TESGNN: Temporal Equivariant Scene Graph Neural Networks for Efficient and Robust Multi-View 3D Scene Understanding

Anonymous authors
Paper under double-blind review

Abstract

Scene graphs have proven to be highly effective for various scene understanding tasks due to their compact and explicit representation of relational information. However, current methods often overlook the critical importance of preserving symmetry when generating scene graphs from 3D point clouds, which can lead to reduced accuracy and robustness, particularly when dealing with noisy, multi-view data. Furthermore, a major limitation of prior approaches is the lack of temporal modeling to capture time-dependent relationships among dynamically evolving entities in a scene. To address these challenges, we propose Temporal Equivariant Scene Graph Neural Network (TESGNN), consisting of two key components: (1) an Equivariant Scene Graph Neural Network (ESGNN), which extracts information from 3D point clouds to generate scene graph while preserving crucial symmetry properties, and (2) a Temporal Graph Matching Network, which fuses scene graphs generated by ESGNN across multiple time sequences into a unified global representation using an approximate graph-matching algorithm. Our combined architecture TESGNN outperforms current state-of-the-art methods in scene graph generation, achieving higher accuracy and faster training convergence. Moreover, we show that leveraging the symmetry-preserving property produces a more stable and accurate global scene representation compared to existing approaches. Last but not least, it is computationally efficient and easily implementable using existing frameworks, making it well-suited for real-time applications in robotics and computer vision. This approach paves the way for more robust and scalable solutions to complex multi-view scene understanding challenges.

1 Introduction

Holistic scene understanding plays a critical role in the advancement of robotics and computer vision systems Kim et al. (2024); Li et al. (2022b); Koch et al. (2024). The primary methodologies in this domain include 2D mapping, 3D reconstruction, and scene graph representation. Although 2D mapping remains a straightforward and widely adopted approach Pfaff et al. (2006), it inherently lacks a comprehensive spatial understanding of the environment. In contrast, 3D reconstruction techniques Jin et al. (2024) provide richer spatial information but are resource intensive and particularly susceptible to noise interference Li et al. (2022a). Recent innovations, such as LMNav Shah et al. (2022) and VLMap Huang et al. (2023), leverage Large Language Models (LLMs) to generate semantic maps from 3D data, thereby enhancing scene understanding. However, these approaches often require substantial computational resources. As an alternative, scene graphs, particularly those utilizing Graph Neural Networks (GNNs), offer a more efficient and semantically rich representation compared to traditional 3D reconstruction techniques Chang et al. (2023). Initial methodologies mainly focused on generating scene graphs from sequences of 2D images. In more recent work Wald et al. (2020); Wu et al. (2021), researchers have explored scene graph generation by incorporating 3D features, such as depth data and point clouds. The latest advancements Wu et al. (2023) propose hybrid approaches that integrate both 2D and 3D data to further refine scene graph representations. However, based on our analysis, existing approaches reveal two major limitations.

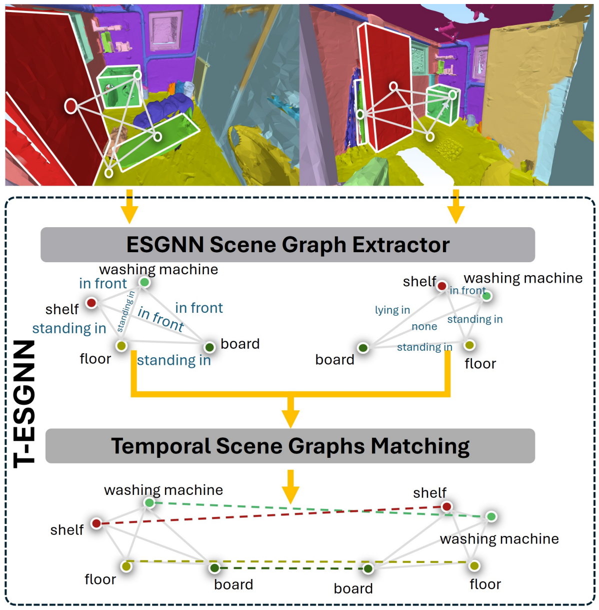


Figure 1: A visualization of multi-view scene graph from multiple 3D point cloud sequences. Our proposed TESGNN first generates local scene graphs for each sequence using Equivariant GNN. Then, the local scene graphs are merged by passing through a temporal layer to form a global scene graph representing the entire scene.

Current methods struggle with noisy, multi-view 3D data, leading to inconsistent scene graphs due to sensitivity to camera angles and poor symmetry preservation. Enhancing Graph Neural Networks (GNNs) with invariant node and edge embeddings and using Equivariant GNNs Köhler et al. (2020); Liao & Smidt (2023) can address this, though prior works Wu et al. (2021; 2023) haven't explored these approaches. Additionally, existing methods lack robust strategies for building coherent global scene graphs from multiple point cloud sequences. For example, Wu et al. (2021) uses heuristic averaging to merge local graphs but skips optimization, resulting in global graphs that lack semantic coherence and structural consistency.

We introduce the **Temporal Equivariant Scene Graph Neural Network (TESGNN)**, a novel architecture that combines two key innovations:

- **Equivariant Scene Graph Neural Network (ESGNN) for Scene Graph Extraction**: extracts scene graphs from 3D point clouds using an Equivariant GNN with Equivariant Graph Convolution and Feature-wise Attention layers, ensuring rotational and translational invariance. This design delivers high-quality scene representations with superior accuracy, fewer training iterations, and reduced computational demands compared to existing methods—marking the first application of symmetry-preserving Equivariant GNNs for this task.
- **Temporal Scene Graph Matching for Global Scene Representation**: fuses the local scene graphs into a unified global representation by solving a graph matching problem based on node embedding similarity, with ESGNN as the backbone scene graph extractor. This embedding-based temporal approach achieves robust multi-view scene understanding, outperforming other extractors and enabling applications like multi-agent coordination and navigation in dynamic environments.

2 Related Work

Graph Representation and Equivariance Graph representation learning encodes structural and relational data into low-dimensional embeddings using methods like GNNs, Graph Transformers Yun et al. (2019); Kim et al. (2022); Cai et al. (2023), and Graph Attention Networks Veličković et al. (2018), enabling tasks such as scene graph understanding without dense 3D data. For 3D applications, standard GNNs lack rotational symmetry; recent SE(3)/E(3)-equivariant architectures like EGNN Satorras et al. (2021) and Equiformer Liao & Smidt (2023) address this by preserving geometric symmetries during transformations, critical for point clouds Uy et al. (2019), molecular structures, and spatial reasoning.

Scence Graph Understanding Research on scene graphs has significantly advanced tasks in vision, natural language processing, and interdisciplinary domains such as mobile robot navigation. Originally introduced as a structure to represent object instances and their relationships within a scene Johnson et al. (2015), scene graphs effectively capture rich semantics in various data modalities, including 2D/3D images Johnson et al. (2015); Armeni et al. (2019), videos Qi et al. (2018), and point clouds Wald et al. (2020). Wald et al. (2020) pioneered 3DSSG, the first method for generating 3D scene graphs from point clouds, accompanied by the annotated 3RScan dataset. Building on this, Wu et al. (2021) introduced Scene Graph Fusion, which employs feature-wise attention to improve scene graph representations, followed by an incremental approach Wu et al. (2023) that integrates both RGB image sequences and sparse 3D point clouds. Scene Graph Fusion remains the state-of-the-art method for scene graph generation from 3D point clouds. Our approach builds upon this by integrating Feature-wise Attention to optimize the Message-Passing process, thereby enhancing the quality of the generated scene graphs.

