IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Have We Scene It All? Scene Graph-Aware Deep Point Cloud Compression

Nikolaos Stathoulopoulos, Christoforos Kanellakis and George Nikolakopoulos

***Abstract*—Efficient transmission of 3D point cloud data is critical for advanced perception in centralized and decentralized multi-agent robotic systems, especially nowadays with the growing reliance on edge and cloud-based processing. However, the large and complex nature of point clouds creates challenges under bandwidth constraints and intermittent connectivity, often degrading system performance. We propose a deep compression framework based on semantic scene graphs. The method decomposes point clouds into semantically coherent patches and encodes them into compact latent representations with semantic-aware encoders conditioned by Feature-wise Linear Modulation (FiLM). A folding-based decoder, guided by latent features and graph node attributes, enables structurally accurate reconstruction. Experiments on the SemanticKITTI and nuScenes datasets show that the framework achieves state-of-the-art compression rates, reducing data size by up to 98% while preserving both structural and semantic fidelity. In addition, it supports downstream applications such as multi-robot pose graph optimization and map merging, achieving trajectory accuracy and map alignment comparable to those obtained with raw LiDAR scans.**

***Index Terms*—Range Sensing, Deep Learning Methods, Localization, Point Cloud Compression, Semantic Scene Graphs**

I. Introduction

In recent years, robotic perception has shifted from image-based representations to rich, three-dimensional point clouds, driven by the need for greater robustness in complex, unstructured and ill-illuminated environments. At the same time, multi-robot systems are proliferating, and both centralized coordination and peer-to-peer collaboration increasingly rely on transmitting 3D data to edge or cloud servers for joint processing [1]. The advent of high-bandwidth 5G networks and edge-computing platforms promises lower latency and higher throughput [2], yet the sheer volume and irregular structure of point clouds still impose prohibitive demands on storage, transmission, and real-time responsiveness.

Point clouds are fundamentally large since each sweep can contain thousands of spatial samples and they lack the regular grid structure that makes images compliant to classical compression schemes. Under realistic bandwidth constraints and with intermittent connections, naive streaming of full-resolution clouds leads to dropped frames, degraded map accuracy [3], and ultimately failures in perception-driven tasks such as obstacle avoidance or cooperative mapping [4]. While

This paper was supported in part and received funding from the European Union's Horizon Europe Research and Innovation Programme under the Grant Agreement No.101138451.

The authors are with the Robotics and AI Group, Department of Computer, Electrical and Space Engineering, Luleå University of Technology, 971 87 Luleå, Sweden.Corresponding Author's e-mail: niksta@ltu.se

The code implementation and pre-trained model weights of this letter are available at: https://github.com/LTU-RAI/sga-dpcc.git.

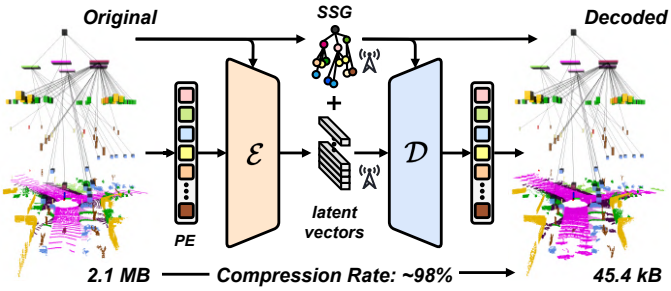

Fig. 1. **Overview.** A raw point cloud is first converted into a semantic scene graph (SSG), capturing object- and layer-level structure. The patch extractor (PE) then subdivides the scene into layer-specific patches, which are encoded by a transformer-based autoencoder into compact, per-patch latent vectors. These are later decoded to reconstruct the full point cloud. The proposed framework achieves extreme compression rates of up to 98%, with the encoded representation consisting solely of the scene graph and the set of latent vectors.

semantic scene graphs have proven effective as compact, task-oriented abstractions for navigation and planning existing compression pipelines rarely exploit their structural information to guide lossy encoding of raw geometry.

While previous works demonstrate the advantages of embedding relational and semantic information in 3D representations, they focus on mapping, localization [5], or scene synthesis [6], [7] rather than data compression. To our knowledge, no existing method has systematically fused graph- or structure-aware abstractions with point cloud compression. In this letter, we close this gap by introducing a scene graph–conditioned autoencoder for point cloud compression. Our framework learns compact latent vectors that jointly preserve geometric fidelity and relational structure, yielding a unified representation that not only enables highly efficient compression but could also further support downstream tasks such as semantic scene understanding, place recognition and beyond.

In more detail, we propose a novel deep compression framework that combines semantic scene graphs with patch-based autoencoding. First, a semantic scene graph generator partitions each LiDAR scan into node patches, each corresponding to a coherent object, agent, infrastructure element, or terrain segment. Each patch is then fed into a semantic-aware encoder where point embeddings and positional encodings are modulated via Feature-wise Linear Modulation (FiLM) layers conditioned on learned class embeddings, and a transformer backbone distills the points into a compact latent vector. On the decoder side, a folding-based upsampler, conditioned on the node's bounding box attributes, reconstructs a high-fidelity point set from each encoded latent vector. By transmitting only these latents and the minimal metadata of the graph structure, our method drastically reduces the communication load.

In sum, the main contributions of this letter are: (a) A graph-conditioned autoencoder for deep point cloud compression. We introduce a novel framework that decomposes

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

raw LiDAR scans into semantically coherent patches using a semantic scene graph, and encodes each patch via a FiLM-conditioned transformer to jointly compress geometric and semantic cues. (b) Integration of relational structure into compression. Our method leverages semantic scene graphs throughout the pipeline, using semantic conditioning in the encoder and attribute-guided folding in the decoder to preserve relational consistency in the compressed representation. (c) Extensive experiments on SemanticKITTI and nuScenes demonstrate state-of-the-art performance, achieving up to 98% data reduction while preserving both geometric and semantic fidelity. Our method outperforms leading codecs such as Draco [8] and the MPEG approach by Mekuria et al. [9], and maintains strong performance in downstream robotic tasks such as multi-agent pose graph optimization and map merging, even under strict bandwidth constraints.

II. Related Work

In this section, we split the related work in two areas. First, we discuss methods for point cloud compression, from classical spatial partitioning to recent learning-based approaches. Second, we highlight works that incorporate relational and structural information into 3D point cloud representations, motivating our scene graph-aware compression framework.

Point Cloud Compression: Classical methods based on spatial partitioning, such as Octrees [10], provide compact representations by recursively subdividing space into occupied and free regions. While effective at reducing memory usage, they often fail to preserve fine-grained geometry at high compression ratios. Beyond this, several approaches exploit structural redundancies. Sun et al. [11] proposed a clustering-based method that segments range images into ground and object regions, followed by region-specific predictive encoding. Feng et al. [12] introduced a spatio-temporal compression framework combining keyframe selection and iterative plane fitting to model static structures across sequences.

