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

Boying Li[1*], Vuong Chi Hao[2], Peter J. Stuckey[1], Ian Reid[2], and Hamid Rezatofighi[1]

***Abstract*— We propose Hier-SLAM++, a comprehensive Neuro-Symbolic semantic 3D Gaussian Splatting 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 and general 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 a novel semantic loss designed to optimize hierarchical semantic information through both inter-level and cross-level optimization. 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 Splatting 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 NeRF-based and Gaussian-based SLAM systems, while significantly reducing storage and training time requirements.**

I. INTRODUCTION

Visual Simultaneous Localization and Mapping (SLAM) is a fundamental technology for ego-motion estimation and scene perception. Semantic information, which provides high-level environmental knowledge, is essential for comprehensive scene understanding and enabling robots to perform complex tasks. Advancements in image segmentation and enhanced map representations have driven significant progress in semantic visual SLAM [1], [2], [3], [4], [5].

Recently, 3D Gaussian Splatting has gained attention as a leading 3D world representation thanks to its fast rendering and optimization efficiency. It models the *continuous* distributions of geometric parameters through Gaussian distributions, supporting efficient optimization, making it particularly well-suited for SLAM tasks. While promising, current 3D Gaussian Splatting SLAM systems [6], [7] primarily focus on geometric reconstruction. However, the lack of semantic integration limits its potential for complex tasks.

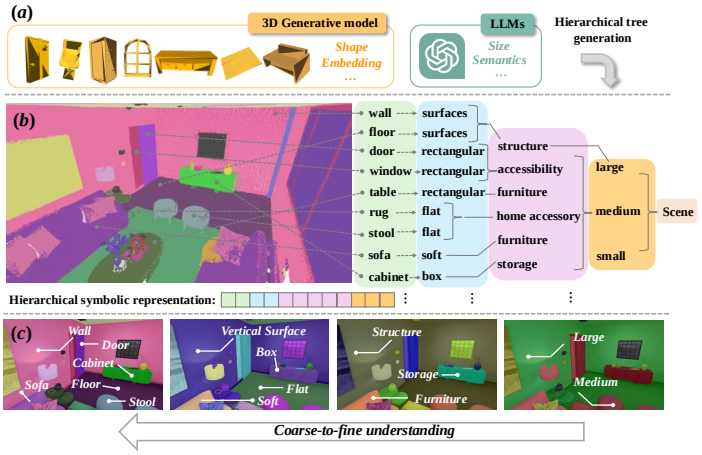

Fig. 1. **(a).** The hierarchical tree is generated by integrating geometric information and semantic messages, utilizing 3D generative models and Large Language Models (LLMs). **(b).** The global 3D Gaussian map generated by Hier-SLAM++ with learned semantic labels is shown on the left. The established 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.

A straightforward approach to integrating semantic parameters is to model their distribution using a categorical (Softmax-based) representation for each 3D primitive. However, this significantly increases parameter usage in semantic SLAM systems, making it impractical for scene understanding. To address this, we observe that semantic information naturally forms a hierarchical structure, as shown in Fig. 1. Attributes and properties of semantic classes are extracted as symbolic nodes to construct a hierarchical tree, where each semantic class is represented as a root-to-leaf path. This hierarchical organization enables efficient encoding of extensive information using a compact set of symbolic nodes, resulting in a more memory-efficient representation. Building on this concept, Hier-SLAM [8] was first introduced as an RGB-D Gaussian Splatting SLAM system that utilizes a hierarchical tree to represent semantic information, achieving strong performance with improved storage efficiency and operational speed.

In this work, we propose Hier-SLAM++, a neuro-symbolic semantic Gaussian Splatting SLAM framework with a hierarchical representation for semantic information. We integrate geometric and semantic information into a hierarchical

[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.