Temporal Graph Learning Temporal graph learning has gained significant attention for modeling dynamic relationships over time, with applications in traffic prediction Yu et al. (2018); Li et al. (2018); Nguyen et al. (2024). Traditional GNNs assume static graph structures, limiting their applicability in scenarios like video analysis and multi-view scene understanding. To address this, Temporal GNNs Rossi et al. (2020; 2023) incorporate time-series learning to capture evolving patterns. In scene graph matching, methods such as SGAligner Sarkar et al. (2023) and SG-PGM Xie et al. (2024) align nodes across graphs but rely on ground truth graphs, making them impractical for real-world tasks where predicted graphs contain noise, ambiguous edges, and node permutations. To overcome these challenges, we introduce a symmetry-preserving temporal layer in ESGNN, leveraging equivariant properties to merge scene graphs across time steps. Unlike prior approaches that treat temporal information as sequential snapshots, our method integrates graph matching to construct a unified representation that maintains temporal consistency while reducing computational overhead.

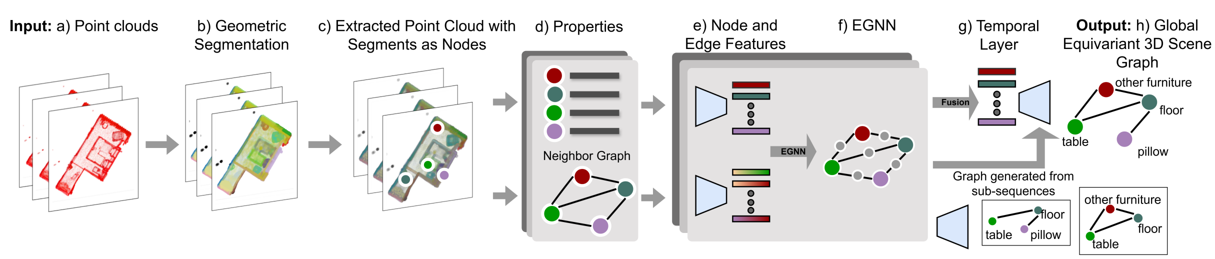


Figure 2: Overview of our proposed TESGNN. Our approach takes sequences of point clouds a) as input to generate a geometric segmentations b). Subsequently, the properties of each segment and a neighbor graph between segments are constructed. The properties d) and neighbor graph e) of the segments that have been updated in the current frame c) are used as the inputs to compute node and edge features f) and to predict a 3D scene graph g). Then it goes through the temporal layer to fuse graphs from different sequences to a global one h).

Large-scale Point Cloud Reconstruction Reconstructing large-scale point clouds is inherently challenging due to their irregular structure, often comprising multiple sub-point cloud sequences Huang et al. (2022). To tackle this issue, we introduce a temporal model that merges graphs generated from these sequences by utilizing embeddings from our proposed models. This approach not only improves integration across sequences but also extends to applications such as multi-agent SLAM and dynamic environment adaptation Jiang et al. (2019); Zou et al. (2019), offering an efficient, lightweight solution based on multi-view scene graph representations.

3 Overall Framework

Fig. 2 illustrates the proposed framework's capability to iteratively estimate a global 3D semantic scene graph from sequence of point clouds. The framework consists of three key phases: Feature Extraction (a-c), discussed in Section 3.2; Scene Graph Extraction (d-f); and Temporal Graph Matching (g-h), detailed in Section 3.3. Our main works, scene graph extraction and temporal graph matching, are further elaborated in Sections 4 and 5.

3.1 Problem Formulation

Our system processes point cloud representations D_i for each scene i in a sequence $(D_1, D_2, \ldots, D_n)$ to generate corresponding scene graphs $(\mathcal{G}_{s,1}, \mathcal{G}_{s,2}, \ldots, \mathcal{G}_{s,n})$ and a global graph $\mathcal{G}_{s,\text{global}}$ that aggregates scene information. Assuming effective feature extraction and geometric segmentation, our focus is on scene graph generation rather than feature extraction. Each point cloud D_i undergoes pre-processing for input to a graph neural network (GNN). A geometric segmentation model partitions D_i into segments, forming nodes of the scene graph, with segment attributes extracted via a point encoder. Detailed feature extraction methods are in Section 3.2.

We model symmetry preservation using the Euclidean group $E(3)$, which encompasses 3D rotations and translations. By employing equivariant layers, our model remains robust to variations in object orientation and position, enhancing generalization and performance while accelerating convergence in scene graph generation.

The Semantic Scene Graph is denoted as $\mathcal{G}_s = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ represent sets of entity nodes and directed edges, respectively. In this case, each node $v_i \in \mathcal{V}$ contains an entity label $l_i \in L$, point clouds $\mathcal{P}_i$, an 3D Oriented Bounding Box (OBB) b_i, and a node category $c_i^{\text{node}} \in \mathcal{C}^{\text{node}}$. Conversely, each edge $e_{i\to j} \in \mathcal{E}$, connecting node v_i to v_j where $i \neq j$, is characterized by an edge category or semantic relationship denoted by $c_{i\to j}^{\text{edge}} \in \mathcal{C}^{\text{edge}}$, or can be written in a relation triplet $\langle$*subject, predicate, object*$\rangle$. Here, L, $\mathcal{C}^{\text{node}}$, and $\mathcal{C}^{\text{edge}}$ represent the sets of all entity labels, node categories, and edge categories, respectively. Given the 3D scene data D_i and D_j that represent the same point cloud of a scene but from different views (rotation and transition), we try to predict the probability distribution of the equivariant scene graph prediction in which the equivariance is preserved:

$$\begin{cases} P(\mathcal{G}|D_i) = P(\mathcal{G}|D_j)_{i\neq j} \\ D_j = R_{i\to j} D_i + T_{i\to j} \end{cases} \tag{1}$$

where $R_{i\to j}$ is the rotation matrix and $T_{i\to j}$ is the transition matrix.

Let us say we have a global point cloud, that is,

$$D_{\text{global}} = \Phi_{\text{matching}}\left(D_1, D_2, ..., D_n\right),$$

where Φ_{matching} is a model / algorithm to align subsequences together, we can define the global scene graph distribution as $P(\mathcal{G}_{s,\text{global}}|D_{\text{global}})$. However, Φ_{matching} is hard to define, especially in the case that the origin coordinate of each sequence is not aligned, which consumes time and resources for sampling or large estimation models. Instead, as the symmetry-perserving property is maintained, the embedding between each graph is similar, it is much easier to perform the matching between the graph embedding vectors, so that:

$$\begin{aligned} P(\mathcal{G}_{s,\text{global}}|D_{\text{global}}) &= P(\mathcal{G}_s|\Phi_{\text{matching}}((D_1, D_2, ..., D_n)) \\ &= \Phi_{\text{graph}}\left(\mathcal{G}_{s,1}, \mathcal{G}_{s,2}, ..., \mathcal{G}_{s,n}\right) \end{aligned}$$

where Φ_{graph} is the model matching the embedding vectors together, which will be described in the later section.

3.2 Feature Extraction

In this phase, the framework extracts the feature for scene graph generation, following the two main steps:

1. **Point Cloud Reconstruction:** The proposed framework takes the point cloud data, which can be reconstructed from various techniques such as ORB-SLAM3 or HybVIO Campos et al. (2021) as the input. However, for the objective validation purpose of scene graph generation, we use the indoor point cloud dataset 3RScan Wald et al. (2019) for ground truth data D_i.
2. **Geometric Segmentation and Point Cloud Extraction with Segment Nodes:** Given a point cloud D_i, this geometric segmentation will provide a segment set $\mathbf{S} = \{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$. Each segment s_i consists of a set of 3D points $\mathbf{P}_i$ where each point is defined as a 3D coordinate $p_i \in \mathbb{R}^3$ and RGB color. Then, the point cloud concerning each entity is fed to the point encoder named PointNet Charles et al. (2017) $f_p(\mathbf{P_i})$ to encode the segments s_i into latent node and edge features, which are then passed to the model detailed in Section 4.