More recently, learning-based approaches have further enhanced compression performance. Wiesmann et al. [13] presented a deep convolutional autoencoder that directly operates on dense point cloud maps, using Kernel Point Convolutions (KPConv [14]) for feature extraction and a deconvolution operator for flexible upsampling. Voxel-based and sparse tensor methods have also gained traction, with examples like VoxelContext-Net [15], that augment octree encoding with voxel context modeling through deep entropy estimation, or SparsePCGC [16] which introduced a multiscale sparse tensor framework focused on the most probable occupied voxels.

Regarding range image-based methods, FLiCR [17] compresses 3D LiDAR data by projecting it into 2D range images and applying lossy quantization and downsampling, enabling lightweight real-time operation on resource-constrained platforms. RIDDLE [18] improves compression by modeling local differences between range image pixels through deep delta encoding, significantly reducing redundancy while preserving fine structural details. Moreover, You et al. proposed RENO [19], a real-time neural compression framework that leverages sparse tensor representations for efficient geometry encoding, achieving high compression efficiency while maintaining fast inference suitable for time-critical applications. Finally, semantic-aware compression methods [20] have also emerged, with scene priors integrated into the compression pipeline, jointly optimizing for geometric fidelity and downstream perception tasks.

While most compression approaches focus on geometric preservation and memory reduction, some recent works explore compression jointly with downstream applications. RecNet [21] proposes an invertible point cloud encoding via range image embeddings, enabling efficient multi-robot map sharing, reconstruction and place recognition. This trend towards combining compression with task-driven objectives highlights the potential of designing representations that are not only compact but also structurally and semantically informative.

Graph– and Structure–Aware Representations: Beyond compression, several works focus on enriching point cloud representations with structural, relational, and semantic information. SegMap [5] introduced a segment-based mapping framework, partitioning 3D LiDAR point clouds into local segments and encoding them with compact learned descriptors. While it demonstrates the benefits of structure-aware feature learning for tasks such as global localization and 3D reconstruction, it primarily focuses on segment-level representations and does not explicitly model the relational structure between objects. Other works pursued explicit scene graph construction from 3D data. Armeni et al. [6] proposed 3D Scene Graphs that organize environments into hierarchical semantic and geometric structures, while Gao et al. [7] developed hierarchical scene graphs for indoor scene synthesis, highlighting the role of relational context in modeling complex spaces.

At the network architecture level, point cloud processing has evolved toward dynamic, graph-based methods. DGCNN [22] introduced dynamic edge convolution operations that adaptively capture local geometric structures during learning, while RandLA-Net [23] proposed lightweight local feature aggregation schemes for large-scale outdoor scenes. These advances emphasize the advantages of relational inductive biases for 3D understanding, while object-centric mapping approaches further demonstrate the value of structure-aware modeling.

Existing works focus on scene-aware representations for perception and mapping, but not compression. We leverage scene graph organization during compression to structure the encoding process according to semantic and geometric layers, enabling more efficient and consistent latent representations.

III. The Proposed Framework

Our goal is to compress a point cloud into compact latent representations, significantly reducing data size, while preserving structural and semantic integrity for efficient transmission between robotic agents. Given a point cloud $\mathcal{P} \subset \mathbb{R}^3$, we first apply a semantic segmentation mapping $S : \mathbb{R}^3 \to \mathbb{R}^4$, yielding a labeled point cloud $\bar{\mathcal{P}} = S(\mathcal{P}) \subset \mathbb{R}^4$. We then construct a semantic scene graph, decomposing the cloud into semantically coherent clusters (nodes). Each node is then independently encoded by a semantic-aware encoder conditioned via FiLM [24], generating compact latent vectors. These latent

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

vectors, combined with the scene graph, guide a folding-based decoder to accurately reconstruct the original point cloud.

A. Semantic Scene Graph (SSG) Generation

Given the labeled point cloud $\bar{\mathcal{P}}$, we construct a layered 3D semantic scene graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, capturing both semantic and spatial relationships. Each node $v_i \in \mathcal{V}$ corresponds to a semantic cluster defined by its class l_i, spatial center $\boldsymbol{c}_i \in \mathbb{R}^3$, extent $\boldsymbol{e}_i \in \mathbb{R}^3$, and orientation $\boldsymbol{R}_i \in SO(3)$. We partition the graph into hierarchical layers: (1) *terrain*, (2) *infrastructure*, (3) *objects*, and (4) *agents*. Nodes are created using their class labels and their geometric structure through well-established methods, such as plane or cylinder fitting, region growing, or using instance labels if available. Edges $(v_i, v_j) \in \mathcal{E}$ represent proximity-based or hierarchical connections across layers. Specifically, each frame *(layer 0)* connects to terrain nodes *(layer 1)*, encompassing road, sidewalk, parking, grass and so on. Subsequent layers *(infrastructure*, *objects*, *agents)* connect to terrain nodes based on spatial proximity. For example, moving vehicles typically connect to road nodes, parked vehicles to parking nodes, and vegetation or signs to grass or sidewalk nodes. Nodes without clear terrain associations (e.g., buildings or bushes) connect to a generic "other" *terrain* node. While these connections do not directly influence the compression pipeline's performance, they provide a structured context beneficial for downstream tasks such as map filtering.

B. Autoencoder Architecture

Given a semantic point cloud $\bar{\mathcal{P}}_t$ and its corresponding semantic scene graph $\mathcal{G}_t$, our goal is to compress each semantic node independently using a dedicated, layer-specific autoencoder. Each autoencoder comprises a semantic-aware encoder and a scene graph-conditioned decoder, trained end-to-end with a multi-component loss function. This ensures compact, semantically consistent latent representations.

1) Patch Extraction: To maintain compactness for transmission, 3D points are not stored directly within the graph. Instead, given a node $v_i \in \mathcal{V}$ of the semantic scene graph, we retrieve the corresponding point cloud patch π_i from the labeled point cloud $\bar{\mathcal{P}}$. Specifically, for *non-terrain* nodes, extraction is straightforward, and each node corresponds to a single patch ($v_i \equiv \pi_i$), defined by the node's oriented bounding box with center $\boldsymbol{c}_i$, extent $\boldsymbol{e}_i$, and orientation $\boldsymbol{R}_i$:

$$\pi_i = \{\boldsymbol{x}_j \in \bar{\mathcal{P}} \mid l_j = l_i,\ \boldsymbol{R}_i^\top (\boldsymbol{x}_j - \boldsymbol{c}_i) \in [-\tfrac{\boldsymbol{e}_i}{2}, \tfrac{\boldsymbol{e}_i}{2}]\}. \quad (1)$$

Conversely, for *terrain* nodes, which typically represent larger spatial extents (e.g., road segments), we subdivide the area into multiple smaller patches $\{\pi_{i,k}\}$, ensuring uniform point distribution and sufficient geometric detail. The set of these smaller patches collectively reconstruct the original node, $v_i = \{\pi_{i,1}, \pi_{i,2}, \ldots, \pi_{i,K}\}$. For efficient computation, all extracted patches are uniformly subsampled or padded with masking to a fixed maximum number of points N_{layer} (unique to each layer), ensuring consistent input shapes for downstream processing.

