\text{Depth}}(G, \mathbf{T}_i) + w_4 L'_{\text{Color}}(G, \mathbf{T}_i) + w_5 L_{\text{Semantic}}(G, \mathbf{T}_i) \quad (24)$$

where L_{Semantic} is the proposed semantic loss introduced in Section III-B, and L'_{Color} is the weighted sum of SSIM color loss and L1-Loss. The weights w_3 , w_4 , and w_5 are used to balance the different loss terms. Similar to the tracking process, depth supervisory signals come from sensors in the RGB-D setting. For the monocular setting, geometric priors derived from the feed-forward method are applied, and the two parameters (scale and shift) are further jointly optimized within the mapping process.

IV. EXPERIMENTS

A. Experiment settings

The experiments are conducted on both synthetic and real-world datasets, including ScanNet [77], Replica [78], and

TABLE I
RGB-D TRACKING PERFORMANCE ATE RMSE (CM) ON THE REPLICA.
BEST RESULTS ARE HIGHLIGHTED AS FIRST, SECOND, THIRD.

| Methods | Avg. | R0 | R1 | R2 | Of0 | Of1 | Of2 | Of3 | Of4 |
|------------------------------|------|------|------|------|------|------|------|------|------|
| iMap [58] | 4.15 | 6.33 | 3.46 | 2.65 | 3.31 | 1.42 | 7.17 | 6.32 | 2.55 |
| NICE-SLAM [59] | 1.07 | 0.97 | 1.31 | 1.07 | 0.88 | 1.00 | 1.06 | 1.10 | 1.13 |
| Vox-Fusion [80] | 3.09 | 1.37 | 4.70 | 1.47 | 8.48 | 2.04 | 2.58 | 1.11 | 2.94 |
| co-SLAM [81] | 1.06 | 0.72 | 0.85 | 1.02 | 0.69 | 0.56 | 2.12 | 1.62 | 0.87 |
| ESLAM [61] | 0.63 | 0.71 | 0.70 | 0.52 | 0.57 | 0.55 | 0.58 | 0.72 | 0.63 |
| Point-SLAM [62] | 0.52 | 0.61 | 0.41 | 0.37 | 0.38 | 0.48 | 0.54 | 0.69 | 0.72 |
| MonoGS-RGBD [17] | 0.79 | 0.47 | 0.43 | 0.31 | 0.70 | 0.57 | 0.31 | 0.31 | 3.2 |
| SplaTAM [16] | 0.36 | 0.31 | 0.40 | 0.29 | 0.47 | 0.27 | 0.29 | 0.32 | 0.55 |
| SNI-SLAM [11] | 0.46 | 0.50 | 0.55 | 0.45 | 0.35 | 0.41 | 0.33 | 0.62 | 0.50 |
| DNS SLAM [7] | 0.45 | 0.49 | 0.46 | 0.38 | 0.34 | 0.35 | 0.39 | 0.62 | 0.60 |
| SemGauss-SLAM [68] | 0.33 | 0.26 | 0.42 | 0.27 | 0.34 | 0.17 | 0.32 | 0.36 | 0.49 |
| SGS-SLAM [67] | 0.41 | 0.46 | 0.45 | 0.29 | 0.46 | 0.23 | 0.45 | 0.42 | 0.55 |
| Hier-SLAM [25] | 0.33 | 0.21 | 0.49 | 0.24 | 0.29 | 0.16 | 0.31 | 0.37 | 0.53 |
| Hier-SLAM++ (one-hot) | 0.31 | 0.24 | 0.36 | 0.23 | 0.30 | 0.15 | 0.28 | 0.39 | 0.51 |
| Hier-SLAM++ (binary) | 0.31 | 0.23 | 0.46 | 0.23 | 0.29 | 0.15 | 0.27 | 0.34 | 0.54 |

TUM-RGBD [79]. Following the evaluation metrics which used in previous SLAM methods [16], [59], we leverage ATE RMSE (cm) to quantify SLAM tracking accuracy. For mapping evaluation, Depth L1 (cm) is used to measure accuracy. To assess image rendering quality, we adopt PSNR (dB), SSIM, and LPIPS as evaluation metrics. In line with previous works [7], [11], [68], we assess semantic rendering performance using mIoU (mean Intersection over Union across all classes) to represent global semantic information. To evaluate computational efficiency, we report the runtime of the proposed SLAM method. For comparison, we benchmark our approach against state-of-the-art dense visual SLAM methods, including both NeRF-based and 3D Gaussian SLAM approaches, to demonstrate its effectiveness. Additionally, we compare with state-of-the-art semantic SLAM methods, covering both NeRF-based and Gaussian-based techniques, to highlight its capability in hierarchical semantic representation and scalability. Experiments are conducted on an NVIDIA L40S GPU, while runtime tests are evaluated on both the L40S and RTX 4090 to enable broader comparisons.

For experimental settings, the semantic embedding of each Gaussian primitive is initialized randomly. The semantic optimization loss weights ω_1 and ω_2 are set to 1.0 and 0.0, respectively, for the first η iterations, where η is set as 15. Afterwards, ω_1 remains 1.0, while ω_2 is increased to 5.0. This strategy means that we first apply the Inter-level loss to initialize the hierarchical semantic coding, followed by incorporating the Cross-level loss to refine the whole embedding. For tracking loss, we use $\delta = 0.99$, with weights $w_1 = 1.0$ and $w_2 = 0.5$. For mapping, the loss weights are set as follows: $w_3 = 1.0$, $w_4 = 0.5$, and $w_5 = 0.2$. The two types of semantic embeddings proposed in our method, one-hot and binary, are evaluated and reported across multiple experiments, including tracking, mapping, rendering, runtime, and semantic understanding, providing a comprehensive ablation study on the effectiveness and compactness of the hierarchical semantic representation.

TABLE II

RGB-D TRACKING PERFORMANCE ATE RMSE (CM) ON THE SCANNET. BEST RESULTS ARE HIGHLIGHTED AS FIRST, SECOND, THIRD, FOURTH.

| Methods | Avg. | 0000 | 0059 | 0106 | 0169 | 0181 | 0207 |
|------------------------------|--------------|--------------|-------------|--------------|--------------|-------------|-------------|
| NICE-SLAM [59] | 10.70 | 12.00 | 14.00 | 7.90 | 10.90 | 13.40 | 6.20 |
| Vox-Fusion [80] | 26.90 | 68.84 | 24.18 | 8.41 | 27.28 | 23.30 | 9.41 |
| Point-SLAM [62] | 12.19 | 10.24 | 7.81 | 8.65 | 22.16 | 14.77 | 9.54 |
| SplaTAM [16] | 11.88 | 12.83 | 10.10 | 17.72 | 12.08 | 11.10 | 7.46 |
| SGS-SLAM [67] | 9.87 | 11.15 | 9.54 | 10.43 | 10.70 | 11.28 | 6.11 |
| SemGauss-SLAM [68] | — | 11.87 | 7.97 | — | 8.70 | 9.78 | 8.97 |
| Hier-SLAM [25] | 11.36 | 11.45 | 9.61 | 17.80 | 11.93 | 10.04 | 7.32 |
| Hier-SLAM++ (one-hot) | 11.72 | 12.93 | 9.59 | 17.70 | 11.63 | 10.98 | 7.51 |
| Hier-SLAM++ (binary) | 11.59 | 13.24 | 9.60 | 17.83 | 11.75 | 9.83 | 7.29 |

B. Camera Tracking Accuracy

Tab. I presents our tracking performance on the Replica dataset [78] in the RGB-D setting, compared with state-of-the-art dense SLAM methods, both with and without semantic information. By accurately learning semantic information, our Hier-SLAM++ outperforms all existing methods in 6 out of 8 sequences. For the remaining two sequences, our method achieves nearly the same performance with sub-millimeter level localization error. The lowest average tracking error clearly demonstrate that our approach surpasses other state-of-the-art methods. We present our RGB-D tracking performance on the real-world ScanNet dataset [77] in Tab. II. Compared to the results on the Replica dataset, all methods exhibit degraded performance due to the noisy and sparse depth sensor input, as well as lower color image quality caused by motion blur. Specifically for semantic understanding, the semantic annotations provided by ScanNet are significantly noisier than those in Replica, containing noticeable noise and imprecise boundaries. All of these issues negatively impact the overall performance of semantic SLAM systems. We evaluate all six sequences and show that our method achieves performance comparable to state-of-the-art approaches [16], [25], demonstrating the robustness of our proposed method under these challenging conditions. The tracking performance on the TUM-RGBD dataset is shown in Tab. III. As this dataset does not provide semantic ground truth, we evaluate our method solely based on tracking accuracy. Without access to semantic information, our system operates as a standard visual SLAM pipeline. As a result, the tracking accuracy becomes comparable to the state-of-the-art method [16] in the TUM-RGBD dataset.