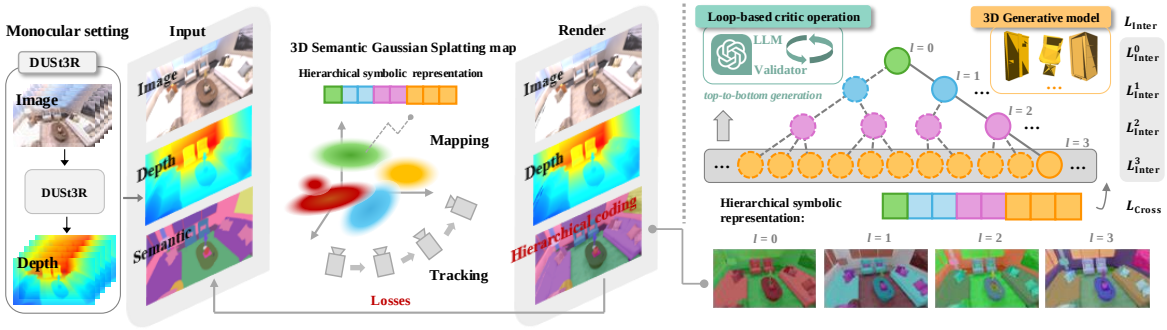

Fig. 2. **Left:** Overview of the Hier-SLAM++ pipeline. The global 3D Gaussian map is initialized using the first frame of the video stream input. The system then alternates between the *Tracking* and *Mapping* steps as new frames are processed (see Section III-D). In the RGB-D setting, depth is directly obtained from sensor input, whereas in the monocular setting the 3D feed-forward method (DUSt3R) is used to generate a geometric prior for depth estimation (see Section III-C). **Top Right:** Hierarchical representation of semantic information. The Tree Generation process leverages the capabilities of both LLMs and 3D Generative Models in a top-to-bottom manner. This hierarchical tree is used to establish a symbolic coding for each Gaussian primitive (see Section III-A). Additionally, we introduce a novel loss function that combines Inter-level Loss L_{Inter} and Cross-level Loss L_{Cross} to optimize the hierarchical semantic representation (see Section III-B). **Bottom Right:** An example of hierarchical semantic rendering.

symbolic tree using Large Language Models (LLMs) and 3D generative models, where semantic nodes along root-to-leaf paths are learned end-to-end during SLAM. To improve semantic understanding, we introduce a hierarchical loss that optimizes inter-level and cross-level relationships, enabling a coarse-to-fine approach beneficial for real-world applications, especially in distant-to-near observations. Our system supports both RGB-D and monocular inputs, leveraging the 3D feed-forward model [9] as a geometric prior to eliminate depth dependency. This allows semantic SLAM without dedicated depth sensors, expanding its applicability. Additionally, we enhance 3D Gaussian Splatting SLAM for improved performance and efficiency. Experiments on synthetic and real-world datasets show that Hier-SLAM++ outperforms or matches state-of-the-art NeRF-based and Gaussian-based SLAM methods, while significantly reducing storage and training time.

II. METHOD

The entire pipeline of our method is illustrated in Fig. 2. We model semantic space as a hierarchical tree, where semantic information is structured from root-to-leaf. To construct the tree, we integrate semantic and geometric information using LLMs and 3D Generative Models. Each 3D Gaussian primitive is then augmented with a hierarchical semantic embedding, learned end-to-end via inter-level and cross-level semantic losses. Additionally, our method supports monocular input by leveraging geometric priors from a feed-forward model, removing the need for depth sensors. The following sections detail each component.