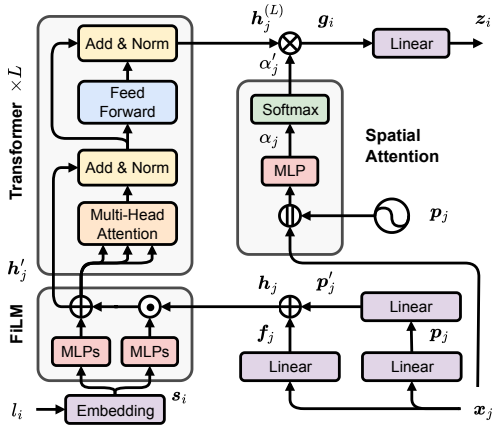

Fig. 2. **Semantic-aware Encoder.** Overview of the proposed semantic-aware encoder, where each patch is processed alongside its semantic class. The right section illustrates the positional encoding module, which maps 3D coordinates to a high-dimensional space and projects them into the feature space. A FiLM module conditions point features using the semantic embedding of the patch, enabling the network to adapt representations based on semantic context. These features are then refined through a series of transformer blocks and pooled via spatial attention to produce a compact latent descriptor.

2) Encoder: Given a patch $\pi_i \in \mathbb{R}^{N\times 3}$ and the corresponding semantic label l_i, the encoder, as seen in Fig. 2, maps it to a latent vector $\boldsymbol{z}_i = \mathcal{E}(\pi_i, l_i)$ through the following stages.

Point-wise Encoding: Each point $\boldsymbol{x}_j \in \mathbb{R}^3$ is first mapped to a higher-dimensional feature space $\mathbb{R}^{D_f}$ via a point-wise Linear layer Λ_θ^f, yielding: $\boldsymbol{f}_j = \Lambda_\theta^f(\boldsymbol{x}_j)$, where $\boldsymbol{f}_j \in \mathbb{R}^{D_f}$ and $j = 1, \ldots, N$. Here θ are the trainable parameters of the embedding. In parallel, each point is passed through a separate Linear layer Λ_θ^p to obtain a positional embedding $\boldsymbol{p}_j \in \mathbb{R}^{D_p}$, which is then projected into the same feature space with a second projection layer $\Lambda_\theta^{p'} : \mathbb{R}^{D_p} \to \mathbb{R}^{D_f}$: $\boldsymbol{p}'_j = \Lambda_\theta^{p'}\left(\Lambda_\theta^p(\boldsymbol{x}_j)\right)$. The final encoded representation per point is the sum of feature and positional embeddings: $\boldsymbol{h}_j = \boldsymbol{f}_j + \boldsymbol{p}'_j$, where $\boldsymbol{h}_j \in \mathbb{R}^{D_f}$.

Semantic Conditioning via FiLM: To inject semantic context, we embed the node-level semantic label l_i using an Embedding layer E_θ^s, producing a semantic descriptor: $\boldsymbol{s}_i = E_\theta^s(l_i)$, where $\boldsymbol{s}_i \in \mathbb{R}^{D_s}$. We condition the per-point features using Feature-wise Linear Modulation (FiLM) [24], which applies a learned affine transformation to each feature dimension: $\boldsymbol{h}'_j = \gamma(\boldsymbol{s}_i) \odot \boldsymbol{h}_j + \beta(\boldsymbol{s}_i)$, where $\gamma(\cdot)$ and $\beta(\cdot)$ are `MLPs` that output scaling and shifting vectors respectively, and $\odot$ denotes element-wise multiplication. This conditioning enables the encoder to adapt its representation based on semantic class, improving generalization across heterogeneous patches.

Transformer-based Encoding: The semantically conditioned features $\boldsymbol{H}^{(0)} = [\boldsymbol{h}'_1, \ldots, \boldsymbol{h}'_N]$ are processed by a sequence of L Transformer blocks, each comprising Multi-Head Self-Attention (`MH-Attn`), Feed-Forward Networks (Φ), and residual connections with Layer Normalization (`Norm`). Formally, at each layer $\ell = 1, \ldots, L$, the update is computed as:

$$\boldsymbol{H}' = \texttt{Norm}\big(\boldsymbol{H}^{(\ell-1)} + \texttt{Drop}\big(\texttt{MH-Attn}\big(\boldsymbol{H}^{(\ell-1)}\big)\big)\big), \quad (2)$$

$$\boldsymbol{H}^{(\ell)} = \texttt{Norm}\big(\boldsymbol{H}' + \texttt{Drop}\big(\Phi_\theta(\boldsymbol{H}')\big)\big), \quad (3)$$

The Multi-Head Self-Attention module [25] computes attention across all point features, enabling rich contextual encoding

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

of intra-patch geometric relationships. Residual connections and dropout are applied at both attention and Feed-Forward stages to improve stability and generalization.

Spatial-Attention Pooling: To aggregate point-wise features into a global patch descriptor, we employ Spatial-Attention pooling. For each point, we compute an attention score via a lightweight MLP, taking as input the concatenation of coordinates and positional encodings: $\alpha_j = \texttt{MLP}_\theta^\alpha(\boldsymbol{x}_j \parallel \boldsymbol{p}_j) \in \mathbb{R}$. The attention weights are then normalized using softmax:

$$\alpha'_j = \frac{\exp(\alpha_j)}{\sum_{k=1}^{N} \exp(\alpha_k)}, \text{ where } j = 1, \ldots, N. \tag{4}$$

The global feature vector $\boldsymbol{g}_i \in \mathbb{R}^{D_f}$ is computed as a weighted sum of the final transformer outputs: $\boldsymbol{g}_i = \sum_{j=1}^{N} \alpha'_j \boldsymbol{h}_j^{(L)}$.

Latent Representation: The aggregated global feature is projected to a compact latent vector through a final Linear layer: $\boldsymbol{z}_i = \Lambda_\theta^z(\boldsymbol{g}_i) \in \mathbb{R}^{D_z}$. This latent representation captures both the geometric and semantic context of the input patch, enabling precise reconstruction by the following decoder.

3) Decoder: The decoder reconstructs a point cloud patch from the latent vector $\boldsymbol{z}_i \in \mathbb{R}^{D_z}$, conditioned on the node's geometric parameters: center $\boldsymbol{c}_i$, extent $\boldsymbol{e}_i$, and orientation $\boldsymbol{R}_i$. As input patches are padded to a fixed size N for batching, the decoder also predicts a confidence mask to indicate valid points, accounting for sparsity due to LiDAR range. We define the decoder function, as seen in Fig. 3, as: $\hat{\pi}_i, \hat{\mu}_i = \mathcal{D}(\boldsymbol{z}_i, \boldsymbol{c}_i, \boldsymbol{e}_i, \boldsymbol{R}_i)$, where $\hat{\pi}_i \in \mathbb{R}^{N\times 3}$ is the reconstructed patch, and $\hat{\mu}_i \in [0,1]^N$ is the per-point confidence mask.