C. Global Mapping Performance

Following prior work [25], [67], [68], we evaluate depth accuracy on the Replica dataset [78], which provides high-quality ground truth depth maps for reliable evaluation of overall mapping performance. The quantitative results are presented in Tab. IV. Our method achieves the lowest depth error in 7 out of 8 sequences and maintains a consistently low average error across all sequences, with only a 0.1 cm difference compared to the best-performing baseline. By leveraging semantic understanding, our approach enables the

TABLE III

RGB-D TRACKING PERFORMANCE ATE RMSE (CM) ON THE TUM-RGBD. BEST RESULTS ARE HIGHLIGHTED AS FIRST, SECOND, THIRD, FOURTH.

| Methods | Avg. | fr1/desk | fr1/desk2 | fr1/room | fr2/xyz | fr3/off |
|--------------------|-------------|-------------|-------------|-------------|---------|-------------|
| Kintinuous [82] | 4.84 | 3.70 | 7.10 | 7.50 | 2.90 | 3.00 |
| ElasticFusion [83] | 6.91 | 2.53 | 6.83 | 21.49 | 1.17 | 2.52 |
| ORB-SLAM3 [30] | 2.21 | 1.60 | 2.37 | 5.83 | 0.32 | 0.94 |
| NICE-SLAM [59] | 15.87 | 4.26 | 4.99 | 34.49 | 31.73 | 3.87 |
| Vox-Fusion [80] | 11.31 | 3.52 | 6.00 | 19.53 | 1.49 | 26.01 |
| Point-SLAM [62] | 8.92 | 4.34 | 4.54 | 30.92 | 1.31 | 3.48 |
| SplaTAM [16] | 5.48 | 3.35 | 6.54 | 11.13 | 1.24 | 5.16 |
| Hier-SLAM++ | 5.62 | 3.31 | 6.59 | 11.73 | 1.34 | 5.14 |

TABLE IV

RECONSTRUCTION PERFORMANCE DEPTH L1 (CM) ON REPLICA. BEST RESULTS ARE HIGHLIGHTED AS FIRST, SECOND, THIRD.

| Methods | Avg. | R0 | R1 | R2 | Of0 | Of1 | Of2 | Of3 | Of4 |
|------------------------------|-------------|-------------|-------------|-------------|-------------|------|-------------|-------------|-------------|
| NICE-SLAM [59] | 2.97 | 1.81 | 1.44 | 2.04 | 1.39 | 1.76 | 8.33 | 4.99 | 2.01 |
| Vox-Fusion [80] | 2.46 | 1.09 | 1.90 | 2.21 | 2.32 | 3.40 | 4.19 | 2.96 | 1.61 |
| Co-SLAM [81] | 1.51 | 1.05 | 0.85 | 2.37 | 1.24 | 1.48 | 1.86 | 1.66 | 1.54 |
| ESLAM [61] | 0.95 | 0.73 | 0.74 | 1.26 | 0.71 | 1.02 | 0.93 | 1.03 | 1.18 |
| SNI-SLAM [11] | 0.77 | 0.55 | 0.58 | 0.87 | 0.55 | 0.97 | 0.89 | 0.75 | 0.97 |
| SGS-SLAM [67] | 0.36 | — | — | — | — | — | — | — | — |
| SemGauss-SLAM [68] | 0.50 | 0.54 | 0.46 | 0.43 | 0.29 | 0.22 | 0.51 | 0.98 | 0.56 |
| Hier-SLAM [25] | 0.49 | 0.58 | 0.40 | 0.40 | 0.29 | 0.19 | 0.51 | 0.95 | 0.57 |
| Hier-SLAM++ (one-hot) | 0.48 | 0.56 | 0.41 | 0.39 | 0.29 | 0.20 | 0.53 | 0.92 | 0.55 |
| Hier-SLAM++ (binary) | 0.49 | 0.59 | 0.40 | 0.40 | 0.30 | 0.17 | 0.52 | 0.95 | 0.57 |

learning of precise geometry in unseen environments, which in turn benefits both tracking and mapping within the whole semantic SLAM system. These results confirm the strong semantic mapping capability of our method and highlight its superior performance in dense 3D reconstruction.

D. Rendering Quality

Rendering performance is becoming an increasingly important evaluation metric, particularly for SLAM systems built upon new scene representations, such as neural implicit representations and gaussian splatting. Following prior works [59], [62], the evaluation is conducted on the input views of the Replica dataset. We present a quantitative analysis of the rendering performance of our proposed method in Tab. V. Our method outperforms state-of-the-art approaches in rendering quality across all 8 sequences, as well as in the overall average, surpassing both semantic and non-semantic SLAM baselines. The accurate semantic understanding learned by our system enhances both geometric and appearance representations through joint optimization, further improving the overall rendering fidelity of the SLAM system.

The visual comparison of rendered images is shown in Fig. 4. Benefiting from accurate semantic understanding and an improved SLAM framework, our method achieves significantly better detail preservation. Notable improvements include sharper and more accurate lamp boundaries in the first and fifth columns, as well as fine structural details, such as the shutters in the second column. Furthermore, our method produces high-quality renderings with fewer noise artifacts,

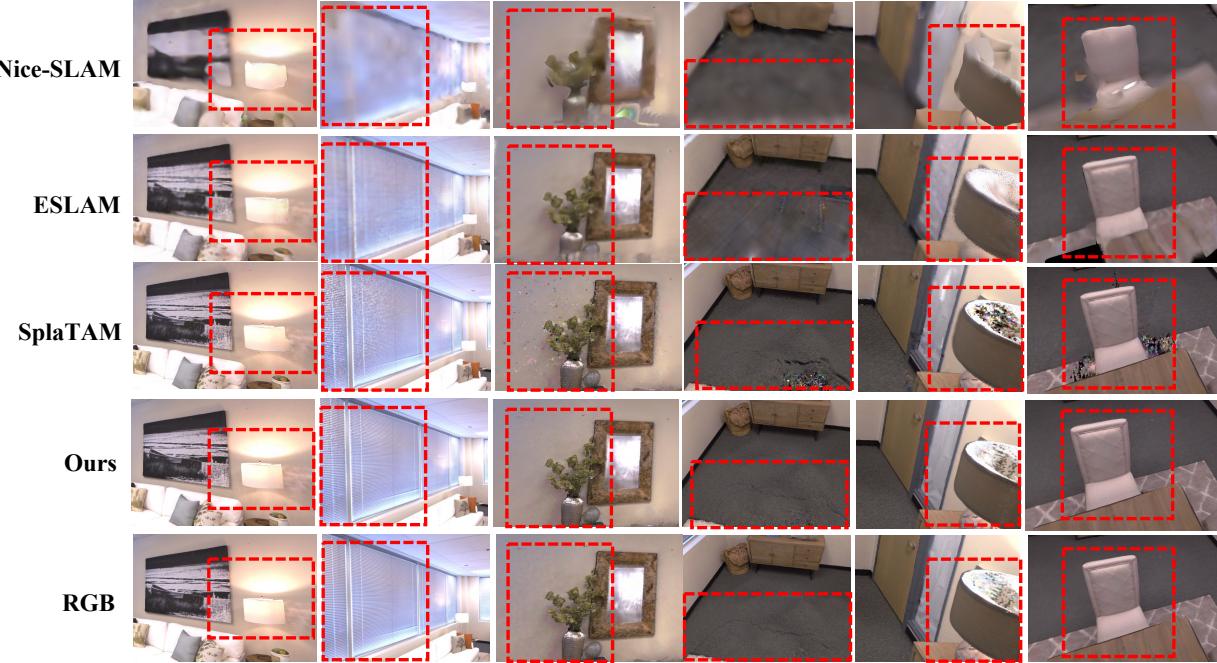


Fig. 4. Visualization of RGB image rendering performances on the Replica dataset [78]. **The first to third rows** show the rendering outputs of the comparison methods. **The fourth row** presents the image rendering results of our Hier-SLAM++. **The last row** displays the ground truth images for reference. We use red boxes to highlight key differences regions within the image. The visual comparison clearly demonstrates that our method achieves superior rendering quality, closely resembling the ground truth images compared to state-of-the-art approaches.

evident in the clearer floor in the fourth column and the cleaner reconstruction of the chair in the sixth column. These results demonstrate that improved mapping quality directly contributes to enhanced rendering fidelity. The joint optimization of semantic and photometric losses proves mutually beneficial: semantic learning facilitates more accurate global 3D reconstruction, which in turn leads to better rendering performance in our Hier-SLAM++ system.