A. Hierarchical representation

Tree Parametrization. The hierarchical tree $\boldsymbol{G}$ is represented as the set of vertices and edges, $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$. The vertices set $\boldsymbol{V} = \cup_{l=0}^{L} \{\boldsymbol{v}_l\}$ comes from the classes from all tree levels, where $\{\boldsymbol{v}_l\}$ represents the set of nodes at the l-th level of the tree, and we use L to represent depth of the whole tree. The edge set $\boldsymbol{E} = \cup_{m=0}^{L-1} \{\boldsymbol{e}_m\}$ captures the subordinate relationships, reflecting semantic attribution, size message, and geometric knowledge. For the i-th semantic class g^i, which is treated as a single leaf node in the tree view, its hierarchical expression is:

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

which corresponds to the root-to-leaf path: $g^i = v_0^i \xrightarrow{e_0} v_1^i \xrightarrow{e_1} \cdots \xrightarrow{e_{L-2}} v_{L-1}^i \xrightarrow{e_{L-1}} v_L^i$, as illustrated in Fig. 3. The hierarchical attributes and properties of class g^i are represented by the symbolic node information v_l^i, while its relationships are represented by the edge information $e_m^i :\rightarrow$. In this way, every semantic class can be coded in a progressive, symbolic hierarchical manner, incorporating both semantic and geometric perspectives. Moreover, the standard flat representation can be seen as a single-level tree coding from the tree viewpoint.

Tree Generation with LLMs and 3D Generative Model. As shown in Fig. 3, we construct the hierarchical tree using LLMs and 3D generative models. For semantic attributes, we use GPT-4 Turbo [10] to extract functional relationships and descriptive labels, leveraging its strong commonsense reasoning and language capabilities. For geometric information, we employ text-to-3D generative models, which generate generalized 3D objects with accurate shapes from text prompts. Specifically, we adopt MeshGPT [11], which learns 3D symbolic embeddings based on local mesh geometry and topology, making it well-suited for our purpose. However, text-to-3D methods generate objects in a unified coordinate system, lacking size information. To compensate, we derive size attributes from LLMs, ensuring a comprehensive geometric representation for each semantic class. Additionally, we employ a loop-based critic operation using LLMs, where one LLM performs clustering and another acts as a validator. By integrating LLMs and 3D generative models, we construct a well-structured hierarchical tree, where each level encodes distinct class attributes, enhancing global 3D understanding.

B. Hierarchical optimization

Based on the generated hierarchical tree structure, the semantic information is represented as compact hierarchical

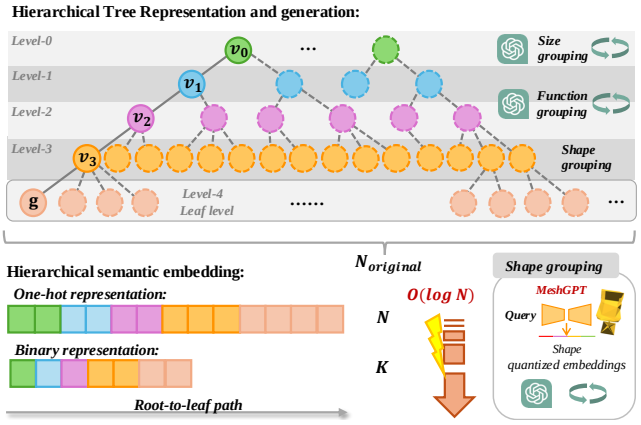

Fig. 3. Visualization of the hierarchical tree and semantic embedding. For **tree representation**, each semantic class is expressed hierarchically as $g = \{v_l, e_m\}$, where edges are depicted as lines connecting tree nodes. The nodes within the same level are illustrated via the same color. For **tree generation**, we utilize both LLMs and 3D Generative Models to extract and group messages. For **hierarchical semantic embedding**, we represent semantic information using a root-to-leaf 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 $O(\log N)$.

symbolic nodes, and further integrated into the 3D Gaussians, learned in an end-to-end manner with the input stream.