Coarse Point Generation: We begin by generating M coarse patch points $\hat{\pi}_i^{\text{init}} = \{\boldsymbol{x}_m^{\text{init}}\}$ within the node's bounding box, where each point is sampled using a uniform distribution $\mathcal{U}(\boldsymbol{c}_i, \boldsymbol{e}_i, \boldsymbol{R}_i)$. Next, the latent vector $\boldsymbol{z}_i$ is mapped to coarse point features via a feed-forward network comprising a two-layer MLP with ReLU activations: $\boldsymbol{f}_i^c = \Phi_\theta^c(\boldsymbol{z}_i)$, $\boldsymbol{f}_i^c \in \mathbb{R}^{M\times D_f^c}$, where D_f^c denotes the dimensionality of the associated coarse features, and $\boldsymbol{f}_i^c = \{\boldsymbol{f}_m^c\}_{m=1}^M$ with each $\boldsymbol{f}_m^c \in \mathbb{R}^{D_f^c}$ representing the feature of the m-th coarse point. These features are then processed by a separate MLP: $\Delta\boldsymbol{x}_m^c = \texttt{MLP}_\theta^{\text{offset}}(\boldsymbol{f}_m^c)$, $m = 1, \ldots, M$, $\Delta\boldsymbol{x}_m^c \in \mathbb{R}^3$, allowing the model to learn the rough shape of each patch. This uniform initialization provides a structured set of base coordinates, while the latent-conditioned offsets adapt these points to the true geometry, ensuring stable training and faithful reconstruction. The coarse patch points $\hat{\pi}_i^c = \{\boldsymbol{x}_m^c\} \in \mathbb{R}^{M\times 3}$ are formed as: $\hat{\pi}_i^c = \left\{\boldsymbol{x}_m^{\text{init}} + \Delta\boldsymbol{x}_m^c \mid \boldsymbol{x}_m^{\text{init}} \in \hat{\pi}_i^{\text{init}};\ m = 1, \ldots, M\right\}$. Additionally, we predict a coarse confidence mask $\hat{\mu}_i^c$ indicating the validity of each coarse point via a small MLP followed by a sigmoid activation function σ: $\hat{\mu}_i^c = \sigma\left(\texttt{MLP}_\theta^{\text{mask}}(\boldsymbol{f}_i^c)\right)$, $\hat{\mu}_i^c \in [0,1]^M$. This coarse patch defines the initial structure upon which the fine reconstruction is built.

Fine Reconstruction via Folding: To produce a higher resolution output of size N, we employ a folding-based upsampling strategy [26] that refines each of the M coarse points using a structured 2D grid of G positions, such that $N = M \cdot G$. Each coarse point is thus expanded into a local neighborhood of fine points through learned geometric

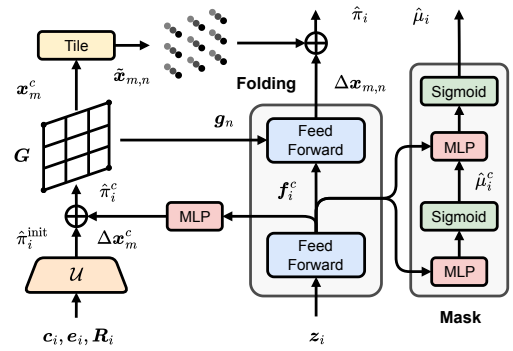

Fig. 3. **Scene graph-conditioned Decoder.** Overview of the proposed decoder conditioned on the semantic scene graph attributes. Given the latent vector of a patch, the decoder generates a set of coarse points using the bounding box and learns per-point offsets. These coarse features are then upsampled via a folding operation guided by a fixed 2D grid, producing a dense reconstruction. A final confidence mask is predicted to prune low-quality outputs.

offsets. Given the set of coarse patch points $\hat{\pi}_i^c$ and their associated features $\boldsymbol{f}_i^c = \{\boldsymbol{f}_m^c\}$, we define a folding grid $\boldsymbol{G}(\hat{\pi}_i^c) = \{\boldsymbol{g}_n\}_{n=1}^{g\times g} \in \mathbb{R}^2$, where $G = g^2$ and each $\boldsymbol{g}_n$ is a 2D coordinate. For each pair of coarse features and grid locations, a shared feed-forward network predicts a 3D offset: $\Delta\boldsymbol{x}_{m,n} = \Phi_\theta^{\text{fold}}(\boldsymbol{f}_m^c, \boldsymbol{g}_n)$, $\Delta\boldsymbol{x}_{m,n} \in \mathbb{R}^3$, where $m = 1, \ldots, M$ and $n = 1, \ldots, G$. Then, we tile the grid $\boldsymbol{G}$ around each coarse point by defining base coordinates: $\tilde{\boldsymbol{x}}_{m,n} = \boldsymbol{x}_m^c$, $\forall\ n = 1, \ldots, G$, and obtain the final fine-level reconstruction by adding the learned offsets: $\hat{\pi}_i = \left\{\tilde{\boldsymbol{x}}_{m,n} + \Delta\boldsymbol{x}_{m,n} \mid m = 1, \ldots, M;\ n = 1, \ldots, G\right\}$. Finally, we predict the per-point confidence mask $\hat{\mu}_i$ using another MLP network, conditioned on the coarse features and their associated confidence scores: $\hat{\mu}_i = \sigma(\texttt{MLP}_\phi^{\text{mask}}(\boldsymbol{f}_i^c, \hat{\mu}_i^c))$, $\hat{\mu}_i \in [0,1]^N$. This upsampling process offers a resolution-aware reconstruction of fine-grained geometries, while maintaining semantic coherence through the structured latent representation and bounding box context.

C. Loss Functions

We train the autoencoders to reconstruct a point cloud as faithful as possible to the input. Given the ground-truth patch π_i with associated valid point mask μ_i, and the predicted reconstructions $\hat{\pi}_i$ (fine) and $\hat{\pi}_i^c$ (coarse) along with the predicted confidence masks $\hat{\mu}_i$ and $\hat{\mu}_i^c$, the loss is defined as:

$$\begin{aligned}\mathcal{L} = D_{\text{CD}}(\pi_i, \hat{\pi}_i) + \lambda_1 D_{\text{CD}}(\pi_i, \hat{\pi}_i^c) + \lambda_2 \mathcal{L}_d(\pi_i, \hat{\pi}_i) \\ + \lambda_3 \mathcal{L}_m(\mu_i, \hat{\mu}_i) + \lambda_4 \mathcal{L}_m(\bar{\mu}_i, \hat{\mu}_i^c),\end{aligned} \tag{5}$$

where $\bar{\mu}_i$ denotes a downsampled version of μ_i aligned to the number of coarse points. The weights λ_1 to λ_4 are scheduled during training, progressively decaying to reduce the influence of the auxiliary regularization terms as optimization focuses more on detailed reconstruction. The first term enforces accurate fine-level reconstruction, while the remaining terms regularize coarse reconstruction accuracy, spatial uniformity, and confidence mask prediction respectively. Each component is detailed below.