3.3 Scene Graph Extraction and Temporal Graph Matching

In this phase, the framework processes the input from feature extraction (Section 3.2) to generate the scene graph:

- **Properties and Neighbor Graph Extraction:** From the point cloud, we extract features including the centroid $\overline{\mathbf{p}}_i \in \mathbb{R}^3$, standard deviation $\boldsymbol{\sigma}_i \in \mathbb{R}^3$, bounding box size $\mathbf{b}_i \in \mathbb{R}^3$, maximum length $l_i \in \mathbb{R}$, and volume $\nu_i \in \mathbb{R}$. We create edges between nodes only if their bounding boxes are within 0.5 meters of each other, following Wu et al. (2021).
- **Scene Graph Extraction and Temporal Graph Matching:** The processed input from above is then fed to the ESGNN (Section 4) to generate the node and edge embeddings. These embeddings are used to generate the sub-graph for each sequence and merge multiple scene graphs with our Temporal Model (Section 5). For ESGNN, the node classes and edge predicates are predicted using two Multi-Layer Perceptron (MLP) classifiers. ESGNN is trained end-to-end with a joint cross-entropy loss for both objects $\mathcal{L}_{\text{obj}}$ and predicates $\mathcal{L}_{\text{pred}}$, as described in Wald et al. (2020). Meanwhile, our Temporal Graph Matching is trained to minimize the representation distance between identical nodes with contrastive loss Hadsell et al. (2006).

4 Equivariant Scene Graph Neural Network (ESGNN)

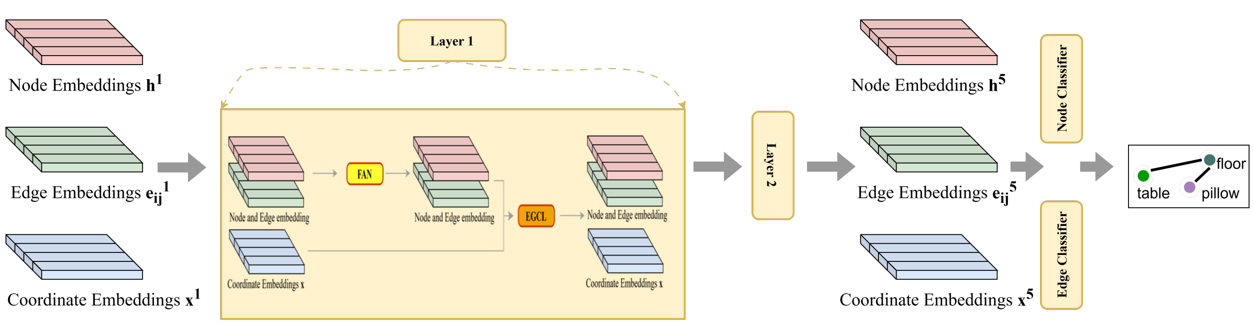


Figure 3: ESGNN Scene Graph Extractor pipeline. The model comprises two main layers: (1) Feature-wise Attention Graph Convolution Layer (FAN-GCL) and (2) Equivariant Graph Convolution Layer (EGCL). FAN-GCL handles large inputs with multi-head attention to update node and edge features, while EGCL ensures symmetry preservation by incorporating bounding box coordinates into the message-passing mechanism. ESGNN leverages these layers to maintain rotation and permutation equivariance, thus enhance the quality of scene graph generation.

To build a network architecture that effectively generate scene graph from point cloud, we propose a combination of Equivariant Graph Convolution Layers Satorras et al. (2021) and the Graph Convolution Layers with Feature-wise Attention Wu et al. (2021).

4.1 Graph Initialization

Node features The node feature includes the invariant features $\mathbf{h_i}$ and the 3-vector coordinate $\mathbf{x_i} \in \mathbb{R}^3$. The invariant features $\mathbf{h}_i$ consists of the latent feature of the point cloud after going through the PointNet $f_p(\mathbf{P_i})$, standard deviation $\boldsymbol{\sigma}_i$, log of the bounding box size $\ln(\mathbf{b}_i)$ where $\mathbf{b}_i = (b_x, b_y, b_z) \in \mathbb{R}^3$, log of the bounding box volume $\ln(v_i)$ where $v_i = b_x b_y b_z$, and log of the maximum length of bounding box $\ln(l_i)$. The coordinate of the bounding box x_i is defined by the coordinates of the two furthest corners of the bound box. The formula can be written as follows:

$$
\begin{aligned}
&\mathbf{v}_i = (\mathbf{h}_i, \mathbf{x}_i),\\
&\mathbf{h}_i = [f_p(\mathbf{P}_i), \boldsymbol{\sigma}_i, \ln(\mathbf{b}_i), \ln(\nu_i), \ln(l_i)],\\
&\mathbf{x}_i = [\mathbf{x}_i^{\text{bottom right}}, \mathbf{x}_i^{\text{top left}}].
\end{aligned}
$$

Edge features The visual characteristics of the edges are determined by the properties of the connected segments. For an edge between a source node i and a target node j where $j \neq i$, the edge visual feature $\mathbf{e}_{ij}$ is computed as follows:

$$
\begin{gathered}
\mathbf{r}_{ij} = \left[\overline{\mathbf{p}}_i - \overline{\mathbf{p}}_j, \boldsymbol{\sigma}_i - \boldsymbol{\sigma}_j, \mathbf{b}_i - \mathbf{b}_j, \ln\left(\frac{l_i}{l_j}\right), \ln\left(\frac{\nu_i}{\nu_j}\right)\right],\\
\mathbf{e}_{ij} = g_s(\mathbf{r}_{ij}),
\end{gathered}
$$

where $g_s(\cdot)$ is a Multi-Layer Perceptron (MLP) that projects the paired segment properties into a latent space.

4.2 Graph Neural Network Architecture and Weights Updating

Fig. 3 describes our GNN network with two core components: ① Feature-wise Attention Graph Convolution Layer (FAN-GCL); and ② Equivariant Graph Convolution Layer (EGCL). The FAN-GCL, proposed by Wu et al. (2021), is used to handle the large input queries Q of dimensions d_q and targets T of dimensions d_τ by utilizing multi-head attention. On the other hand, EGCL, proposed by Satorras et al. (2021), is used to maintain symmetry-preserving equivariance, allowing us to incorporate the bounding box coordinates x_i as node features and update them through the message-passing mechanism.

ESGNN has 2 main layers, each consists of a FAN-GCL followed by an EGCL, forming a total of 4 message-passing layers. For each main layer, the formulas for updating node and edge features are defined as:

- Update FAN-GCL:

$$
\begin{aligned}
\mathbf{v}_i^{\ell+1} &= g_v\left(\left[\mathbf{v}_i^{\ell}, \max_{j \in \mathcal{N}(i)}\left(\text{FAN}\left(\mathbf{v}_i^{\ell}, \mathbf{e}_{ij}^{\ell}, \mathbf{v}_j^{\ell}\right)\right)\right]\right),\\
\mathbf{e}_{ij}^{\ell+1} &= g_e\left(\left[\mathbf{v}_i^{\ell}, \mathbf{e}_{ij}^{\ell}, \mathbf{v}_j^{\ell}\right]\right),
\end{aligned}
$$

- Update EGCL:

$$
\begin{aligned}
h_i^{(l+1)} &= h_i^{(l)} + \mathrm{g_v}\left(\text{concat}\left(h_i^{(l)}, \sum_{j \in \mathcal{N}(i)} e_{ij}^{(l)}\right)\right),\\
e_{ij}^{(l+1)} &= \mathrm{g_e}\left(\text{concat}\left(h_i^{(l)}, h_j^{(l)}, \|\mathbf{x}_i^{(l)} - \mathbf{x}_j^{(l)}\|^2, e_{ij}^{(l)}\right)\right),\\
\mathbf{x}_i^{(l+1)} &= \mathbf{x}_i^{(l)} + \sum_{j \in \mathcal{N}(i)} (\mathbf{x}_i^{(l)} - \mathbf{x}_j^{(l)}) \cdot \phi_{\text{coord}}(e_{ij}^{(l)}).
\end{aligned}
$$