Tree Encoding. We propose two types of hierarchical semantic embeddings, both of which compact the original semantic information. **1) One-hot representation:** the hierarchical semantic embedding $\boldsymbol{h}$ is composed of the one-hot embeddings across all levels:

$$\boldsymbol{h} = \mathcal{C}(\boldsymbol{h}_l) \in \mathbb{B}^N, \quad \boldsymbol{h}_l \in \mathbb{B}^n \tag{2}$$

where l represents the l-th level, and $\mathcal{C}$ denotes the concatenation operation. For each level's representation $\boldsymbol{h}_l$, we use n-dimensional one-hot codes (in colored boxes) to encode the symbolic nodes (in colored circles), as shown in Fig. 3. The dimension n corresponds to the maximum number of nodes at each level. And the overall dimension of the semantic embedding is the sum of the dimensions across all levels, given by $N = \sum n$. This allows a maximum reduction of up to $O(\log N)$ compared to the original flat dimension. **2) Binary representation:** a further compact version, the binary representation of the hierarchical semantic embedding $\boldsymbol{b}$ consists of the binary embeddings $\boldsymbol{b}_l$ across all tree levels:

$$\boldsymbol{b} = \mathcal{C}(\boldsymbol{b}_l) \in \mathbb{B}^K, \quad \boldsymbol{b}_l \in \mathbb{B}^k \tag{3}$$

where $\boldsymbol{b}_l$ is the binary representation converted from $\boldsymbol{h}_l$ of each level: $\boldsymbol{h}_l \rightarrow \boldsymbol{b}_l$, as depicted in the last row in Fig. 3. By leveraging binary representation, the hierarchical coding dimension at each level is reduced to $k = \log n$, compared to the one-hot encoding with dimension n. Consequently, the total semantic encoding dimension across all levels is given by $K = \sum k$, resulting in a more compact representation.

Loss Calculation. To fully optimize the hierarchical semantic coding effectively, we propose the hierarchical loss as follows:

$$L_{\text{Semantic}} = \omega_1 L_{\text{Inter}} + \omega_2 L_{\text{Cross}} \tag{4}$$

where L_{Inter} and L_{Cross} stands for the Inter-level loss and Cross-level loss respectively. We use ω_1 and ω_2 to balance the weights between each loss. The Inter-level loss L_{Inter} is employed within each level:

$$L_{\text{Inter}} = \sum_{l=0}^{L} L_{\text{ce}}(\mathcal{S}(\boldsymbol{h}^l), \mathcal{P}^l) \tag{5}$$

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 $\boldsymbol{b}^l$, we replace the original cross-entropy loss with binary cross-entropy loss L_{bce} for the binary Inter-level loss calculation:

$$L_{\text{Inter}}^{\text{bin}} = \sum_{l=0}^{L} L_{\text{bce}}(\mathcal{S}(\boldsymbol{b}^l), \mathcal{P}^l) \tag{6}$$

In contrast, the Cross-level loss is computed based on the entire hierarchical coding. Specifically, we first perform a 2D convolution operation on the semantic embedding, following an activation layer (i.e., ReLU). Then, a second 2D convolution layer is applied to map the hidden embedding into the flat coding, followed by a softmax operation to convert the embeddings into probabilities. The overall Cross-level loss L_{Cross} is defined as:

$$L_{\text{Cross}} = L_{\text{ce}}\big(\mathcal{S}(\text{decoder}(\boldsymbol{h})), \mathcal{P}\big), \tag{7}$$

where $\mathcal{P}$ represents the semantic ground truth, and the definition of decoder$(\boldsymbol{h})$ is:

$$\text{decoder}(\boldsymbol{h}) = \text{Conv}\big(\text{ReLU}(\text{Conv}(\boldsymbol{h}))\big) \tag{8}$$

C. Monocular setting with geometric priority

Accurate depth information is crucial for global map reconstruction and pose estimation in SLAM. In RGB-D settings, depth is directly obtained from sensors, whereas monocular SLAM [12], [13], [14], [15] relies on triangulation, often leading to lower accuracy due to the coupling of pose estimation and mapping. In Hier-SLAM++, we incorporate a feed-forward geometric prior using DUSt3R [9], which generates 3D point maps and pose estimations from sparse-view images, enhancing monocular SLAM accuracy.