E. Running time

The runtime results for the compared methods are provided in Tab. VI, covering both NeRF-based and Gaussian Splatting-based SLAM methods. For the reason that one compared baseline using a flat representation (Hier-SLAM++-flat) cannot run successfully on an NVIDIA RTX 4090 due to GPU memory limitations, we report the runtime separately for RTX 4090 and L40S. In the first block of Tab. VI, we observe that, compared to NeRF-based methods [59], Gaussian-based state-of-the-art methods [16] achieve significantly higher operating speeds, benefiting from the fast rendering and efficient optimization capabilities inherent to Gaussian Splatting representation. Furthermore, with the carefully refined forward and backward gaussian module introduced in our system, which serves as the core component of gaussian splatting-based SLAM systems, our method (Hier-SLAM++ w/o sem) achieves up to 2.4× faster tracking and 2.2× faster mapping compared to the best-performing method [16]. In the second block of Tab. VI, we evaluate the impact of different semantic representations on runtime performance. Our method maintains high efficiency

by employing hierarchical semantic coding (Hier-SLAM++-one-hot), achieving nearly 3× faster tracking compared to the flat semantic representation (Hier-SLAM++-flat). Furthermore, our proposed binary semantic representation (Hier-SLAM++-binary) demonstrates even greater efficiency, enabling more lightweight performance. The flat version is slightly faster in mapping, with an improvement of 0.38 seconds per-frame, this is due to its cross-entropy loss calculation directly without hierarchical optimization for the proposed methods. Notably, our Hier-SLAM++ achieves a rendering speed of 2000 FPS, which increases to 3000 FPS when semantic information is not included.

F. Hierarchical semantic understanding

We perform semantic understanding experiments on the synthetic dataset Replica [78] to demonstrate the comprehensive performance of our proposed method. Replica [78] is a synthetic indoor dataset that includes 102 semantic classes with high-quality semantic annotations. Similar to methods [7], [11], [68], we present our quantitative results, evaluated in mIoU (%) across all original semantic classes (102 classes) in Tab. VII, where the rendered semantic map is compared against the semantic ground truth. Additionally, consistent with the previous approach [67], we also report mIoU scores on a subset of semantic classes in the second block of the table. To further analyze the storage requirements of semantic SLAM methods, we report the parameter storage usage in the second block of Tab. VII. The generated hierarchical tree used in our method is shown in Appendix-II, where semantic concepts are

TABLE V
RENDERING PERFORMANCE PSNR, SSIM, LPIPS ON REPLICA. BEST RESULTS ARE HIGHLIGHTED AS **FIRST**, **SECOND**, **THIRD**.

| Methods | Metrics | Avg. | room0 | room1 | room2 | office0 | office1 | office2 | office3 | office4 |
|-----------------------|--------------------|-------|-------|-------|-------|---------|---------|---------|---------|---------|
| Visual SLAM | | | | | | | | | | |
| NICE-SLAM [59] | PSNR \uparrow | 24.42 | 22.12 | 22.47 | 24.52 | 29.07 | 30.34 | 19.66 | 22.23 | 24.94 |
| | SSIM \uparrow | 0.809 | 0.689 | 0.757 | 0.814 | 0.874 | 0.886 | 0.797 | 0.801 | 0.856 |
| | LPIPS \downarrow | 0.233 | 0.330 | 0.271 | 0.208 | 0.229 | 0.181 | 0.235 | 0.209 | 0.198 |
| Vox-Fusion [80] | PSNR \uparrow | 24.41 | 22.39 | 22.36 | 23.92 | 27.79 | 29.83 | 20.33 | 23.47 | 25.21 |
| | SSIM \uparrow | 0.801 | 0.683 | 0.751 | 0.798 | 0.857 | 0.876 | 0.794 | 0.803 | 0.847 |
| | LPIPS \downarrow | 0.236 | 0.303 | 0.269 | 0.234 | 0.241 | 0.184 | 0.243 | 0.213 | 0.199 |
| Co-SLAM [81] | PSNR \uparrow | 30.24 | 27.27 | 28.45 | 29.06 | 34.14 | 34.87 | 28.43 | 28.76 | 30.91 |
| | SSIM \uparrow | 0.939 | 0.910 | 0.909 | 0.932 | 0.961 | 0.969 | 0.938 | 0.941 | 0.955 |
| | LPIPS \downarrow | 0.252 | 0.324 | 0.294 | 0.266 | 0.209 | 0.196 | 0.258 | 0.229 | 0.236 |
| ESLAM [61] | PSNR \uparrow | 29.08 | 25.32 | 27.77 | 29.08 | 33.71 | 30.20 | 28.09 | 28.77 | 29.71 |
| | SSIM \uparrow | 0.929 | 0.875 | 0.902 | 0.932 | 0.960 | 0.923 | 0.943 | 0.948 | 0.945 |
| | LPIPS \downarrow | 0.239 | 0.313 | 0.298 | 0.248 | 0.184 | 0.228 | 0.241 | 0.196 | 0.204 |
| SpliTAM [16] | PSNR \uparrow | 34.11 | 32.86 | 33.89 | 35.25 | 38.26 | 39.17 | 31.97 | 29.70 | 31.81 |
| | SSIM \uparrow | 0.968 | 0.978 | 0.969 | 0.979 | 0.977 | 0.978 | 0.969 | 0.949 | 0.949 |
| | LPIPS \downarrow | 0.102 | 0.072 | 0.103 | 0.081 | 0.092 | 0.093 | 0.102 | 0.121 | 0.152 |
| Semantic SLAM | | | | | | | | | | |
| SNI-SLAM [11] | PSNR \uparrow | 29.43 | 25.91 | 28.17 | 29.15 | 31.85 | 30.34 | 29.13 | 28.75 | 30.97 |
| | SSIM \uparrow | 0.921 | 0.884 | 0.900 | 0.921 | 0.935 | 0.925 | 0.930 | 0.932 | 0.936 |
| | LPIPS \downarrow | 0.237 | 0.307 | 0.292 | 0.265 | 0.185 | 0.211 | 0.230 | 0.209 | 0.198 |
| SGS-SLAM [67] | PSNR \uparrow | 34.66 | 32.50 | 34.25 | 35.10 | 38.54 | 39.20 | 32.90 | 32.05 | 32.75 |
| | SSIM \uparrow | 0.973 | 0.976 | 0.978 | 0.981 | 0.984 | 0.980 | 0.967 | 0.966 | 0.949 |
| | LPIPS \downarrow | 0.096 | 0.070 | 0.094 | 0.070 | 0.086 | 0.087 | 0.101 | 0.115 | 0.148 |
| SemGauss-SLAM [68] | PSNR \uparrow | 35.03 | 32.55 | 33.32 | 35.15 | 38.39 | 39.07 | 32.11 | 31.60 | 35.00 |
| | SSIM \uparrow | 0.982 | 0.979 | 0.970 | 0.987 | 0.989 | 0.972 | 0.978 | 0.972 | 0.978 |
| | LPIPS \downarrow | 0.062 | 0.055 | 0.054 | 0.045 | 0.048 | 0.046 | 0.069 | 0.078 | 0.093 |
| Hier-SLAM [25] | PSNR \uparrow | 35.70 | 32.83 | 34.68 | 36.33 | 39.75 | 40.93 | 33.29 | 32.48 | 35.33 |
| | SSIM \uparrow | 0.980 | 0.976 | 0.979 | 0.987 | 0.988 | 0.989 | 0.975 | 0.971 | 0.976 |
| | LPIPS \downarrow | 0.067 | 0.060 | 0.063 | 0.052 | 0.050 | 0.049 | 0.083 | 0.081 | 0.094 |
| Hier-SLAM++ (one-hot) | PSNR \uparrow | 35.61 | 32.76 | 34.64 | 36.27 | 39.65 | 40.12 | 33.46 | 32.49 | 35.52 |
| | SSIM \uparrow | 0.980 | 0.978 | 0.979 | 0.987 | 0.987 | 0.985 | 0.977 | 0.971 | 0.977 |
| | LPIPS \downarrow | 0.067 | 0.058 | 0.058 | 0.051 | 0.053 | 0.064 | 0.077 | 0.082 | 0.100 |
| Hier-SLAM++ (binary) | PSNR \uparrow | 35.68 | 32.34 | 34.67 | 36.47 | 39.61 | 40.82 | 33.59 | 32.46 | 35.47 |
| | SSIM \uparrow | 0.981 | 0.975 | 0.979 | 0.988 | 0.987 | 0.990 | 0.978 | 0.971 | 0.976 |
| | LPIPS \downarrow | 0.066 | 0.064 | 0.061 | 0.049 | 0.056 | 0.043 | 0.076 | 0.081 | 0.098 |

TABLE VI
RUNTIME ON REPLICA/R0. BEST RESULTS ARE HIGHLIGHTED AS **FIRST**.

| | Methods | Tracking /Iteration | Mapping /Iteration | Tracking /Frame | Mapping /Frame |
|----------|-----------------------|---------------------|--------------------|-----------------|----------------|
| RTX 4090 | NICE-SLAM [59] | 122.42 | 104.25 | 1.22 | 6.26 |
| | SplaTAM [16] | 44.27 | 50.07 | 1.77 | 3.00 |
| | Hier-SLAM++ (w/o sem) | 18.71 | 22.93 | 0.75 | 1.38 |
| | Hier-SLAM++ (one-hot) | 37.63 | 194.78 | 1.50 | 11.69 |
| | Hier-SLAM++ (binary) | 31.22 | 92.63 | 1.25 | 5.56 |
| L40S | Hier-SLAM [25] | 61.23 | 170.30 | 2.45 | 10.22 |
| | Hier-SLAM++ (flat) | 168.94 | 204.25 | 6.75 | 12.26 |
| | Hier-SLAM++ (one-hot) | 62.21 | 212.62 | 2.49 | 12.76 |
| | Hier-SLAM++ (binary) | 58.91 | 210.65 | 2.36 | 12.64 |

Hier-SLAM++ (w/o sem) represents our proposed system without semantic information.