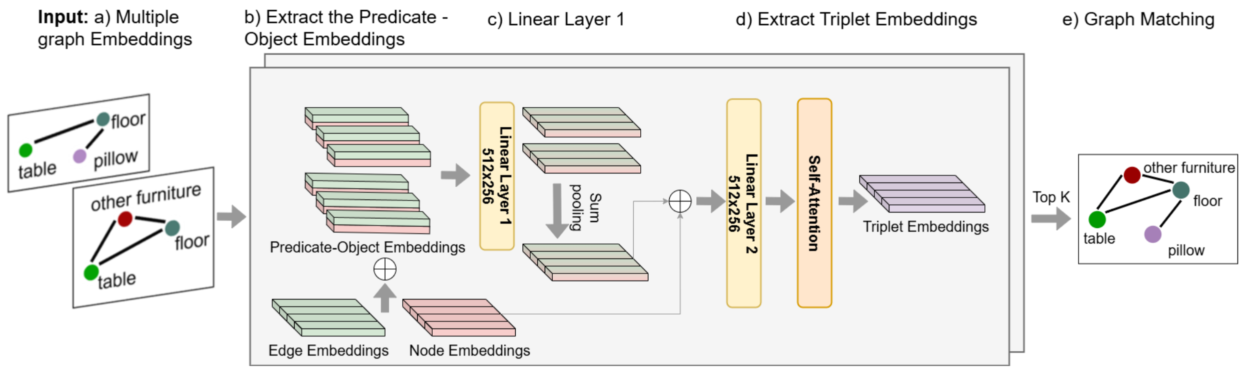


Figure 4: Temporal Graph Matching pipeline. First, node and edge embeddings derived from scene graphs of different sequences (a). For each sequence, edge embeddings are concatenated with the target node embeddings to create Predicate-Object embeddings (b), which then pass through a linear layer followed by sum pooling (c). For each node, the embeddings from all associated edges are concatenated and processed through another linear layer and self-attention mechanism to generate the final representation of each segment (d). These final Triplet Embeddings are utilized for top-K retrieval graph matching (e).

5 Temporal Graph Matching Network

Leveraging the symmetry-preserving properties of our scene graph extractor, we hypothesize that its node and edge embeddings can remain inherently distinguishable, especially for *isomorphic* scene graphs subjected to rotations and translations. This unique feature opens the gate for efficient temporal scene graph matching: Given multiple point cloud sequences containing overlapping regions, our model can reliably match the corresponding parts, regardless of viewpoint changes. As a result, multiple scene graphs can be aligned and fused into a unified, spatially consistent global graph over time.

Figure 4 illustrates our graph matching model. The core idea is to generate a compact representation for each unique triplet ⟨*subject, predicate, object*⟩, then merge identical ones by their similarity. This approach avoids explicit graph isomorphism search, while designed to ensure ensure order-invariant and scale to graphs of any size.

Triplet Representation For each triplet ⟨*subject, predicate, object*⟩ we:

- Concatenate the predicate (edge) embedding with the Object-node embedding to produce a Predicate–Object vector.
- Pass the Predicate–Object vectors connected to the same Subject through a linear layer and sum-pooled to ensure permutation invariance.
- Fuse the pooled vector with the Subject-node embedding, then refine it through a second linear layer and a self-attention block to capture higher-order context.

The result is a compact, rotation-invariant Triplet embedding for every local scene graph.

Similarity-based Graph Matching Given two sequences, we compute cosine similarities between all Triplet embeddings and perform top-K retrieval. Pairs whose similarity exceeds a fixed threshold are treated as the same physical object; the corresponding local graphs are merged node by node to build the global scene graph.

Training Objective We train the graph matching model such that distances between similar Triplet embeddings are minimized, while those between dissimilar embeddings are maximized. We achieve this using Contrastive Loss Hadsell et al. (2006), a well-established method for training embeddings Reimers & Gurevych (2019a). Unlike traditional loss functions that aggregate over samples, Contrastive Loss operates on pairs:

$$\mathcal{L}(W, Y, X_1, X_2) = \frac{1-Y}{2}(D_W)^2 + \frac{Y}{2}\max(0, m - D_W)^2,$$

where (Y, X_1, X_2) are pairs of Triplet Embeddings, D_W is a distance function, and $m > 0$ is a margin. We use Siamese Cosine Similarity Reimers & Gurevych (2019a) for D_W with hard-positive / hard-negative mining to focus learning on the most ambiguous cases.

6 Experiment

We evaluate TESGNN on the 3DSSG dataset Wald et al. (2020) and compare the results with state-of-the-art works. Section 6.3 provides result for scene graph generation from 3D point clouds, in comparison to 3DSSG Wald et al. (2020) and Scene Graph Fusion (SGFN) Wu et al. (2021). Our method is evaluated on full scenes given geometric segments mentioned in Section 3.2. Section 6.5 reports our Temporal Graph Matching.

6.1 Dataset

Scene Graph Extraction: We use the 3DSSG Wald et al. (2020) [1] - a popular dataset for 3D scene graph evaluation built upon 3RScanWald et al. (2019), adapting the setting from SGFN Wu et al. (2021). The original 3RScan contains 1482 3D reconstructions/snapshots of 478 naturally changing indoor environments. After processed with ScanNet Dai et al. (2017) for geometric segmentation and ground truth scene graph generation, the final dataset consists of 1061 sequences from 382 scenes for training, 157 sequences from 47 scenes for validation, and 117 sequences from 102 scenes for test.

Temporal Graph Matching: While reproducing prior works including SGAligner Sarkar et al. (2023) and SG-PGM Xie et al. (2024), we identified limitations in their dataset. Since their work does not contain scene graph extraction, they synthesize scene graph node and edge features using one-hot encoding of the ground truth labels, which does not accurately reflect scene graph embeddings. This set-up impractically ignores real-world noises affecting local scene graphs. To address these shortcomings, we leverage our above set-up for scene graph extraction to evaluate scene graph matching, using the same sequences and scenes. In this context, a positive pair consists of the same object appearing in two or more different sequences. There are 16,156 positive pairs over 321,292 total pairs for train set, and 3,062 positive pairs over 63,890 total pairs for test set. In our set-up, node and edge features are derived from frozen scene graphs such as ESGNN (ours) or SGFN Wu et al. (2021).

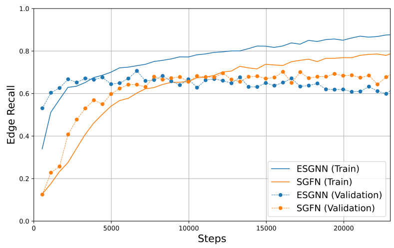
6.2 Metrics

Scene Graph Extraction: We mainly use the **Recall** of node and edge as our evaluation metrics, given the dataset is unbalanced Wald et al. (2020) and the objective of scene graphs is to effectively capture the semantic meaning of the surrounding world. In train phase, we calculate the **Recall** as the true positive overall positive prediction. In test phase, for more detailed analysis, we use **Recall@k (R@k)** Wald et al. (2020); Wu et al. (2021; 2023), which takes **k** most confident predictions and marks it as correct if at least one of these predictions is correct. For a relationship triplet, **R@1** is the accuracy of the predictions. We apply the recall metrics for the predicate (edge classification), object (node classification), and relationship (triplet ⟨*subject, predicate, object*⟩).

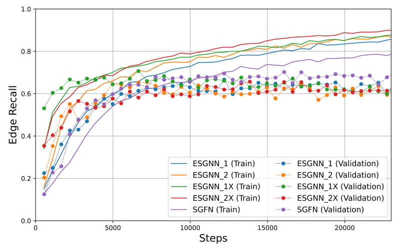
Temporal Graph Matching: We follow the Information Retrieval evaluation metrics, including **R@k**, **MRR@k**, and **mAP@k** [2]. The **R@k** used in this evaluation is slightly different: It is the proportion of instances where at least one correct match appears among the top$k$ matches with a **confidence score** above 0.5. If the matching score is below 0.5, the node is considered unique, indicating that it does not have a common match across sequences. **MRR@k** evaluates the effectiveness of retrieval by averaging the

[1] https://github.com/ShunChengWu/3DSSG. The scene graph ground truth in 3DSSG contains 2 versions: **l20**, which includes 20 objects and 8 predicates, and **l160**, which includes 160 objects and 26 predicates. We mainly use the test set of the **l20** version for our experiment as the **l160** is currently having some issues within the dataset, as confirmed by the authors.