D. Semantic Gaussian Splatting SLAM

The global semantic 3D Gaussian Splatting SLAM system is illustrated in the left part of Fig. 2. Based on the established semantic 3D Gaussian representation, global 3D reconstruction (Mapping) and pose estimation (Tracking) are alternately performed with the video stream input. For the RGB-D setting, depth information for each image frame is obtained directly from depth sensors. For the monocular setting, we employ a feed-forward method to obtain the geometric prior. Following [8], every semantic 3D Gaussian primitive within the current global map is projected to the 2D image space using the tile-based differentiable α-compositing rendering. The semantic map H is rendered as follows:

$$H = \sum_{i=1}^{n} \boldsymbol{h}_i \alpha_i(\boldsymbol{X}) T_i \quad \text{with} \quad T_i = \prod_{j=1}^{i-1} (1 - \alpha_j(\boldsymbol{X})) \tag{9}$$

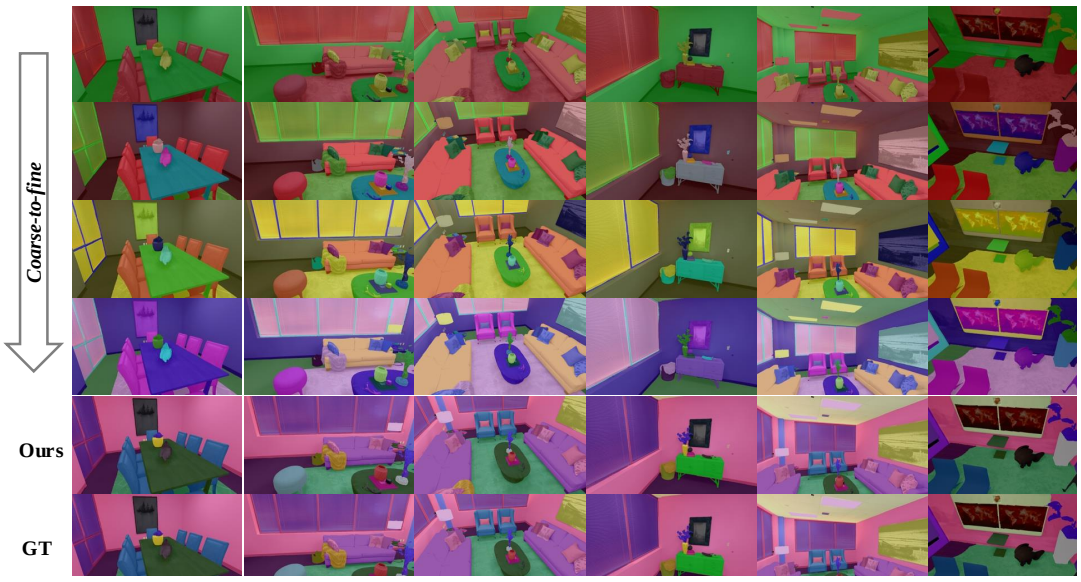

Fig. 4. Visualization of our semantic rendering performance on the Replica [16] dataset. **The first four rows** demonstrate rendered semantic segmentation in a coarse-to-fine manner. **The fifth row** exhibits the finest semantic rendering, equivalent to the flat representation with 102 original semantic classes from the Replica dataset. **The last row** visualizes the semantic ground truth for comparison.

where $\boldsymbol{X}$ stands for the 3D point. Different from previous work [6], which uses separate forward and backward Gaussian modules for different parameters, our Hier-SLAM++ adopt a unified forward and backward modules that processes all parameters, including semantics, color, depth, and silhouette images, significantly improving overall efficiency.

III. Experiments

The experiments are conducted on both synthetic and real-world datasets, including 6 scenes from ScanNet [17] and 8 scenes from Replica [16]. We evaluate SLAM tracking, mapping, and rendering performance, along with runtime and 2D semantic rendering accuracy, to provide a comprehensive assessment of our proposed Hier-SLAM++.