Chamfer Distance (D_{CD}): We supervise both the fine-level and coarse-level reconstructions using the Chamfer Distance, which measures the symmetric average of the squared distances between each point and its nearest-neighbor in the other point cloud:

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

$$D_{\text{CD}}(\mathcal{P}_{\text{trg}}, \mathcal{P}_{\text{src}}) = \frac{D_e(\mathcal{P}_{\text{trg}}, \mathcal{P}_{\text{src}})}{2} + \frac{D_e(\mathcal{P}_{\text{src}}, \mathcal{P}_{\text{trg}})}{2}, \quad (6)$$

$$D_e(\mathcal{P}_i, \mathcal{P}_j) = \frac{1}{|\mathcal{P}_i|} \sum_{\boldsymbol{x}_i \in \mathcal{P}_i} \min_{\boldsymbol{x}_j \in \mathcal{P}_j} \underbrace{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}_{d(\boldsymbol{x}_i, \boldsymbol{x}_j)}. \quad (7)$$

Here, $\mathcal{P}_{\text{trg}}$ and $\mathcal{P}_{\text{src}}$ are the sets of target and source 3D points, and $D_e(\mathcal{P}_i, \mathcal{P}_j)$ denotes the mean squared nearest-neighbor distance from $\mathcal{P}_i$ to $\mathcal{P}_j$.

Density Regularization ($\mathcal{L}_d$): To encourage uniform spatial coverage and improve structural similarity, we introduce a voxel-based density loss term. Given the target and source point clouds, we discretize the space into a regular voxel grid and compute occupancy distributions. The density loss is defined as the mean squared error between the normalized occupancy grids of the target and source point clouds:

$$\mathcal{L}_d(\mathcal{P}_{\text{trg}}, \mathcal{P}_{\text{src}}) = \frac{1}{V} \sum_{v=1}^{V} \left(o_v^{(\text{trg})} - o_v^{(\text{src})}\right)^2, \quad (8)$$

where V is the total number of voxels, and $o_v^{(\text{trg})}$ and $o_v^{(\text{src})}$ denote the normalized occupancy values in voxel v for target and source respectively. In sum, minimizing $\mathcal{L}_d$ encourages predicted points to match the global spatial distribution of the ground-truth.

Confidence Mask ($\mathcal{L}_m$): We supervise the predicted confidence mask $\hat{\mu}$ using binary cross-entropy loss against the corresponding ground-truth valid mask μ, defined as:

$$\mathcal{L}_m(\mu, \hat{\mu}) = -\frac{1}{N} \sum_{j=1}^{N} \left[\mu_j \log(\hat{\mu}_j) + (1-\mu_j) \log(1-\hat{\mu}_j)\right], \quad (9)$$

where N denotes the number of points and $\mu_j \in \{0, 1\}$ is the ground-truth validity for point j. Confidence prediction is essential to account for the varying number of points across patches, allowing the network to reconstruct realistic point densities despite the fixed-size padding used for batching.

IV. Experimental Evaluation

We evaluate our method's capability to efficiently and accurately compress a point cloud scan. We compare against the commonly used codecs: Draco [8] and the octree-based compression algorithm from Mekuria et al. [9], which we refer to as "MPEG". Additionally we include the current state-of-the-art approaches RENO [19] and OctAttention [10], as well as RecNet [21], that in contrast to ours, encodes each scan into a single latent vector through a range-image embedding. We note that the last three methods, unlike the proposed approach, or Draco/MPEG, do not encode the semantic classes.

A. Experimental Setup

We benchmark our method on the SemanticKITTI [27] dataset. In order to create the semantic scene graph we use the ground truth semantic labels and in addition we crop each point cloud to 50 meters for all methods, following common practices in the field [28]. Similar to SegMap [5] we use sequences 05 and 06 for training and the rest (00 to 04 and 07 to 10) for the evaluation and comparison to the baselines.

Evaluation Metrics: The quality of a compression algorithm typically involves a trade-off between compression ratio and reconstruction error. To quantify compression, we use the average bits-per-point (bpp) required to store the encoded point cloud. For our method, this includes all latent vectors as well as the semantic scene graph. Following the evaluation protocol in [13], we assess reconstruction error using three metrics. The first is the symmetric point distance D_{CD}, defined earlier in Eq. (6) and Eq. (7), which measures the average nearest-neighbor distance between the reconstructed and ground truth point clouds. However, for many robotic tasks that rely on point-to-plane ICP registration, reconstructing the exact same points is less critical than ensuring that points lie on the correct surfaces. To address this, we also report the symmetric plane distance $D_\perp$, which is computed similarly to D_{CD} but replaces the Euclidean distance $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ with the point-to-plane distance $d_n(\boldsymbol{x}_i, \boldsymbol{x}_j) = |\boldsymbol{n}^\mathsf{T}(\boldsymbol{x}_i - \boldsymbol{x}_j)|$, where $\boldsymbol{n} \in \mathbb{R}^3$ is the ground truth surface normal at the target point, estimated using a 50 cm neighborhood. Finally, we include the mean intersection-over-union *(IoU)* between occupancy grids of the original and reconstructed point clouds, defined as: $IoU = |G_{\text{src}} \cap G_{\text{trg}}| \,/\, |G_{\text{src}} \cup G_{\text{trg}}|$. As in [13], the occupancy grids are computed with a voxel resolution of $20 \times 20 \times 10\,\text{cm}^3$.

Implementation Details: Our method is implemented in PyTorch. We use distributed training on a set of A100 GPUs and train the model for 150 epochs using the Adam optimizer with a learning rate of 5×10^{-4} and a weight decay of 1×10^{-6}. Each layer is constrained to a max number of points per patch, specifically $\{N_1 : 720,\ N_2 : 1720,\ N_3 : 320,\ N_4 : 1536\}$. To vary the compression rate, we evaluate multiple latent vector sizes for each layer. Notably, all other network parameters are kept fixed; only the output dimensionality of each layer is varied across $D_z \in \{8,\ 16,\ 32,\ 64,\ 128\}$. The grid size G is set to 2×2 for an upsampling factor of 4.

B. Compression Results and Qualitative Analysis

We first evaluate the compression performance of our proposed framework in comparison to the established baselines. Our encoding consists of a semantic scene graph whose nodes include both geometric attributes and latent vectors, forming the complete representation required for transmission and decompression. Compression results on SemanticKITTI are presented in the left column of Fig. 4, where we vary the latent dimensionality across layers. Our method consistently outperforms the classical baselines at lower bits-per-point across all three metrics: D_{CD}, $D_\perp$, and IoU. In particular, in the low bitrate regime (< 4 bpp) that is most relevant for bandwidth-limited robotic applications, our approach yields lower D_{CD} and $D_\perp$ and higher IoU compared to Draco, MPEG, and RecNet. While MPEG and Draco degrade significantly at low bitrates, and RecNet remains almost flat due to its single-latent design, our method preserves geometric structure and semantic consistency. The slight plateau in performance at higher bitrates is attributed to the fixed network capacity, since only latent size is varied while other architectural parameters remain constant (e.g., number of transformer blocks).