The units are as follows: Tracking/Iteration (ms), Mapping/Iteration (ms), Tracking/Frame (s), and Mapping/Frame (s).

organized into symbolic nodes, as illustrated by the colored nodes.

From Tab. VII, our method, along with the flat coding version and the proposed one-hot version, both demonstrate superior semantic performance compared to all existing methods. By utilizing hierarchical semantic representation, Hier-SLAM++ (one-hot) achieves a similar semantic performance with less than a 1% difference in mIoU while requiring 4.5x less storage compared to the flat version. The training

time shown in Tab. VI demonstrates that our hierarchical version (Hier-SLAM++ (one-hot)) requires only 37% of the tracking time per frame while maintaining similar frame mapping performance compared to the flat version (Hier-SLAM++ (flat)). The comparable mIoU performance, along with significantly reduced storage and computational cost, clearly demonstrates the effectiveness and efficiency of our proposed method. Furthermore, our binary version (Hier-SLAM++ (binary)) achieves state-of-the-art semantic understanding while achieves the best parameter storage efficiency, reducing storage usage to 63% of the one-hot version and requiring only 20% of the storage compared to the flat version. This reduction in storage usage highlights the efficiency of our proposed method. Since works [11], [67], [68] report mIoU only on a subset of classes, making direct comparison unfair, we follow the same evaluation protocol as [67], which computes mIoU using only the visible semantic classes in each frame. Our method achieves an mIoU of 96.79% with a storage usage of 910.50 MB, demonstrating on-par semantic rendering performance with state-of-the-art methods [68], while maintaining efficient storage usage. Meanwhile, given that [68] benefits significantly from a large foundation model pre-trained on much larger and more diverse datasets, our results demonstrate that comparable performance can be

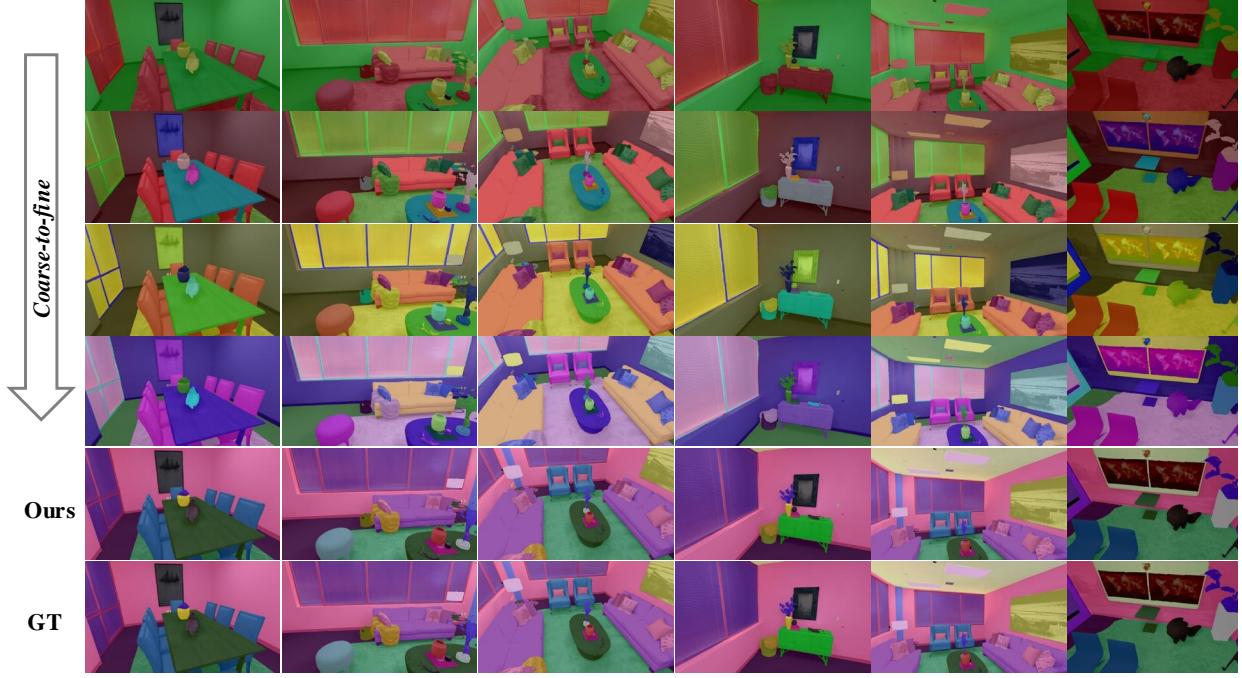


Fig. 5. Visualization of our semantic rendering performance on the Replica [78] dataset. **The first four rows** demonstrate rendered semantic segmentation in a coarse-to-fine manner. **The fifth row** exhibits the finest semantic rendering, equivalent to the flat representation with 102 original semantic classes from the Replica dataset. **The last row** visualizes the semantic ground truth for comparison.

achieved without relying on such heavy models.

Fig. 5 presents our qualitative results on the Replica dataset. We construct a five-level tree to hierarchically encode semantic classes, with the first five rows of Fig. 5 illustrate the transition from level-0 to level-4, progressively refining the semantic understanding from coarse to detailed. At the coarsest level (level-0), shown in the first row, segmentation is categorized into three broad classes: *Small items*, *Medium items*, *Large items*. In contrast, the finest level encompasses all 102 original semantic classes. For example, the hierarchical understanding of the semantic label '*Stool*' progresses from *Medium items* \rightarrow *Seating furniture* \rightarrow *Public seating* \rightarrow *Flat* \rightarrow *Stool*, as demonstrated in the second column. From Fig. 5, we can see that our method accurately renders semantics at each level, enabling a structured coarse-to-fine understanding of the entire scene.

Overall, our method demonstrates superior performance in semantic understanding while significantly reducing storage requirements and training time, benefiting from the proposed hierarchical semantic representation.

G. Scaling up capability

To showcase the scaling-up capability of Hier-SLAM++, we evaluate it on the real-world complex dataset, ScanNet [77], which contains up to 550 distinct semantic classes. In contrast to Replica [78], where the semantic ground truth is synthesized from a global world model and can be considered ideal, the semantic annotations in ScanNet are significantly noisier, containing considerable noise and incorrect or unclear class boundaries. Moreover, the dataset contains noisy depth

TABLE VII
SEMANTIC PERFORMANCE MIoU (%) AND PARAMETER USAGE (MB) ON
REPLICA. RESULTS ARE HIGHLIGHTED AS **FIRST**, **SECOND**.

| | Methods | Avg. | R0 | R1 | R2 | Ofo |
|-------------------------------|--|----------------|--------------|--------------|--------------|--------------|
| mIoU (%) total 102 classes | NIDS-SLAM [12] | 82.37 | 82.45 | 84.08 | 76.99 | 85.94 |
| | DNS-SLAM [7] | 84.77 | 88.32 | 84.90 | 81.20 | 84.66 |
| | Hier-SLAM [25] | 76.44 | 76.62 | 78.31 | 80.39 | 70.43 |
| | Hier-SLAM++ (flat) | 90.35 | 91.21 | 90.62 | 89.11 | 90.45 |
| | Hier-SLAM++ (one-hot) | 89.41 | 86.38 | 89.26 | 91.55 | 90.46 |
| | Hier-SLAM++ (binary) | 81.27 | 82.77 | 73.87 | 89.64 | 78.80 |
| mIoU (%) subset classes | SNI-SLAM [11] | 87.41 | 88.42 | 87.43 | 86.16 | 87.63 |
| | SemGauss-SLAM [68] | 96.34 | 96.30 | 95.82 | 96.51 | 96.72 |
| | SGS-SLAM [67] | 92.72 | 92.95 | 92.91 | 92.10 | 92.90 |
| | Hier-SLAM++ (one-hot)[†] | 96.79 | 96.93 | 96.78 | 96.92 | 96.53 |
| | Hier-SLAM++ (binary)[†] | 95.26 | 96.21 | 92.62 | 96.34 | 95.85 |
| Param (MB) | Hier-SLAM [25] | 910.50 | 793 | 1126 | 843 | 880 |
| | Hier-SLAM++ (flat) | 2662.25 | 2355 | 3072 | 2560 | 2662 |
| | Hier-SLAM++ (one-hot) | 927.50 | 814 | 1126 | 867 | 903 |
| | Hier-SLAM++ (binary) | 591.75 | 528 | 690 | 563 | 586 |

Ours[†] represents our method with a hierarchical representation, evaluated on a subset of semantic classes, consistent with [67].

sensor inputs and blurred color images, making semantic understanding particularly challenging in this scenes.