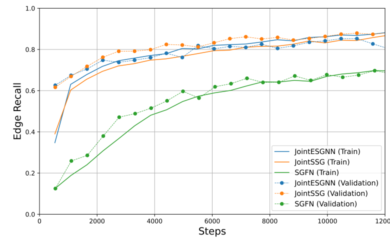
[2] Since each node has only one ground-truth match, mAP@k and MRR@k produce similar result in our setting.



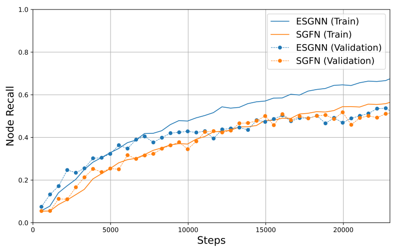
(a) Edge recall on evaluation set 3DSSG-**l20**



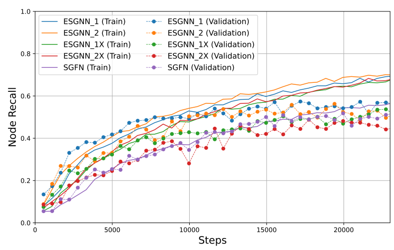
(b) Edge recall on evaluation set 3DSSG-**l20**



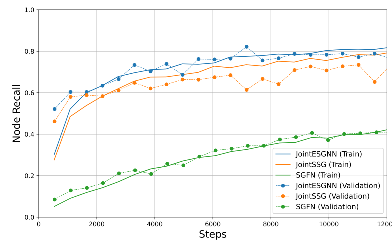
(c) Edge recall on evaluation set 3DSSG-**l20**



(d) Node recall on evaluation set 3DSSG-**l20**



(e) Node recall on evaluation set 3DSSG-**l20**



(f) Node recall on evaluation set 3DSSG-**l20**

Figure 5: Comparison of recall result over training steps, column-wise interpretation. (a), (d) illustrates the result of our model versus SGFN. (b), (e) shows the ablation study. (c), (f) shows result while applying our model with the image encoder.

reciprocal ranks of the k similar nodes. **mAP@k** measures the quality of rankings by emphasizing k relevant nodes at higher ranks.

6.3 Scene Graph Extraction Evaluation

Better prediction outcome: Table 1 compares the results between ESGNN (ours) with existing models 3DSSG Wald et al. (2020) and SGFN Wu et al. (2021) on the 3DSSG-l20 dataset with geometric segmentation setting. Our method obtains high results in both relationship, object, and predicate classification. Especially, ESGNN outperforms the existing methods in relationship prediction and obtains significantly higher **R@1** in predicate compared to SGFN. For unseen data, ESGNN is competitive with SGFN, as shown in Table 2.

Method	Relationship		Object		Predicate		Recall	
	R@1	R@3	R@1	R@3	R@1	R@2	Obj.	Rel.
3DSSG	32.65	50.56	55.74	83.89	**95.22**	98.29	55.74	**95.22**
SGFN	37.82	48.74	62.82	**88.08**	81.41	98.22	63.98	94.24
Ours	**43.54**	**53.64**	**63.94**	<u>86.65</u>	<u>94.62</u>	**98.30**	**65.45**	<u>94.62</u>

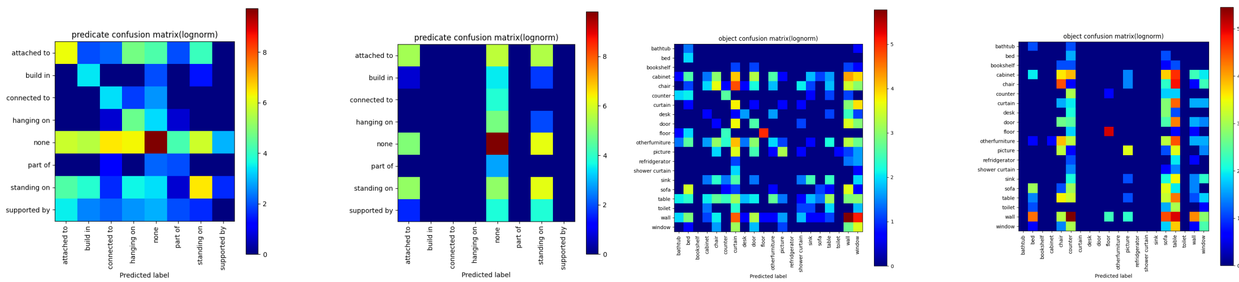
Table 1: Scene graph prediction performance on relationship triplets, objects, and predicates, measured on 3DSSG-l20. The *Recall* columns report object and relationship recall scores (Obj., Rel.).

Faster training convergence: Figure 5a and 5d report the recalls for nodes (objects) and edges (relationships) during training of ESGNN and SGFN on both train and validation sets. The recall slope of ESGNN in the first 10 epochs (5000 steps) is significantly higher than that of SGFN. This shows that ESGNN has faster convergence and higher initial recall.

More robust against imbalanced data: We additionally provide the training confusion matrices after the first 1000 steps in Figure 6. It is notably observable that ESGNN 6a, 6c shows clearer diagonal patterns compared to SGFN 6b, 6d, which was biased towards some dominant classes. These patterns demonstrate that ESGNN is more data-efficient than SGFN, as it does not need to generalize over rotations and transla-

Method	New Relationship		New Object		New Predicate	
	R@1	R@3	R@1	R@3	R@1	R@2
3DSSG	39.74	49.79	55.89	84.42	**70.87**	83.29
SGFN	**47.01**	55.30	64.50	**88.92**	68.71	**83.76**
Ours	<u>46.85</u>	**56.95**	**65.47**	<u>87.52</u>	66.90	82.88

Table 2: Scene graph prediction performance on previously unseen relationship triplets, objects, and predicates. Evaluated on 3DSSG-l20 with geometric segmentation.



(a) ESGNN edge (b) SGFN edge (c) ESGNN node (d) SGFN node

Figure 6: Confusion matrices after the first 1000 steps (2 epochs). ESGNN is better at handling imbalanced classes, as the predictions spread across multiple classes rather than "blindly" predicting towards dominant classes like SGFN.

tions of the data, while still harnessing the flexibility of GNNs in larger datasets. However, due to its high bias Satorras et al. (2021), our EGCL-based message passing performs well with limited data but struggles to learn the dataset's subtleties as the training size increases. In later training steps, we observe the overfitting. Yet, our model consistently outperforms the pioneering works overall.

Scene Graph Extractor with Image Encoder: ESGNN are also potential to apply with point-image encoder instead of 3D point clouds only. Figures 5c and 5f show the performance of our implementation with image encoders, called Joint-ESGNN, compared to existing methods including JointSSG Wu et al. (2023) and SGFN.

6.4 Ablation Study

Table 3 evaluates of ESGNN with different architectures and settings that we experimented. ⓪ is the SGFN, run as the baseline model for comparison. ① is the ESGNN with 1 FAN-GCL layer and 1 EGCL layer. This is our best performer and is used for experiments in Section 6.3. ② is similar to ①. The only difference is that we concatenate coordinate embedding to the output edge embedding after message passing. ③ and ④ are similar to ① and ② respectively, but with 2 FAN-GCL layers and 2 EGCL layers.

Method	Relationship		Object		Predicate	
	R@1	R@3	R@1	R@3	R@1	R@2
⓪ SGFN	37.82	48.74	62.82	**88.08**	81.41	<u>98.22</u>
① **ESGNN_1**	**42.30**	**53.30**	**63.21**	<u>86.70</u>	<u>94.34</u>	**98.30**
② ESGNN_1X	34.96	42.59	57.55	86.18	92.68	98.08
③ ESGNN_2	35.63	44.63	57.55	84.41	93.93	97.94
④ ESGNN_2X	<u>37.94</u>	<u>50.58</u>	59.97	85.23	**94.53**	98.01

Table 3: Evaluation of different ESGNN architectures on the scene graph generation task using the 3DSSG-l20 dataset. ② is our best performer and is used for evaluation in Section 6.3.