SLAM Performance: For the **tracking performance** evaluated on Replica dataset, our proposed method surpasses all current approaches, as shown in Tab. A.1. For ScanNet [17] dataset, our method (Avg. $\downarrow$: 11.72) performs comparably to state-of-the-art methods [6], [18], including SplaTAM (Avg. $\downarrow$: 11.88) and NICE-SLAM (Avg. $\downarrow$: 10.70) , with differences primarily attributed to the limited quality of sensor inputs from the dataset. We also evaluate **mapping performance** using the L1 depth loss on the Replica [16]. The results show that our method (Avg.$\downarrow$: 0.48) performs comparably to state-of-the-art methods. For **rendering quality**, our method (PSNR $\uparrow$: 35.61, SSIM $\uparrow$: 0.980, LPIPS $\downarrow$: 0.067) demonstrates improved performance compared to all existing methods, including the non-semantic SLAM approaches.

Running time: As shown in Tab. A.2, our method achieves up to 2.4× faster tracking and 2.2× faster mapping than the SOTA method, SplaTAM [6]. When incorporating semantic information, our method remains efficient, leveraging hierarchical semantic coding to achieve nearly 3× faster tracking compared with the semantic SLAM with flat semantic coding. Notably, our Hier-SLAM++ achieves **a rendering speed of 2000 FPS**. For Hier-SLAM++ without semantic information, **the rendering speed increases to 3000 FPS**.

Semantic understanding: We evaluate **semantic rendering segmentation** on the Replica [16]. From Tab. A.3, the results show that our method is on par with state-of-the-art methods while achieving better storage efficiency than the flat version. From visualization in Fig. 4, we observe that our method achieves precise semantic rendering at each level, providing a comprehensive coarse-to-fine semantic understanding for overall scenes. We further conduct experiments on the real-world ScanNet dataset [17], which includes up to 550 unique semantic classes. As shown in Fig. A.1, our estimated 3D global semantic map at different levels demonstrates coarse-to-fine semantic understanding, highlighting our scaling-up capability in complex scenes.

IV. Conclusions and Future Work

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. Experiments show 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 advancements establish Hier-SLAM++ as a robust solution for semantic 3D understanding across diverse environments. Looking ahead, we aim to further enhance semantic understanding by leveraging foundation models.

REFERENCES

[1] 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.

[2] Kunyi Li, Michael Niemeyer, Nassir Navab, and Federico Tombari. Dns slam: Dense neural semantic-informed slam. *arXiv preprint arXiv:2312.00204*, 2023.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] Gpt-4o-turbo, 2024. `https://platform.openai.com/docs/models#gpt-4-turbo-and-gpt-4`.

[11] 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.

[12] 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.

[13] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. 33(5):1255–1262, 2017.

[14] 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.

[15] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. 40(3):611–625, 2017.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] Siting Zhu, Renjie Qin, Guangming Wang, Jiuming Liu, and Hesheng Wang. Semgauss-slam: Dense semantic gaussian splatting slam. *arXiv preprint arXiv:2403.07494*, 2024.

[26] Mingrui Li, Shuhong Liu, and Heng Zhou. Sgs-slam: Semantic gaussian splatting for neural dense slam. *arXiv preprint arXiv:2402.03246*, 2024.

[27] Yasaman Haghighi, Suryansh Kumar, Jean-Philippe Thiran, and Luc Van Gool. Neural implicit dense semantic slam. *arXiv preprint arXiv:2304.14560*, 2023.

APPENDIX

The detailed results for tracking performance, runtime, and semantic performance are presented in Tab. A.1, Tab. A.2, and Tab. A.3, showing that our method achieves superior or on-par performance compared to state-of-the-art approaches. Additionally, the visualization of the global coarse-to-fine semantic mapping is shown in Fig. A.1, demonstrating the scaling-up capability of our method.