Using the flat semantic representation cannot even run successfully due to memory limitations. In contrast, we establish the hierarchical tree with the assistance of LLMs and 3D Generative Models, as introduced in Section-III, which guides the compaction of the coding from the original 550 semantic classes to 73 semantic symbolic codings, resulting in over 7 times reduction in coding usage. As visualized in Fig. 6, our estimated 3D global semantic map at different

TABLE VIII

MONOCULAR QUANTITATIVE RESULTS ON THE TUM RGB-D DATASET.
BEST RESULTS ARE HIGHLIGHTED AS **FIRST**, **SECOND**.

| Sequences | | fr1-desk | | | | fr2-xyz | | | |
|-----------------------|-----------------|-------------|--------|--------|---------|-------------|--------|--------|---------|
| Loop | Method | RMSE (cm) ↓ | PSNR ↑ | SSIM ↑ | LPIPS ↓ | RMSE (cm) ↓ | PSNR ↑ | SSIM ↑ | LPIPS ↓ |
| w/ w/o DSO [31] | DROID-SLAM [84] | 1.80 | - | - | - | 0.50 | - | - | - |
| | ORB-SLAM2 [29] | 2.00 | - | - | - | 0.60 | - | - | - |
| | ORB-SLAM3 [30] | 1.89 | - | - | - | 0.29 | - | - | - |
| | Photo-SLAM [19] | 1.54 | 20.97 | 0.743 | 0.228 | 0.98 | 21.07 | 0.726 | 0.166 |
| w/o Hier-SLAM++ | DSO [31] | 22.4 | - | - | - | 1.10 | - | - | - |
| | MonoGS [17] | 4.15 | 21.02 | 0.703 | 0.362 | 4.79 | 22.21 | 0.728 | 0.288 |
| | Hier-SLAM++ | 6.20 | 21.91 | 0.836 | 0.269 | 3.22 | 24.67 | 0.946 | 0.110 |

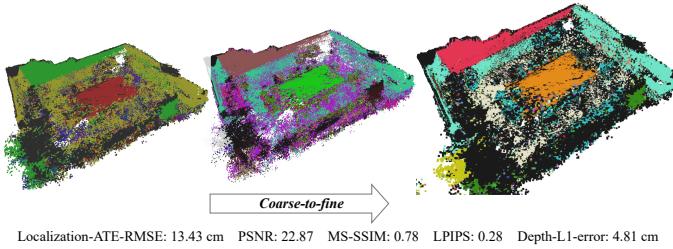


Fig. 6. Visualization of the established semantic 3D map across multiple levels, illustrating a coarse-to-fine semantic understanding of the complex scene. The bottom of the figure presents localization, rendering, and depth performance, offering a comprehensive overview of Hier-SLAM++’s effectiveness and demonstrating the scaling-up capability of our proposed method.

levels demonstrates a coarse-to-fine semantic understanding, showcasing our method’s scaling-up capability in handling real-world complex scenes.

H. Monocular performance

We present our monocular SLAM performance in Tab. VIII. For SLAM methods without loop closure, our approach achieves competitive results compared to state-of-the-art methods in the same setting. For methods with loop closure, where well-executed loop mechanisms typically boost tracking accuracy and overall SLAM performance, our method still delivers better rendering accuracy compared to Gaussian-based SLAM systems [19]. With the assistance of the geometric prior provided by the feed-forward model, the joint optimization of both photometric loss and aligned depth loss contributes to improved rendering performance across the entire SLAM system.

I. Ablation study

In this section, we evaluate the SLAM performance under different hierarchical tree designs. The results are presented in Tab. IX. Specifically, the experiments are conducted using our baseline setup, which adopts a one-hot hierarchical semantic representation and a naïve decoder with a single Softmax layer, denoted as *Baseline* in the table. This setup ensures that any observed performance differences are solely attributable to the tree structure itself. Each metric in Tab. IX represents the average result over two runs on the first four sequences—Room0, Room1, Room2, and Office0—from the Replica dataset. We generate two distinct 5-level hierarchical trees (tree1 and tree2) using our proposed tree construction framework, which integrates both LLM-generated semantics and 3D generative priors, and evaluate them within our semantic SLAM system.

TABLE IX

PERFORMANCE OF DIFFERENT TREES ON REPLICA. BEST RESULTS ARE HIGHLIGHTED AS **FIRST**.

| Methods | mIoU↑ | ATE | RMSE↓ | Depth↓ | PSNR↑ | MS-SSIM↑ | LPIPS↓ |
|------------------|---------------|--------------|--------------|---------------|--------------|--------------|--------|
| Baseline (tree0) | 75.517 | 0.292 | 0.416 | 35.885 | 0.982 | 0.057 | |
| Baseline (tree1) | 75.061 | 0.280 | 0.413 | 35.792 | 0.982 | 0.057 | |
| Baseline (tree2) | 76.754 | 0.302 | 0.408 | 35.825 | 0.982 | 0.058 | |

For comparison, we also include our previous Hier-SLAM [25] (tree0), which relies solely on LLM-generated semantics. As shown in the table, all configurations yield strong performance, which we attribute to the quality of the tree structures produced by the joint LLM and 3D prior-based generation process. In addition, the fully optimized hierarchical semantic loss contributes to the robustness of SLAM performance across tree variants. Among the compared configurations, tree2 achieves the highest semantic mIoU, the best depth reconstruction accuracy, and comparable rendering quality (in PSNR, SSIM, and LPIPS) relative to the alternatives. Therefore, we adopt tree2 as the default hierarchical structure in our semantic SLAM framework.

We also evaluate our proposed one-hot and binary semantic embeddings in previous experiments, including camera tracking performance in Tab. I and Tab. II, reconstruction accuracy in Tab. IV, rendering quality in Tab. V, as well as runtime and semantic understanding in Tab. VI and Tab. VII, respectively. For camera tracking, mapping, and rendering, both embeddings exhibit similar results. However, for runtime and semantic understanding, the two embeddings show differences. Specifically, our one-hot representation achieves better semantic understanding (higher mIoU in Tab. VII), while the binary embedding shows an 8% drop in mIoU on the full semantic class evaluation and less than 1% drop on the subset evaluation, but provides faster operation and significantly reduced storage—approximately 36% savings, as illustrated in Tab. VI and Tab. VII. As introduced in Section III, our proposed semantic embeddings provide two types of compactness: the one-hot representation effectively compresses semantic information, achieving faster training and lower memory usage, while the binary embedding further reduces storage and computation time at the cost of a small decrease in semantic accuracy (which remains minimal on subset evaluations). For tracking, mapping, and rendering, the performance difference between the two embeddings is negligible, as both integrate semantic information into the SLAM process with interpretable compactness derived from our carefully designed hierarchical tree representation.

V. CONCLUSION

We introduced Hier-SLAM++, a neuro-symbolic semantic 3D Gaussian Splatting SLAM system that supports both RGB-D and monocular inputs, incorporating a novel hierarchical categorical representation. Specifically, we proposed a compact and generalized hierarchical representation that encodes both semantic meaning and geometric attributes (i.e., size and shape), leveraging the capabilities of LLMs and 3D

generative models. To enhance global semantic understanding, we introduced a novel hierarchical semantic loss. Furthermore, we extended the system from an RGB-D-only SLAM to support monocular input, utilizing geometric priors derived from a feed-forward SLAM approach. Experimental results demonstrate that Hier-SLAM++ achieves superior or on-par performance with state-of-the-art NeRF-based and Gaussian-based SLAM methods in terms of tracking, mapping, and semantic understanding accuracy, while significantly reducing storage requirements and runtime. These advances establish Hier-SLAM++ as a robust solution for semantic 3D understanding across diverse environments.