Models ③ and ④ perform well on the train set but poorly on the validation and test sets, potentially suffering overfitting as they contain more layers. Models ② and ④ have higher edge recalls in several initial epochs, but decline in the later epochs, as shown in Fig 5b, 5e.

6.5 Temporal Graph Matching Evaluation

Experiment Setup: We assess our work from two key perspectives. First, we evaluate our Graph Matching Network using different backbone scene graph extractors, including the state-of-the-art SGFN Wu et al. (2021) and our proposed ESGNN. Second, we compare the effectiveness of the SGFN and ESGNN backbones on the state-of-the-art SG-PGM Xie et al. (2024). To ensure the same set-up, we reimplement and evaluate SG-PGM with our proposed dataset mentioned in Section 6.1.

Method	Backbone	R@k				MRR@k	
		@1	@2	@3	@5	@2	@3
SG-PGM	SGFN	10.61	19.16	28.56	44.71	14.88	18.02
	ESGNN	**30.14**	**46.94**	**57.63**	**73.86**	**38.54**	**42.10**
SG-PGM (point)	SGFN	18.54	33.28	**45.73**	**62.82**	25.91	30.06
	ESGNN	**20.91**	**33.55**	44.41	62.01	**27.23**	**30.85**
Ours	SGFN	54.48	72.95	82.11	90.45	63.71	66.77
	ESGNN	**70.15**	**84.06**	**90.09**	**95.25**	**78.93**	**81.94**

Table 4: Evaluation of the Graph Matching Network with 30 training epochs. Note that **R@1 = MRR@1** reflects node-matching accuracy.

Better outcome with ESGNN backbone scene graph extractor: Table 4 demonstrates that utilizing our proposed ESGNN as the backbone scene graph extractor consistently outperforms SGFN across all metrics, on both graph matching networks. This underscores the advantage of ESGNN's symmetry-preserving property, which enhances the representation of nodes and edges in 3D point clouds, making them more resilient to transformations and significantly improving graph matching performance.

Better outcome with our Graph Matching Network: Although SG-PGM Xie et al. (2024) achieved competitive results for node matching in their setting, it performs poorly in our scenario, where node and edge features are extracted from a scene graph extractor model rather than one-hot encoded from the ground truth. In our case, both nodes and edges have more labels, leading to several ambiguous predictions, including overlaps, incorrect assignments, and permutations. This explains why SG-PGM fails in this setting. In contrast, our model leverages the equivariant properties of nodes and edges while applying sum pooling, ensuring permutation invariance and significantly outperforming SG-PGM in this scenario, indicating the robustness of the model.

7 Conclusion

In this paper, we introduced the TESGNN, a novel method that overcomes key limitations in 3D scene understanding. By leveraging the symmetry-preserving property of the Equivariant GNN, our architecture ensures robust and efficient generation of semantic scene graphs from 3D point clouds. Our proposed Temporal Graph Matching Model provides a global representation from local scene graphs for real-time dynamic environments. Experimental results show that TESGNN outperforms state-of-the-art methods in accuracy, convergence speed, and computational efficiency.

Limitations & Future Work Future work will focus on optimizing TESGNN for complex real-world scenarios by integrating additional sensor modalities like LiDAR and RGB-D data to improve scene graph generation. We also aim to extend the model's temporal capabilities for continuous data streams, enhancing its suitability for autonomous navigation and multi-agent systems.

References

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. *CoRR*, abs/1910.02527, 2019.

Chen Cai, Truong Son Hy, Rose Yu, and Yusu Wang. On the connection between MPNN and graph transformer. In *Proceedings of the 40th International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2023.

Carlos Campos, Richard Elvira, Juan J. Gomez, José M. M. Montiel, and Juan D. Tardós. ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021.

Xiaojun Chang, Pengzhen Ren, Pengfei Xu, Zhihui Li, Xiaojiang Chen, and Alex Hauptmann. A comprehensive survey of scene graphs: Generation and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):1–26, 2023. doi: 10.1109/TPAMI.2021.3137605.

R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR 2017*, pp. 77–85, 2017. doi: 10.1109/CVPR.2017.16.

Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Niessner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR 2017*, July 2017.

R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR 2006*, 2006.

Chenguang Huang, Oier Mees, Andy Zeng, and Wolfram Burgard. Visual language maps for robot navigation. In *ICRA*, 2023.

Jin Huang, Jantien Stoter, Ravi Peters, and Liangliang Nan. City3d: Large-scale building reconstruction from airborne lidar point clouds. *Remote Sensing*, 14(9), 2022. ISSN 2072-4292. doi: 10.3390/rs14092254.

Truong Son Hy, Viet Bach Nguyen, Long Tran-Thanh, and Risi Kondor. Temporal multiresolution graph neural networks for epidemic prediction. In Peng Xu, Tingting Zhu, Pengkai Zhu, David A. Clifton, Danielle Belgrave, and Yuanting Zhang (eds.), *Proceedings of the 1st Workshop on Healthcare AI and COVID-19, ICML 2022*, volume 184 of *Proceedings of Machine Learning Research*, pp. 21–32. PMLR, 22 Jul 2022.

Li Jiang, Hengshuang Zhao, Shu Liu, Xiaoyong Shen, Chi-Wing Fu, and Jiaya Jia. Hierarchical point-edge interaction network for point cloud semantic segmentation. In *CVPR 2019*, 2019.

Linyi Jin, Nilesh Kulkarni, and David Fouhey. 3dfires: Few image 3d reconstruction for scenes with hidden surface, 2024.

Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David Shamma, Michael Bernstein, and Li Fei-Fei. Image retrieval using scene graphs. In *CVPR*, 2015.

Jinwoo Kim, Dat Tien Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. Pure transformers are powerful graph learners. In *Advances in Neural Information Processing Systems*, 2022.

Kibum Kim, Kanghoon Yoon, Yeonjun In, Jinyoung Moon, Donghyun Kim, and Chanyoung Park. Adaptive self-training framework for fine-grained scene graph generation. In *ICLR*, 2024.

Sebastian Koch, Pedro Hermosilla, Narunas Vaskevicius, Mirco Colosi, and Timo Ropinski. Sgrec3d: Self-supervised 3d scene graph learning via object-level scene reconstruction. In *WACV*, 2024. doi: 10.1109/WACV57701.2024.00337.

Jonas Köhler, Leon Klein, and Frank Noé. Equivariant flows: exact likelihood generative learning for symmetric densities. *ArXiv*, abs/2006.02425, 2020.

Jianwei Li, Wei Gao, Yihong Wu, Yangdong Liu, and Yanfei Shen. High-quality indoor scene 3d reconstruction with rgb-d cameras: A brief review. *Computational Visual Media*, 2022a. ISSN 2096-0662. doi: 10.1007/s41095-021-0250-8.

Rongjie Li, Songyang Zhang, and Xuming He. Sgtr: End-to-end scene graph generation with transformer. In *CVPR*, 2022b. doi: 10.1109/CVPR52688.2022.01888.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018.

Yi-Lun Liao and Tess Smidt. Equiformer: Equivariant graph attention transformer for 3d atomistic graphs. In *ICLR 2023*, 2023.

Bach Nguyen, Truong Son Hy, Long Tran-Thanh, and Nhung Nghiem. Predicting COVID-19 pandemic by spatio-temporal graph neural networks: A new zealand's study. In *Temporal Graph Learning Workshop @ NeurIPS 2023*, 2023.