Compared to recent learned codecs, RENO generally achieves the best performance across most operating points,

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

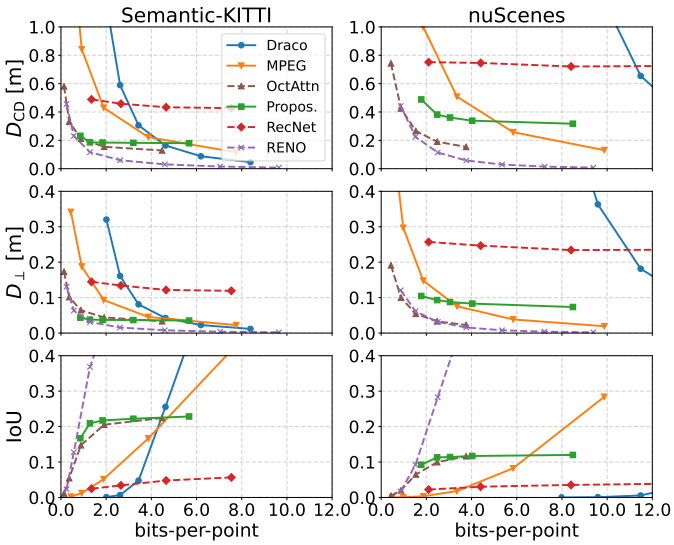

Fig. 4. **Compression results per codec.** The left column presents results on SemanticKITTI (in-dataset evaluation), while the right column shows cross-dataset generalization on nuScenes, where models were trained on SemanticKITTI and applied directly to nuScenes without fine-tuning. Dashed lines indicate methods that do not encode semantic labels.

while our method remains consistently competitive with OctAttention. In terms of efficiency, OctAttention requires roughly 250 s per scan, while our method runs at 0.17 s for encoding and 0.08 s for decoding on a GeForce RTX 4090 GPU (using 1-3 GB of VRAM). The generation of the semantic scene graph currently takes about 0.25 s and remains the main bottleneck in the pipeline. RENO operates in real time (10 FPS). Overall, our approach is competitive with recent learned methods, stronger than the classical codecs and RecNet, and uniquely supports semantic label compression, demonstrating the effectiveness of semantic scene graphs for compression.

Qualitative reconstructions, shown in Fig. 5, further highlight this trend. At low bits-per-point, baseline methods produce sparse outputs that fail to preserve fine structural details, particularly Draco, which struggles to retain any meaningful geometry due to aggressive quantization. MPEG exhibits better preservation but still suffers from blocky artifacts and loss of surface continuity. In contrast, our method reconstructs scenes with high fidelity, preserving both global layout and local geometry, even at extreme compression levels around 1.8 bpp (compared to 60-70 bpp for the original scans). The results shown correspond to latent dimensions $\{D_{z_1}: 16, D_{z_2}: 32, D_{z_3}: 16, D_{z_4}: 32\}$, demonstrating the effectiveness of distributed patch-wise encoding.

C. Generalization Capability

A common challenge with learning-based methods is poor generalization performance, often displaying significant degradation when applied to environments that differ from the training distribution, primarily due to overfitting. We argue that our method is able to generalize by learning local geometric cues within each semantic layer, rather than learning the whole scene. To validate this, we deploy the models trained on SemanticKITTI directly to nuScenes without fine-tuning. This dataset differs substantially from the SemanticKITTI in both sensor configuration (e.g., 32-beam LiDAR with narrower vertical field-of-view and different sensor placement) and geographical context (captured in a different continent, resulting in distinct scene layouts and object appearances). As shown in the right column of Fig. 4, the proposed method maintains better performance compared to baseline approaches at lower bits-per-point. While some degradation in reconstruction accuracy is expected, primarily due to the reduced density of the LiDAR input, our method remains consistently robust across different compression rates. Notably, it also outperforms RecNet, which encodes the entire scene and struggles to generalize to the significantly different distribution of nuScenes. RENO and OctAttention also show strong generalization, with RENO giving the best results overall, while OctAttention remains competitive. In contrast, the baseline methods suffer from drops in both compression efficiency and reconstruction quality, as their dependence on dense local neighborhoods becomes a critical weakness in the sparse and structurally distinct setting of nuScenes.

D. Ablation Studies

To validate our design choices, we conducted an ablation study on two key components of our patch-based autoencoder:

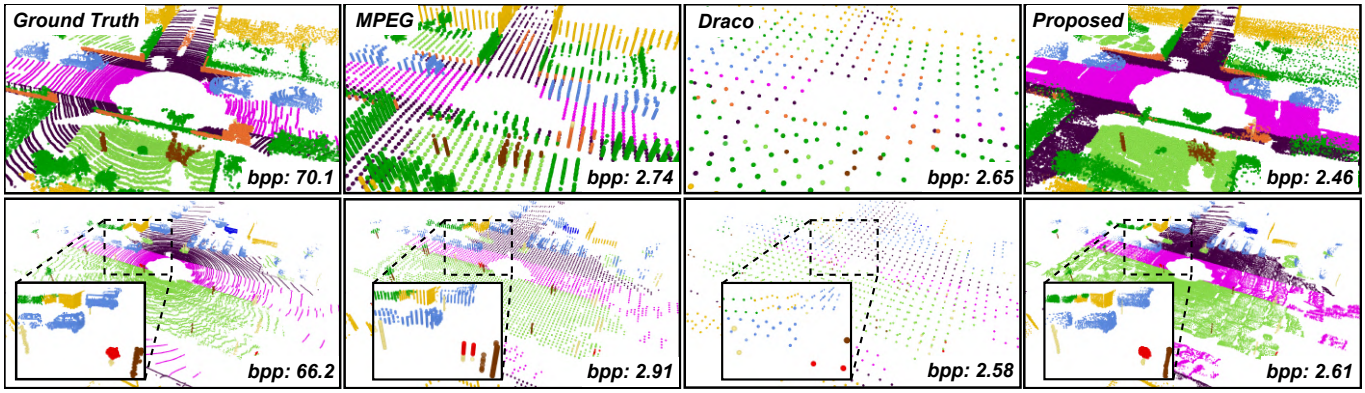

Fig. 5. **Qualitative comparisons.** Qualitative results from the codecs that support semantic label encoding, for two scans of the SemanticKITTI (top: Sequence 00, bottom: Sequence 06), comparing the proposed method with baseline compression algorithms. At low bits-per-point, corresponding to a compression rate of 98%, our method achieves significantly better reconstruction quality in terms of geometric fidelity and scene completeness. For visual clarity, the ground has been removed in the zoomed-in views, which highlight fine-grained structures such as cars, infrastructure, tree trunks, poles, and signage.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

the semantic conditioning via FiLM and the positional encoding. Table I summarizes the performance of each variant across layers 2 and 3 using the metrics introduced in IV-A. Disabling both FiLM and the positional encoding led to the most significant degradation in performance, with Chamfer Distance increasing by approximately 21% and 11% in layers 2 and 3, and IoU dropping by 19% and 25% respectively. This confirms that the two modules provide complementary benefits in enhancing reconstruction fidelity. Ablating each component independently reveals distinct patterns. Removing FiLM caused a noticeable drop in IoU, particularly in layer 3, highlighting its role in enabling semantic-aware reconstructions, especially for object and agent classes. In contrast, omitting the positional encoding resulted in a higher geometric error across both layers, likely due to the loss of localized spatial priors within the encoder. These results support our design rationale that FiLM guides the encoder to modulate features based on class semantics, while the positional encoding improves spatial coherence and geometric precision. Their joint use improves both the structural and semantic quality of the reconstructed patches, and is thus retained in our final model.