REFERENCES

- [1] Lionel Heng, Dominik Honegger, Gim Hee Lee, Lorenz Meier, Petri Tanskanen, Friedrich Fraundorfer, and Marc Pollefeys. Autonomous visual mapping and exploration with a micro aerial vehicle. 31(4):654–675, 2014.
- [2] Henning Lategahn, Andreas Geiger, and Bernd Kitt. Visual slam for autonomous ground vehicles. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1732–1737. IEEE, 2011.
- [3] Denis Chekhlov, Andrew P Gee, Andrew Calway, and Walterio Mayol-Cuevas. Ninja on a plane: Automatic discovery of physical planes for augmented reality using visual slam. In *Proceedings of the IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 1–4. IEEE Computer Society, 2007.
- [4] Javier Civera, Dorian Gálvez-López, Luis Riazuelo, Juan D Tardós, and Jose María Martínez Montiel. Towards semantic slam using a monocular camera. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1277–1284. IEEE, 2011.
- [5] Sean L Bowman, Nikolay Atanasov, Kostas Daniilidis, and George J Pappas. Probabilistic data association for semantic slam. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1722–1729. IEEE, 2017.
- [6] Yun Chang, Yulun Tian, Jonathan P How, and Luca Carlone. Kimera-multi: a system for distributed multi-robot metric-semantic simultaneous localization and mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 11210–11218. IEEE, 2021.
- [7] Kunyi Li, Michael Niemeyer, Nassir Navab, and Federico Tombari. Dns slam: Dense neural semantic-informed slam. *arXiv preprint arXiv:2312.00204*, 2023.
- [8] Antoni Rosinol, Marcus Abate, Yun Chang, and Luca Carlone. Kimera: an open-source library for real-time metric-semantic localization and mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1689–1696. IEEE, 2020.
- [9] Boying Li, Danping Zou, Yuan Huang, Xinghan Niu, Ling Pei, and Wenxian Yu. Textslam: Visual slam with semantic planar text features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.
- [10] Boying Li, Danping Zou, Daniele Sartori, Ling Pei, and Wenxian Yu. Textslam: Visual slam with planar text features. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2102–2108. IEEE, 2020.
- [11] 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 International Conference on Computer Vision and Pattern Recognition*, 2024.
- [12] Yasaman Haghghi, Suryansh Kumar, Jean-Philippe Thiran, and Luc Van Gool. Neural implicit dense semantic slam. *arXiv preprint arXiv:2304.14560*, 2023.
- [13] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):139–1, 2023.
- [14] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, pages 19447–19456, June 2024.
- [15] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, pages 20310–20320, June 2024.
- [16] 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 International Conference on Computer Vision and Pattern Recognition*, 2023.
- [17] Hidenobu Matsuki, Riku Murai, Paul HJ Kelly, and Andrew J Davison. Gaussian splatting slam. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, pages 18039–18048, 2024.
- [18] 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 International Conference on Computer Vision and Pattern Recognition*, pages 19595–19604, 2024.
- [19] 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 *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, pages 21584–21593, 2024.
- [20] Vladimir Yugay, Yue Li, Theo Gevers, and Martin R Oswald. Gaussian-slam: Photo-realistic dense slam with gaussian splatting. *arXiv preprint arXiv:2312.10070*, 2023.
- [21] Tao Lu, Mulin Yu, Lining Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, pages 20654–20664, 2024.
- [22] Yihang Chen, Qianyi Wu, Jianfei Cai, Mehrtash Harandi, and Weiyao Lin. Hac: Hash-grid assisted context for 3d gaussian splatting compression. *Proceedings of the European conference on computer vision*, 2024.
- [23] Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, Tatiana Tommasi, Daniele Sirigatti, Vladislav Rosov, Angela Dai, and Matthias Nießner. Meshgpt: Generating triangle meshes with decoder-only transformers. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, pages 19615–19625, 2024.
- [24] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, pages 20697–20709, 2024.
- [25] Boying Li, Zhixi Cai, Yuan-Fang Li, Ian Reid, and Hamid Rezatofighi. Hier-slam: Scaling-up semantics in slam with a hierarchically categorical gaussian splatting. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2025.
- [26] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. 29(6):1052–1067, 2007.
- [27] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *IEEE and ACM international symposium on mixed and augmented reality*, pages 225–234. IEEE, 2007.
- [28] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. 31(5):1147–1163, 2015.
- [29] Raul Mur-Artal and Juan D Tardos. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. 33(5):1255–1262, 2017.
- [30] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. Orb-slam3: An accurate open-source library for visual, visual-inertial, and multimap slam. 37(6):1874–1890, 2021.
- [31] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. 40(3):611–625, 2017.
- [32] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 15–22. IEEE, 2014.
- [33] Christian Forster, Zichao Zhang, Michael Gassner, Manuel Werlberger, and Davide Scaramuzza. Svo: Semidirect visual odometry for monocular and multicamera systems. 33(2):249–265, 2016.
- [34] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2320–2327. IEEE, 2011.
- [35] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *IEEE international symposium on mixed and augmented reality*, pages 127–136. Ieee, 2011.
- [36] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d

- reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics*, 36(4):1, 2017.
- [37] Renato F Salas-Moreno, Richard A Newcombe, Hauke Strasdat, Paul HJ Kelly, and Andrew J Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, pages 1352–1359, 2013.
- [38] Dorian Gálvez-López, Marta Salas, Juan D Tardós, and JMM Montiel. Real-time monocular object slam. 75:435–449, 2016.
- [39] Shichao Yang and Sebastian Scherer. Cubeslam: Monocular 3-d object slam. 35(4):925–938, 2019.
- [40] Shichao Yang and Sebastian Scherer. Monocular object and plane slam in structured environments. *IEEE Robotics and Automation Letters*, 2019.
- [41] Jingming Dong, Xiaohan Fei, and Stefano Soatto. Visual-inertial-semantic scene representation for 3d object detection. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, pages 960–970, 2017.
- [42] Lachlan Nicholson, Michael Milford, and Niko Sünderhauf. Quadricslam: Dual quadrics from object detections as landmarks in object-oriented slam. *IEEE Robotics and Automation Letters*, 4(1):1–8, 2018.
- [43] Muhammad Sualeh and Gon-Woo Kim. Simultaneous localization and mapping in the epoch of semantics: a survey. *International Journal of Control, Automation and Systems*, 17(3):729–742, 2019.
- [44] Shuaifeng Zhi, Michael Bloesch, Stefan Leutenegger, and Andrew J Davison. Scenocode: Monocular dense semantic reconstruction using learned encoded scene representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11776–11785, 2019.
- [45] John McCormac, Ankur Handa, Andrew Davison, and Stefan Leutenegger. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4628–4635. IEEE, 2017.
- [46] Martin Runz, Maud Buffier, and Lourdes Agapito. Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects. In *Proceedings of the IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 10–20. IEEE, 2018.
- [47] Margarita Grinvald, Fadri Furrer, Tonci Novkovic, Jen Jen Chung, Cesar Cadena, Roland Siegwart, and Juan Nieto. Volumetric instance-aware semantic mapping and 3d object discovery. *IEEE Robotics and Automation Letters*, 4(3):3037–3044, 2019.
- [48] A. Rosinol, A. Gupta, M. Abate, J. Shi, and L. Carlone. 3D dynamic scene graphs: Actionable spatial perception with places, objects, and humans. In *Robotics: Science and Systems (RSS)*, 2020.
- [49] Ue-Hwan Kim, Jin-Man Park, Taek-Jin Song, and Jong-Hwan Kim. 3-d scene graph: A sparse and semantic representation of physical environments for intelligent agents. *IEEE transactions on cybernetics*, 50(12):4921–4933, 2019.
- [50] Iro Armeni, Zhi-Yang He, JunYoung Gwak, Amir R Zamir, Martin Fischer, Jitendra Malik, and Silvio Savarese. 3d scene graph: A structure for unified semantics, 3d space, and camera. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5664–5673, 2019.
- [51] Johanna Wald, Helisa Dhamo, Nassir Navab, and Federico Tombari. Learning 3d semantic scene graphs from 3d indoor reconstructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3961–3970, 2020.
- [52] Shun-Cheng Wu, Johanna Wald, Keisuke Tateno, Nassir Navab, and Federico Tombari. Scenagraphfusion: Incremental 3d scene graph prediction from rgb-d sequences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7515–7525, 2021.
- [53] Chenyangguang Zhang, Alexandros Delitzas, Fangjinhua Wang, Ruida Zhang, Xiangyang Ji, Marc Pollefeys, and Francis Engelmann. Open-vocabulary functional 3d scene graphs for real-world indoor spaces. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 19401–19413, 2025.
- [54] Tarek R Besold, Sebastian Bader, Howard Bowman, Pedro Domingos, Pascal Hitzler, Kai-Uwe Kühnberger, Luis C Lamb, Priscila Machado Vieira Lima, Leo de Penning, Gadi Pinkas, et al. Neural-symbolic learning and reasoning: A survey and interpretation 1. In *Neuro-symbolic artificial intelligence: The state of the art*, pages 1–51. IOS press, 2021.
- [55] Chen Wang, Kaiyi Ji, Junyi Geng, Zhongqiang Ren, Taimeng Fu, Fan Yang, Yifan Guo, Haonan He, Xiangyu Chen, Zitong Zhan, et al. Imperative learning: A self-supervised neural-symbolic learning framework for robot autonomy. *The International Journal of Robotics Research*, 2024.
- [56] 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.
- [57] Lin Yen-Chen, Pete Florence, Jonathan T Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. inferf: Inverting neural radiance fields for pose estimation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1323–1330. IEEE, 2021.
- [58] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J Davison. imap: Implicit mapping and positioning in real-time. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6229–6238, 2021.
- [59] 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 International Conference on Computer Vision and Pattern Recognition*, pages 12786–12796, 2022.
- [60] Zihan Zhu, Songyou Peng, Viktor Larsson, Zhaopeng Cui, Martin R Oswald, Andreas Geiger, and Marc Pollefeys. Nicer-slam: Neural implicit scene encoding for rgb slam. In *2024 International Conference on 3D Vision (3DV)*, pages 42–52. IEEE, 2024.
- [61] 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 International Conference on Computer Vision and Pattern Recognition*, pages 17408–17419, 2023.
- [62] Erik Sandström, Yue Li, Luc Van Gool, and Martin R Oswald. Point-slam: Dense neural point cloud-based slam. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18433–18444, 2023.
- [63] Hongjia Zhai, Gan Huang, Qirui Hu, Guanglin Li, Hujun Bao, and Guofeng Zhang. Nis-slam: Neural implicit semantic rgb-d slam for 3d consistent scene understanding. *IEEE Transactions on Visualization and Computer Graphics*, 2024.
- [64] Tao Hu, Shu Liu, Yilun Chen, Tiancheng Shen, and Jiaya Jia. Efficientnerf efficient neural radiance fields. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, pages 12902–12911, June 2022.
- [65] Jeffrey Yunfan Liu, Yun Chen, Ze Yang, Jingkang Wang, Sivabalan Manivasagam, and Raquel Urtasun. Real-time neural rasterization for large scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8416–8427, October 2023.
- [66] Linfei Li, Lin Zhang, Zhong Wang, and Ying Shen. Gs3lam: Gaussian semantic splatting slam. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 3019–3027, 2024.
- [67] 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 *Proceedings of the European conference on computer vision*, pages 163–179. Springer, 2024.
- [68] Siting Zhu, Renjie Qin, Guangming Wang, Jiuming Liu, and Hesheng Wang. Semgauss-slam: Dense semantic gaussian splatting slam. *arXiv preprint arXiv:2403.07494*, 2024.
- [69] Gpt-4-turbo, 2024. <https://platform.openai.com/docs/models#gpt-4-turbo-and-gpt-4>.
- [70] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.
- [71] Keisuke Tateno, Federico Tombari, Iro Laina, and Nassir Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, pages 6243–6252, 2017.
- [72] Hongcheng Luo, Yang Gao, Yuhao Wu, Chunyuan Liao, Xin Yang, and Kwang-Ting Cheng. Real-time dense monocular slam with online adapted depth prediction network. *IEEE Transactions on Multimedia*, 21(2):470–483, 2018.
- [73] Tristan Laidlow, Jan Czarnowski, and Stefan Leutenegger. Deepfusion: Real-time dense 3d reconstruction for monocular slam using single-view depth and gradient predictions. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4068–4074. IEEE, 2019.
- [74] Xinchen Ye, Xiang Ji, Baoli Sun, Shenglun Chen, Zhihui Wang, and Haojie Li. Drm-slam: Towards dense reconstruction of monocular slam with scene depth fusion. *Neurocomputing*, 396:76–91, 2020.
- [75] Lokender Tiwari, Pan Ji, Quoc-Huy Tran, Bingbing Zhuang, Saket Anand, and Mannohan Chandraker. Pseudo rgb-d for self-improving monocular slam and depth prediction. In *Proceedings of the European conference on computer vision*, pages 437–455. Springer, 2020.
- [76] Wei Zhang, Tiecheng Sun, Sen Wang, Qing Cheng, and Norbert Haala. Hi-slam: Monocular real-time dense mapping with hybrid implicit fields. *IEEE Robotics and Automation Letters*, 9(2):1548–1555, 2023.