TABLE A.1
RGB-D TRACKING PERFORMANCE ATE RMSE (CM) ON THE REPLICA. BEST RESULTS ARE HIGHLIGHTED AS **FIRST**, SECOND.

Methods	Avg.	R0	R1	R2	Of0	Of1	Of2	Of3	Of4
iMap [19]	4.15	6.33	3.46	2.65	3.31	1.42	7.17	6.32	2.55
NICE-SLAM [18]	1.07	0.97	1.31	1.07	0.88	1.00	1.06	1.10	1.13
Vox-Fusion [20]	3.09	1.37	4.70	1.47	8.48	2.04	2.58	1.11	2.94
co-SLAM [21]	1.06	0.72	0.85	1.02	0.69	0.56	2.12	1.62	0.87
ESLAM [22]	0.63	0.71	0.70	0.52	0.57	0.55	0.58	0.72	0.63
Point-SLAM [23]	0.52	0.61	0.41	0.37	0.38	0.48	0.54	0.69	0.72
MonoGS-RGBD [7]	0.79	0.47	0.43	0.31	0.70	0.57	0.31	0.31	3.2
SplaTAM [6]	0.36	0.31	0.40	0.29	0.47	0.27	0.29	0.32	0.55
SNI-SLAM [24]	0.46	0.50	0.55	0.45	0.35	0.41	0.33	0.62	0.50
DNS SLAM [2]	0.45	0.49	0.46	0.38	0.34	0.35	0.39	0.62	0.60
SemGauss-SLAM [25]	0.33	0.26	0.42	0.27	0.34	0.17	0.32	0.36	0.49
SGS-SLAM [26]	0.41	0.46	0.45	0.29	0.46	0.23	0.45	0.42	0.55
Hier-SLAM [8]	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

TABLE A.2
RUNTIME ON REPLICA/R0. BEST RESULTS ARE HIGHLIGHTED AS **FIRST**.

	Methods	Tracking /Iteration	Mapping /Iteration	Tracking /Frame	Mapping /Frame
RTX 4090	NICE-SLAM [18]	122.42	104.25	1.22	6.26
	SplaTAM [6]	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
L40S	Hier-SLAM [8]	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).*

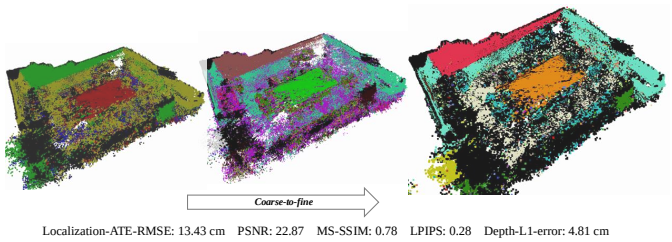

Fig. A.1. Visualization of the established semantic 3D map across multiple levels, illustrating a coarse-to-fine semantic understanding of the complex scene. The bottom of the figure presents localization, rendering, and depth performance, offering a comprehensive overview of Hier-SLAM++'s effectiveness and demonstrating the scaling-up capability of our proposed method.

TABLE A.3
SEMANTIC PERFORMANCE MIOU (%) AND PARAMETER USAGE (MB) ON REPLICA. RESULTS ARE HIGHLIGHTED AS FIRST, SECOND.

	Methods	Avg.	R0	R1	R2	Of0
mIoU (%) total 102 classes	NIDS-SLAM [27]	82.37	82.45	84.08	76.99	85.94
	DNS-SLAM [2]	84.77	88.32	84.90	81.20	84.66
	Hier-SLAM [8]	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 [24]	87.41	88.42	87.43	86.16	87.63
	SemGauss-SLAM [25]	96.34	96.30	95.82	96.51	96.72
	SGS-SLAM [26]	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 [8]	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 [26].*