We further studied the effect of using predicted instead of ground-truth labels for the proposed pipeline, simulating a realistic deployment. Specifically, we tested RangeNet++ [29], KP-FCNN [14], and RandLA-Net [23] as semantic segmentation front-ends. The results, reported in Table II, show that segmentation accuracy ($mIoU$) is not directly proportional to reconstruction quality. Although RangeNet++ achieves the lowest $mIoU$, it yields the best reconstruction among the three networks, since many of its characteristic *"shadow"* artifacts are filtered out during clustering. In contrast, RandLA-Net, despite comparable $mIoU$, produces larger reconstruction errors due to cluster-level misclassifications, while KP-FCNN performs in between with more localized errors. These findings show that the type and distribution of segmentation errors affect compression performance, offering practical insights for deploying segmentation-compression pipelines in real systems.

TABLE I
ABLATION STUDY ON FILM AND THE POSITIONAL ENCODING (POSENC) ACROSS LAYERS 2 & 3. THE SYMBOL (✔) INDICATES THE COMPONENT IS USED, WHILE THE SYMBOL (✗) INDICATES IT IS ABLATED.

Variant		Layer 2			Layer 3		
FiLM	**PosEnc**	$D_{\text{CD}} \downarrow$	$D_{\perp} \downarrow$	$IoU \uparrow$	$D_{\text{CD}} \downarrow$	$D_{\perp} \downarrow$	$IoU \uparrow$
✗	✗	0.145	0.047	0.266	0.196	0.068	0.159
✔	✗	0.139	0.042	0.275	0.190	0.059	0.178
✗	✔	0.135	0.039	0.294	0.180	0.054	0.185
✔	✔	**0.120**	**0.027**	**0.330**	**0.177**	**0.050**	**0.211**

TABLE II
RECONSTRUCTION PERFORMANCE USING PREDICTED VS. GROUND-TRUTH LABELS. RELATIVE DIFFERENCES ARE REPORTED AS Δ.

Labeling	D_{CD}	$D_{\perp}$	IoU	ΔD_{CD}	$\Delta D_{\perp}$	ΔIoU	mIoU^*
Ground Truth	0.165	0.034	0.237	–	–	–	–
RangeNet++	**0.217**	0.036	**0.184**	**32%**	6%	**22%**	52%
RandLA-Net	0.505	0.038	0.045	206%	11%	81%	53%
KP-FCNN	0.253	**0.035**	0.131	53%	**3%**	45%	**58%**

*The labeling accuracy is presented as the percent mean intersection-over-union over all classes.

TABLE III
PERFORMANCE OF THE BASELINE AND THE PROPOSED (PROP.) COMPRESSION PIPELINE IN DOWNSTREAM ROBOTIC TASKS

Robotic Tasks	**Metrics**	**Raw**	**MPEG**	**Draco**	**Prop.**
Pose Graph Opt. with **GTSAM**	↑ Tran. ATE [%] *	**49.78**	20.23	0.21	37.76
	↑ Rot. ATE [%] *	**23.13**	4.43	0.03	10.04
	↓ Rej. Loops†	**13**	110	123	14
Point cloud transmission at 10Hz	↓ Bandw. [Mb/s]	168	3.73	6.04	**3.51**
	↑ Compres. Rate	N/A	97.8%	96.4%	**97.9%**
Map merging w. **KISS-Matcher**	↓ Tran. RTE [m]	**0.156**	*Failed*	*Failed*	0.649
	↓ Rot. RTE [°]	**0.862**	*Failed*	*Failed*	1.310

*The % improvement in ATE. †Rejected Loops after geometric verification.

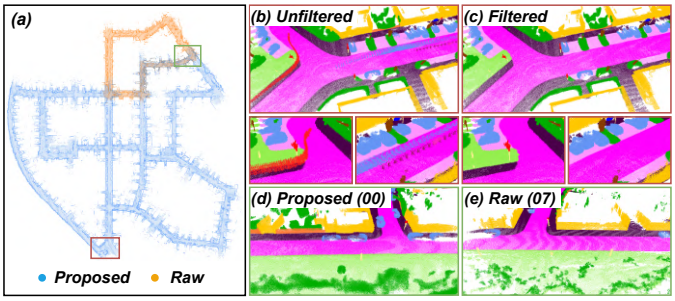

Fig. 6. **Merged maps and zoomed views.** (a) Result of KISS-Matcher [32] on KITTI [27] with the decompressed scans from sequence 00 (using our method) and the raw scans from sequence 07. (b)–(c) Example of the decompressed frames before and after filtering, where dynamic objects (pedestrian, car, motorcyclist) are succesfully removed. (d)–(e) Side-by-side comparison of the overlapping regions: decompressed vs. raw, highlighting preserved structure.

E. Downstream Robotic Tasks

To validate our compression framework beyond standard metrics, we assess its effectiveness in two representative downstream robotic tasks, as described below.

Pose Graph Optimization (PGO): In a multi-agent setup, we simulate point cloud transmission between agents operating on SemanticKITTI sequences 00 and 07, followed by PGO. Each agent estimates an initial trajectory using `KISS-ICP` [28] Loop closure candidates between decompressed and raw scans undergo geometric verification via `GICP` [30]; those with registration error exceeding a predefined threshold are rejected. We evaluate two cases where pose graph edges are obtained by registering raw (00) with decompressed (07) scans and vice versa. Results are averaged over both configurations using the `GTSAM` [31] optimization. Performance is measured by the percent improvement in Absolute Trajectory Error (ATE) relative to ground truth. We also quantify the bandwidth required for transmitting raw vs. compressed scans at 10 Hz. As shown in Table III, MPEG and Draco suffer from high loop rejection and degraded ATE due to poor geometric fidelity, while our method closely follows raw-data trajectory accuracy with roughly 98% bandwidth reduction.

Multi-Robot Map merging: We further evaluate the utility of our compressed representations in multi-robot map merging. Full-session maps are built from SemanticKITTI sequences using: raw-to-raw (00/07), raw 00 with decompressed 07, and the reverse. Map alignment is performed via `KISS-Matcher` [32], with mean Relative Transformation Error (RTE) measured against ground truth (see Fig. 6). Using our scene graph, we apply semantic pruning to remove dynamic obstacles (e.g., pedestrians or vehicles linked to

road terrain) from its layered structure, improving clarity as illustrated in Fig. 6. MPEG and Draco fail to produce reliable correspondences, making them unsuitable for this task, as shown in Table III. In contrast, our method closely follows the alignment quality of raw data while reducing bandwidth by approximately 98%. These results demonstrate that our semantic scene graph-based compression preserves essential geometric and semantic fidelity, while also enabling a practical communication strategy: agents can initially exchange compressed representations, sufficient for tasks such as multi-agent pose graph optimization and map merging, and later, if needed, update with full-resolution scans when higher bandwidth becomes available.