- [77] 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 International Conference on Computer Vision and Pattern Recognition*, July 2017.
- [78] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.
- [79] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgbd slam systems. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2012.
- [80] 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 *Proceedings of the IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 499–507. IEEE, 2022.
- [81] 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 International Conference on Computer Vision and Pattern Recognition*, pages 13293–13302, 2023.
- [82] Thomas Whelan, Michael Kaess, Hordur Johannsson, Maurice Fallon, John J Leonard, and John McDonald. Real-time large-scale dense rgbd slam with volumetric fusion. *The International Journal of Robotics Research*, 34(4-5):598–626, 2015.
- [83] Thomas Whelan, Stefan Leutenegger, Renato F Salas-Moreno, Ben Glocker, and Andrew J Davison. Elasticfusion: Dense slam without a pose graph. In *Robotics: science and systems*, volume 11, page 3. Rome, Italy, 2015.
- [84] Zachary Teed and Jia Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgbd cameras. 34:16558–16569, 2021.

VI. BIOGRAPHY SECTION

Boying Li is a Research Fellow in the Faculty of Information Technology at Monash University, Australia. She received her Ph.D. degree in Information and Communication Engineering from Shanghai Jiao Tong University, Shanghai, China, in 2023, where she was recognized as an Outstanding Graduate. Her research interests include visual SLAM, 3D computer vision, and intelligent robotics.

Vuong Chi Hao is an undergraduate student majoring in computer science at VinUniversity, Vietnam, and a research intern at the Visual and Learning for Autonomous AI Lab at Monash University. His research interests include 3D perception and long-horizon planning.

Peter J. Stuckey is a Professor in the Faculty of Information Technology at Monash University and a project leader at the Data61 CSIRO laboratory. He was recognized as an ACM Distinguished Scientist in 2009 and was awarded the Google Australia Eureka Prize for Innovation in Computer Science in 2010 for his work on lazy clause generation. In 2019, he was elected a Fellow of the Association for the Advancement of Artificial Intelligence. Professor Stuckey is a pioneer in constraint programming, the science of modeling and solving complex combinatorial problems. His research interests include discrete optimization, programming languages, constraint-solving algorithms, bioinformatics, and constraint-based graphics.

Ian Reid Professor Reid is a leading researcher in computer vision and has served as Department Chair at the Mohamed bin Zayed University of Artificial Intelligence (MBZUAI) in the United Arab Emirates. Prior to joining MBZUAI, he was Head of School and Professor of Computer Science at the University of Adelaide, Australia. A Fellow of the Australian Academy of Technological Sciences and Engineering (ATSE) and the Australian Academy of Science (AAS), and a former Rhodes Scholar, Professor Reid held an Australian Laureate Fellowship from 2013–2018 in recognition of his contributions to building Australia’s internationally competitive research capacity. He also served as Deputy Director of the Australian Centre for Robotic Vision from 2014–2021 and, in 2022, received the Australian Computer Society’s Artificial Intelligence Distinguished Researcher Award for his outstanding contributions to research in computer vision and machine learning. Over his 30-year career, Professor Reid has made significant contributions across a wide range of computer vision areas, including active vision, visual SLAM, visual geometry, human motion capture, and visual surveillance. His current research interests converge under the themes of spatial AI and embodied AI, aiming to endow real-world agents with lifelong visual learning capabilities, common-sense reasoning, and spatial awareness through video understanding for real-time robotic decision-making.

Hamid Rezatofighi Dr. Hamid Rezatofighi is an Associate Professor in the Faculty of Information Technology at Monash University, Australia. His research focuses on computer vision, machine learning, and robotics, with notable contributions to robot visual perception in dynamic environments. He was previously an Endeavour Research Fellow at Stanford’s Vision and Learning Lab (SVL) and a Senior Research Fellow at the Australian Institute for Machine Learning (AIML) at the University of Adelaide. Dr. Rezatofighi completed his Ph.D. at the Australian National University in 2015. He has served as an area chair for major conferences such as CVPR, NeurIPS, ECCV, ICCV, IJCAI, and IROS, and as an associate editor for leading journals including IEEE Transactions on Image Processing and the Artificial Intelligence Journal. Over the past four years, he has secured significant research funding, including multiple DARPA projects and an ARC Discovery Project, supporting his ongoing work in computer vision and robotics.

APPENDIX

I. IMPLEMENTATION DETAILS FOR TREE GENERATION

During the hierarchical tree generation process, we utilize LLMs to group flat semantic classes at different levels based on **physical size**, **semantic function**, and to **summarize and refine geometric grouping**, ensuring balanced, meaningful, and descriptive outputs. After generating each tree level, we further employ LLMs as **validators** to examine the results at each level, ensuring that all input categories are fully included with no omissions and that no extra categories are mistakenly introduced.