Duc Thien Nguyen, Manh Duc Tuan Nguyen, Truong Son Hy, and Risi Kondor. Fast temporal wavelet graph neural networks. In Sophia Sanborn, Christian Shewmake, Simone Azeglio, and Nina Miolane (eds.), *Proceedings of the 2nd NeurIPS Workshop on Symmetry and Geometry in Neural Representations*, volume 228 of *Proceedings of Machine Learning Research*, pp. 35–54. PMLR, 16 Dec 2024.

Patrick Pfaff, Wolfram Burgard, and Dieter Fox. Robust monte-carlo localization using adaptive likelihood models. In *European Robotics Symposium*, 2006.

Hang Qi, Yuanlu Xu, Tao Yuan, Tianfu Wu, and Song-Chun Zhu. Scene-centric joint parsing of cross-view videos. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018. doi: 10.1609/aaai.v32i1.12256.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, abs/1908.10084, 2019a.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP 2019*. Association for Computational Linguistics, 11 2019b.

Nils Reimers and Iryna Gurevych. Making monolingual sentence embeddings multilingual using knowledge distillation, 2020.

Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020.

Ryan Rossi, Nesreen Ahmed, and Namyong Park. On graph time-series representations for temporal networks. In *Companion Proceedings of the ACM Web Conference 2023*, WWW '23 Companion, pp. 14–18, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394192. doi: 10.1145/3543873.3587301.

Sayan Deb Sarkar, Ondrej Miksik, Marc Pollefeys, Daniel Barath, and Iro Armeni. Sgaligner: 3d scene alignment with scene graphs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 21927–21937, October 2023.

Víctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks, 18–24 Jul 2021.

Dhruv Shah, Blazej Osinski, Brian Ichter, and Sergey Levine. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action, 2022.

Khanh-Tung Tran, Truong Son Hy, Lili Jiang, and Xuan-Son Vu. Mglep: Multimodal graph learning for modeling emerging pandemics with big data. *Scientific Reports*, 14, 2023.

Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *ICCV*, 2019. doi: 10.1109/ICCV.2019.00167.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *ICLR*, 2018.

Johanna Wald, Armen Avetisyan, Nassir Navab, Federico Tombari, and Matthias Niessner. Rio: 3d object instance re-localization in changing indoor environments. In *ICCV 2019*, pp. 7657–7666, 2019. doi: 10.1109/ICCV.2019.00775.

Johanna Wald, Helisa Dhamo, Nassir Navab, and Federico Tombari. Learning 3D Semantic Scene Graphs from 3D Indoor Reconstructions. In *CVPR*, 2020.

Shun-Cheng Wu, Johanna Wald, Keisuke Tateno, Nassir Navab, and Federico Tombari. Scenegraphfusion: Incremental 3d scene graph prediction from rgb-d sequences. In *CVPR*, 2021.

Shun-Cheng Wu, Keisuke Tateno, Nassir Navab, and Federico Tombari. Incremental 3d semantic scene graph prediction from rgb sequences. In *CVPR*, 2023.

Yaxu Xie, Alain Pagani, and Didier Stricker. Sg-pgm: Partial graph matching network with semantic geometric fusion for 3d scene graph alignment and its downstream tasks. *arXiv preprint arXiv:2403.19474*, 2024.

Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, pp. 3634–3640. AAAI Press, 2018. ISBN 9780999241127.

Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph transformer networks. In *Advances in Neural Information Processing Systems*, 2019.

Danping Zou, Ping Tan, and Wenxian Yu. Collaborative visual slam for multiple agents: A brief survey. *Virtual Reality & Intelligent Hardware*, 1(5):461–482, 2019.

A Anonymous GitHub

We provide the Annonymous GitHub Repository [3] for our TESGNN implementation. All information about us is anonymized and we can also not see the reviewers' information as well.

B Supplementary Related Work

Temporal Graph Learning. Temporal graph learning has gained significant attention in recent years, driven by the need to model dynamically evolving relationships over time with various applications in traffic prediction Yu et al. (2018); Li et al. (2018); Nguyen et al. (2024) and pandemic modeling Hy et al. (2022); Tran et al. (2023); Nguyen et al. (2023). Traditional Graph Neural Networks (GNNs) often assume static graph structures, which limits their applicability in scenarios where graph nodes and edges change continuously, such as in video analysis and multi-view scene understanding. To address these limitations, Temporal GNNs including Rossi et al. (2020; 2023) incorporate temporal dependencies and time-series learning to capture time-evolving patterns and enhance predictive performance on dynamic graphs. Our work builds upon these concepts by introducing a temporal layer that preserves symmetry, leveraging the equivariant properties of our Equivariant Scene Graph Neural Network (ESGNN). Unlike prior approaches that treat temporal information as sequential snapshots, we propose a graph matching mechanism to merge scene graphs from multiple time steps into a unified global representation, which preserves temporal consistency while reducing computational overhead.

C Equivariance of ESGNN

This section demonstrates that our model is translation equivariant with respect to $\mathbf{x}$ for any translation vector $g \in \mathbb{R}^n$, and rotation and reflection equivariant with respect to $\mathbf{x}$ for any orthogonal matrix $Q \in \mathbb{R}^{n \times n}$. Formally, we prove that the model satisfies:

$$Q\mathbf{x}^{l+1} + g, \mathbf{h}^{l+1} = \mathrm{ESGNN}\left(Q\mathbf{x}^l + g, \mathbf{h}^l\right)$$

C.1 FAN-GCL Layer

Node Update:

$$\mathbf{h}_i^{\ell+1} = g_v\left(\left[\mathbf{h}_i^\ell, \max_{j \in \mathcal{N}(i)}\left(\mathrm{FAN}\left(\mathbf{h}_i^\ell, \mathbf{e}_{ij}^\ell, \mathbf{h}_j^\ell\right)\right)\right]\right),$$

Edge Update:

$$\mathbf{e}_{ij}^{\ell+1} = g_e\left(\left[\mathbf{h}_i^\ell, \mathbf{e}_{ij}^\ell, \mathbf{h}_j^\ell\right]\right),$$

Equivariance: Assuming $\mathbf{h}^\ell$ is invariant to $\mathrm{E}(n)$ transformations on $\mathbf{x}$—i.e., no information about the absolute position or orientation of $\mathbf{x}^\ell$ is encoded into $\mathbf{h}^\ell$—then the outputs $\mathbf{h}_i^{\ell+1}$ and $\mathbf{e}_{ij}^{\ell+1}$ of FAN-GCL are also invariant.

C.2 EGCL Layer

Node Update:

$$h_i^{(l+1)} = h_i^{(l)} + g_v\left(\mathrm{concat}\left(h_i^{(l)}, \sum_{j \in \mathcal{N}(i)} e_{ij}^{(l)}\right)\right),$$

Edge Update:

$$e_{ij}^{(l+1)} = g_e\left(\mathrm{concat}\left(h_i^{(l)}, h_j^{(l)}, \|\mathbf{x}_i^{(l)} - \mathbf{x}_j^{(l)}\|^2, e_{ij}^{(l)}\right)\right),$$

[3] Link: https://anonymous.4open.science/r/TESGNN-4BB3/

Coordinate Update:

$$\mathbf{x}_i^{(l+1)} = \mathbf{x}_i^{(l)} + \sum_{j \in \mathcal{N}(i)} (\mathbf{x}_i^{(l)} - \mathbf{x}_j^{(l)}) \cdot \phi_{\text{coord}}(e_{ij}^{(l)}),$$

Equivariance: Since $\mathbf{h}^{\ell}$ is invariant to $\mathrm{E}(n)$ transformations of $\mathbf{x}$, the output $e_{ij}^{(l+1)}$ of EGCL is also invariant because the distance between two particles is invariant to translations:

$$\left\|\mathbf{x}_i^l + g - \left[\mathbf{x}_j^l + g\right]\right\|^2 = \left\|\mathbf{x}_i^l - \mathbf{x}_j^l\right\|^2,$$

and invariant to rotations and reflections:

$$\begin{aligned}
\left\|Q\mathbf{x}_i^l - Q\mathbf{x}_j^l\right\|^2 &= \left(\mathbf{x}_i^l - \mathbf{x}_j^l\right)^\top Q^\top Q \left(\mathbf{x}_i^l - \mathbf{x}_j^l\right) \\
&= \left(\mathbf{x}_i^l - \mathbf{x}_j^l\right)^\top \mathbf{I} \left(\mathbf{x}_i^l - \mathbf{x}_j^l\right) \\
&= \left\|\mathbf{x}_i^l - \mathbf{x}_j^l\right\|^2
\end{aligned}$$