V. Discussion and Conclusions

While our framework achieves strong compression performance, several limitations remain. First, generating the semantic scene graph introduces computational overhead that may hinder real-time deployment, and future work will investigate approximate or accelerated graph construction to reduce this bottleneck. Second, the system depends on a fixed vocabulary from a pre-trained segmentation model, limiting flexibility; integrating open-vocabulary segmentation and dynamic graph generation could enable more adaptable representations. For deployment, feasibility on embedded or edge platforms could be improved through model compression techniques (e.g., pruning, quantization, distillation), parallel or task-specific execution of the layer-specific models, and hardware acceleration on embedded GPUs. These directions would reduce latency and memory requirements and make the approach more practical in real-world systems. In summary, we presented a semantic-aware compression framework that encodes 3D point clouds via patch-wise latent vectors structured by a scene graph. Our method achieves state-of-the-art compression under extreme bit-rate constraints while preserving both geometric and semantic structure, and task-driven evaluations show that the representation is effective for downstream robotic applications.

References

[1] A. S. Seisa, *et al.*, "E-CNMPC: Edge-Based Centralized Nonlinear Model Predictive Control for Multiagent Robotic Systems," *IEEE Access*, vol. 10, pp. 121 590–121 601, 2022.

[2] G. Damigos, *et al.*, "Environmental Awareness Dynamic 5G QoS for Retaining Real Time Constraints in Robotic Applications," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 12 069–12 075.

[3] G. Damigos, *et al.*, "Communication-Aware Control of Large Data Transmissions via Centralized Cognition and 5G Networks for Multi-Robot Map merging," *Journal of Intelligent & Robotic Systems*, vol. 110, no. 1, p. 22, Jan 2024.

[4] N. Stathoulopoulos, *et al.*, "FRAME: A Modular Framework for Autonomous Map Merging: Advancements in the Field," *IEEE Transactions on Field Robotics*, vol. 1, pp. 1–26, 2024.

[5] R. Dubé, *et al.*, "SegMap: Segment-based mapping and localization using data-driven descriptors," *The International Journal of Robotics Research*, vol. 39, no. 2-3, pp. 339–355, 2020.

[6] I. Armeni, *et al.*, "3D Scene Graph: A Structure for Unified Semantics, 3D Space, and Camera," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 5664–5673.

[7] L. Gao, *et al.*, "SceneHGN: Hierarchical Graph Networks for 3D Indoor Scene Generation With Fine-Grained Geometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 7, p. 8902–8919, July 2023.

[8] Google, "Draco 3D Data Compression," 2017. [Online]. Available: https://github.com/google/draco

[9] R. Mekuria, K. Blom, and P. Cesar, "Design, Implementation, and Evaluation of a Point Cloud Codec for Tele-Immersive Video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 4, pp. 828–842, 2017.

[10] C. Fu, *et al.*, "OctAttention: Octree-Based Large-Scale Contexts Model for Point Cloud Compression," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 1, pp. 625–633, Jun. 2022.

[11] X. Sun, *et al.*, "A Novel Point Cloud Compression Algorithm Based on Clustering," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2132–2139, 2019.

[12] Y. Feng, S. Liu, and Y. Zhu, "Real-time spatio-temporal LiDAR point cloud compression," *IEEE International Conference on Intelligent Robots and Systems*, pp. 10 766–10 773, 2020.

[13] L. Wiesmann, *et al.*, "Deep Compression for Dense Point Cloud Maps," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2060–2067, 2021.

[14] H. Thomas, *et al.*, "KPConv: Flexible and Deformable Convolution for Point Clouds," *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[15] Z. Que, G. Lu, and D. Xu, "VoxelContext-Net: An Octree based Framework for Point Cloud Compression," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 6038–6047.

[16] J. Wang, *et al.*, "Sparse Tensor-Based Multiscale Representation for Point Cloud Geometry Compression," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 7, pp. 9055–9071, 2023.

[17] J. Heo, C. Phillips, and A. Gavrilovska, "FLiCR: A Fast and Lightweight LiDAR Point Cloud Compression Based on Lossy RI," in *2022 IEEE/ACM 7th Symposium on Edge Computing (SEC)*, 2022, pp. 54–67.

[18] L. Theis and W. Shi, "RIDDLE: Lidar Data Compression with Range Image Deep Delta Encoding," in *IEEE / CVF Computer Vision and Pattern Recognition Conference 2022*, 2022, pp. 1–19.

[19] K. You, *et al.*, "RENO: Real-time Neural Compression for 3D LiDAR Point Clouds," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 22 172–22 181.

[20] L. Zhao, *et al.*, "Real-Time Scene-Aware LiDAR Point Cloud Compression Using Semantic Prior Representation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 8, pp. 5623–5637, 2022.

[21] N. Stathoulopoulos, *et al.*, "RecNet: An Invertible Point Cloud Encoding through Range Image Embeddings for Multi-Robot Map Sharing and Reconstruction," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[22] Y. Wang, *et al.*, "Dynamic Graph CNN for Learning on Point Clouds," *ACM Trans. Graph.*, vol. 38, no. 5, Oct. 2019.

[23] Q. Hu, *et al.*, "RandLA-Net: Efficient Semantic Segmentation of Large-scale Point Clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 108–11 117.

[24] E. Perez, *et al.*, "FiLM: Visual Reasoning with a General Conditioning Layer," in *AAAI*, 2018.

[25] A. Vaswani, *et al.*, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*. Curran Associates Inc., 2017, p. 6000–6010.

[26] Y. Yang, *et al.*, "FoldingNet: Point Cloud Auto-Encoder via Deep Grid Deformation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 206–215.

[27] J. Behley, *et al.*, "SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[28] I. Vizzo, *et al.*, "KISS-ICP: In Defense of Point-to-Point ICP – Simple, Accurate, and Robust Registration If Done the Right Way," *IEEE Robotics and Automation Letters (RA-L)*, vol. 8, no. 2, pp. 1029–1036, 2023.

[29] A. Milioto, *et al.*, "RangeNet++: Fast and Accurate LiDAR Semantic Segmentation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4213–4220.

[30] K. Koide, *et al.*, "Voxelized GICP for Fast and Accurate 3D Point Cloud Registration," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 11 054–11 059.

[31] F. Dellaert and G. Contributors, "borglab/gtsam," May 2022. [Online]. Available: https://github.com/borglab/gtsam

[32] H. Lim, *et al.*, "KISS-Matcher: Fast and Robust Point Cloud Registration Revisited," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 11 104–11 111.