Size Grouping: The prompt used for the initial grouping based on object physical sizes is illustrated as follows:

Code I.1: Prompt for Size Grouping

You are a smart assistant tasked with dividing the following items into meaningful groups based on their properties. The key requirements are:

Balance: The number of items in each group should be as evenly distributed as possible. A difference of 1 item between groups is acceptable, but larger differences should be avoided.

Meaningfulness: The groups must be meaningful, based on the items' inherent characteristics. Grouping should make logical sense to a human observer.

Descriptive Group Names: Each group must have a clear and descriptive name that reflects its characteristics.

Goal: Cluster items into size groups, based on their typical physical size in scenes. The groups must strictly be by sizes, for example: small, small medium, medium, large, extra large, etc.

Items: {classes input}

Ensure groups are meaningful and provide descriptive group names. Output must follow this JSON format:

```
{
    "<GROUP_1>": ["<ITEM_1>", ...],
    "<GROUP_2>": ["<ITEM_2>", ...],
    ...
}
```

Function Grouping: The prompt adopted for the subsequent semantic function grouping is as follows:

Code I.2: Prompt for Function Grouping

You are a smart assistant tasked with dividing the following items into meaningful groups based on their properties. The key requirements are:

Balance: The number of items in each group should be as evenly distributed as possible. A difference of 1 item between groups is acceptable, but larger differences should be avoided.

Meaningfulness: The groups must be meaningful, based on the items' inherent characteristics. Grouping should make logical sense to a human observer.

Descriptive Group Names: Each group must have a clear and descriptive name that reflects its characteristics.

Goal: Cluster items in the 'size' size group into functionality-based groups. The name of the clusters should not be too specific, it could be as general like *storage*, *furniture*, etc. Ensure that items in each group serve similar purposes or have similar functionalities.

Items: {classes input}

Ensure groups are meaningful and provide descriptive group names. Output must follow this JSON format:

```
{
    "<GROUP_1>": ["<ITEM_1>", ...],
    "<GROUP_2>": ["<ITEM_2>", ...],
    ...
}
```

Summarize and Refine for Geometric Grouping: After grouping with the 3D Generative Model, we employ LLMs

to summarize the clustering results with descriptive labels and balance the nodes within each group. The prompt for LLMs is illustrated as follows:

Code I.3: Prompt for Geometric Grouping Results

The following groups of indoor scene items have been clustered based on their shape: {formatted_clusters}

Instructions: Balance: Ensure the groups are as evenly distributed as possible. A difference of 1 item between groups is acceptable. If needed, move a small number of items from one group to another to achieve balance, feel free to even remove a group if it has only one member (just don't leave any groups empty), ensuring that the groups remain meaningful.

Meaningful Naming: After balancing the groups, assign a descriptive and meaningful name to each group, based on the shared shape characteristic. The name should clearly reflect the shape or geometric property of the items in that group. Always prefer singular shapes names like box, rectangle, soft, flat, etc.

No Duplications: make sure to not repeat any class members, the group names are fine but not the groups' members.

The group names must strictly be by shapes and DO NOT leave any group empty, you could remove it if its empty.

The output must be the same JSON format as below:

```
{
    "<GROUP_1>": ["<ITEM_1>", ...],
    "<GROUP_2>": ["<ITEM_2>", ...],
    ...
}
```

LLM Validators: The prompt used for LLM validator is as following:

Code I.4: Prompt for LLM validator

The following groups of items in a scene have been clustered based on their shape: {formatted_clusters}

Instructions: Balance: Ensure the groups are as evenly distributed as possible. A difference of 1 item between groups is acceptable. If needed, move a small number of items from one group to another to achieve balance, remove a group if it has only one member, ensuring that the groups remain meaningful.

Meaningful Naming: After balancing the groups, assign a descriptive and meaningful name to each group, based on the shared shape characteristic. The name should clearly reflect the shape or geometric property of the items in that group. Always prefer singular shapes names like box, rectangle, soft, flat, etc.

New Group: If it is necessary to create a new group, feel free to do so, DO NOT create any non-original items.

No Duplications: make sure to not repeat any class members, the group names are fine but not the groups' members.

The output must be the same JSON format as below:

```
{
    "<GROUP_1>": ["<ITEM_1>", ...],
    "<GROUP_2>": ["<ITEM_2>", ...],
    ...
}
```

II. GENERATED TREE DEMONSTRATION

We demonstrate the generated hierarchical tree for the Replica dataset [78] in Fig. 1. Based on the semantic categories in Replica, we leverage the capabilities of large language models and 3D generative models to construct a hierarchical tree. This tree represents semantic information as symbolic nodes (illustrated as colored nodes), which are used to produce compact semantic embeddings for each Gaussian primitive.

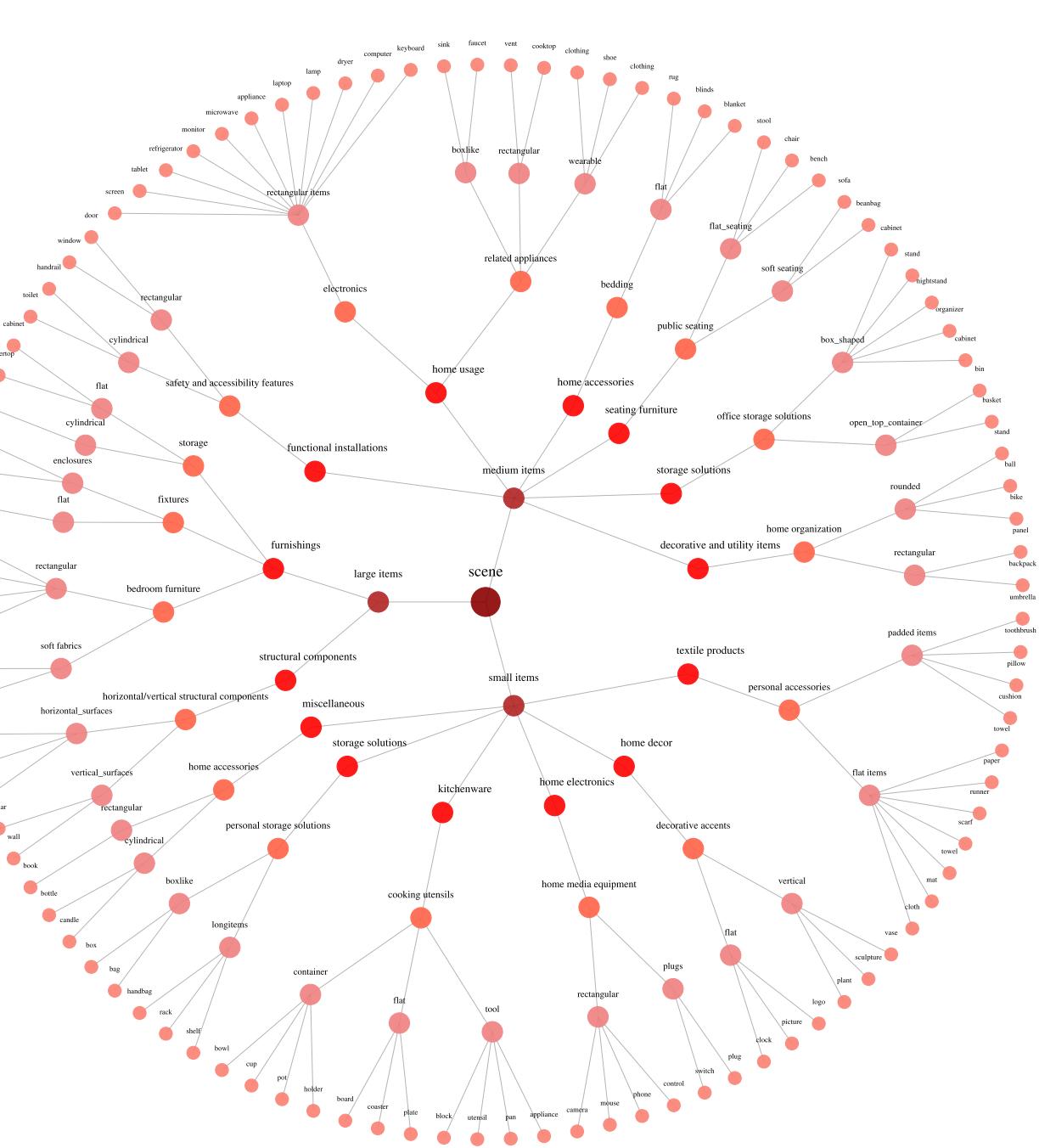


Fig. 1. Visualization of the generated hierarchical tree on the Replica dataset [78]. Different colors represent different tree depths. Each node corresponds to a symbolic concept, with label names shown next to the nodes.

These embeddings are then learned end-to-end across multiple views during SLAM operation.