Thus, the edge operation becomes invariant:

$$\begin{aligned}
e_{ij}^{(l+1)} &= g_e\left(\mathbf{h}_i^l, \mathbf{h}_j^l, \left\|Q\mathbf{x}_i^l + g - \left[Q\mathbf{x}_j^l + g\right]\right\|^2, e_{ij}^{(l)}\right) \\
&= g_e\left(\mathbf{h}_i^l, \mathbf{h}_j^l, \left\|\mathbf{x}_i^l - \mathbf{x}_j^l\right\|^2, e_{ij}^{(l)}\right)
\end{aligned}$$

The coordinates $\mathbf{x}$ are also $\mathrm{E}(n)$ equivariant. We prove this by showing that an $\mathrm{E}(n)$ transformation of the input yields the same transformation of the output. Since $e_{ij}^{(l+1)}$ is already invariant, we aim to show:

$$\begin{aligned}
&Q\mathbf{x}_i^{l+1} + g \\
&= Q\mathbf{x}_i^l + g + C\sum_{j \neq i}\left(Q\mathbf{x}_i^l + g - \left[Q\mathbf{x}_j^l + g\right]\right) g_{coord}\left(e_{ij}^{(l+1)}\right)
\end{aligned}$$

We have the following derivation:

$$\begin{aligned}
&Q\mathbf{x}_i^l + g + C\sum_{j \neq i}\left(Q\mathbf{x}_i^l + g - Q\mathbf{x}_j^l - g\right) g_{coord}\left(e_{ij}^{(l+1)}\right) \\
&= Q\mathbf{x}_i^l + g + QC\sum_{j \neq i}\left(\mathbf{x}_i^l - \mathbf{x}_j^l\right) g_{coord}\left(e_{ij}^{(l+1)}\right) \\
&= Q\left(\mathbf{x}_i^l + C\sum_{j \neq i}\left(\mathbf{x}_i^l - \mathbf{x}_j^l\right) g_{coord}\left(e_{ij}^{(l+1)}\right)\right) + g \\
&= Q\mathbf{x}_i^{l+1} + g
\end{aligned}$$

Thus, a transformation $Q\mathbf{x}^l + g$ on $\mathbf{x}^l$ leads to the same transformation on $\mathbf{x}^{l+1}$, while $\mathbf{h}^{l+1}$ remains invariant, satisfying:

$$Q\mathbf{x}^{l+1} + g, \mathbf{h}^{l+1} = \mathrm{EGCL}\left(Q\mathbf{x}^l + g, \mathbf{h}^l\right).$$

Both FAN-GCL and EGCL layers are designed to maintain equivariance under translation, rotation, and permutation transformations, making them well-suited for tasks requiring spatial and relational reasoning in graph-based neural networks.

D Towards Temporal Model's Training and Future Works

This section explores various training strategies and loss functions applicable to our Temporal Model approach. Our model demonstrates adaptability to diverse training implementations. Furthermore, beyond achieving good accuracy in graph matching, there is potential for future work in model distillation for light-weight application.

D.1 Loss Selection and Training With Multiple Losses

Our Temporal Model adopts the training objectives of embedding models, which is to optimize the embedding model such that distances between similar embeddings are minimized, while those between dissimilar embeddings are maximized. There are several common training strategies [4] Reimers & Gurevych (2019b) as mentioned below.

Contrastive Loss: This loss expects an embedding pair and a label of either 0 or 1. For label 1, then the distance between the two embeddings is reduced. Otherwise, the distance between the embeddings is increased.

Online Contrastive Loss: This leverages the Contrastive Loss, but computes the loss only for pairs of hard positive (positives that are far apart) or hard negative (negative that are close) per iteration. This loss empirically produces better performances. In our training setting for Temporal Model, we utilize this as our main loss.

Multiple Negatives Ranking Loss: This loss expects a batch consisting of embedding pairs $(a_1, p_1), (a_2, p_2)..., (a_n, p_n)$ where we assume that (a_i, p_i) are a positive pair and $(a_i, p_j), i \neq j$ are negative pairs. It then minimizes the negative log-likelihood for softmax normalized scores. To improve the robustness of the model, we can also provide multiple hard negatives pairs per batch during training.

Cosine Similarity Loss: A fundamental approach. In this loss, the similarity of embeddings is computed with Cosine Similarity and compared (by Mean-Squared Error) to the labeled similarity score. In our case, we can treat label 0 (dissimilar pairs) and label 1 (similar pair) like a float similarity score. However, this loss often takes significantly longer time to compute due to the cosine similarity calculation.

Training strategy with multiple losses: In this strategy, we calculate each loss and update the model weights iteratively per training iteration. An example setup is to calculate the Contrastive Loss follows by the Multiple Negatives Ranking Loss. This strategy often enhances the outcome model and provides better results. In future work, we will improve our Temporal Model by applying this strategy.

Strategy for distillation for light-weight application: In future work, after we achieve high accuracy with the Temporal Model training, one potential move is reducing the size of the Temporal Model while trying to balance the accuracy trade-off, a.k.a knowledge distillation. One strategy is to utilize the MSE loss, in which our target is to minimize the distance between the output embeddings of the teacher model and the student (distilled, light-weighted) model Reimers & Gurevych (2020).

E Implementation Details for ESGNN

E.1 Network Architecture

The setup for ESGNN architecture is provided in the code block below. Our output node and edge dimensions are set to be 256. As discussed in Section 6.3. Ablation Study of the main paper, we experimented different layer setting for the training. Each ESGNN layer is defined as a combination of 1 FAN-GCL layer following by 1 EGCL layer to produce the final node and edge embeddings. For the FAN-GCL, we use 8 Attention heads.

```
model:
    node_feature_dim: 256
    edge_feature_dim: 256
    gnn:
        hidden_dim: 256
```

[4] Sentence Transformer Docs: We mostly refer to this well-scripted documentation for loss selection as well as different training strategies. Although its main application is for text embedding, it is very helpful to our proposed approach.

```
    num_layers: 1
    num_heads: 8
    drop_out: 0.3
```

E.2 Training Details

We adopt similar training strategy to state-of-the-art works 3DSSG Wald et al. (2020) and Scene Graph Fusion Wu et al. (2021). This ensures the results of our method ESGNN is comparable to these existing results. Our params setup is as follows:

```
training:
  max_epoch: 200
  lr: 1e-4
  patient: 30
  optimizer: 'adamw'
  amsgrad: true

  lambda_mode: dynamic # calculate the ratio of the number of node and edge.
  lambda_node: 0.1 # learning rate ratio
  lambda_edge: 1.0 # learning rate ratio

  scheduler:
    method: reduceluronplateau
    args: {gamma: 0.5,
           factor: 0.9}

  model_selection_metric: iou_node_cls # Select and save best model based on the IoU
  model_selection_mode: maximize # can be maximize or minimize. e.g. if it's "loss", shou

  metric_smoothing:
    method: ema

eval:
  topK: 10 # Evaluate with topk = 1, 3, 5, 10, etc.
```

Following Wu et al. (2021), we leverage the AdamW optimizer with Amsgrad and an adaptive learning rate inversely proportional to the logarithm of the number of edges. Given a training batch containing n edges and a base learning rate $\mathrm{lr}_{\mathrm{base}} = 1 \times 10^{-3}$, the learning rate is adjusted as follows:

$$\mathrm{lr} = \mathrm{lr}_{\mathrm{base}} \cdot \frac{1}{\ln n}.$$

We set the maximum training epoch to 200, with learning rate of 0.0001.