Incremental 3D Semantic Scene Graph Prediction from RGB Sequences

Shun-Cheng Wu¹ Keisuke Tateno² Nassir Navab¹ Federico Tombari¹,²
¹Technische Universit¨at M¨unchen ²Google

Abstract

3D semantic scene graphs are a powerful holistic representation as they describe the individual objects and depict the relation between them. They are compact high-level graphs that enable many tasks requiring scene reasoning. In real-world settings, existing 3D estimation methods produce robust predictions that mostly rely on dense inputs. In this work, we propose a real-time framework that incrementally builds a consistent 3D semantic scene graph of a scene given an RGB image sequence. Our method consists of a novel incremental entity estimation pipeline and a scene graph prediction network. The proposed pipeline simultaneously reconstructs a sparse point map and fuses entity estimation from the input images. The proposed network estimates 3D semantic scene graphs with iterative message passing using multi-view and geometric features extracted from the scene entities. Extensive experiments on the 3RScan dataset show the effectiveness of the proposed method in this challenging task, outperforming state-of-the-art approaches. Our implementation is available at https://shunchengwu.github.io/MonoSSG.

1. Introduction

Scene understanding is a cornerstone in many computer vision applications requiring perception, interaction, and manipulation, such as robotics, AR/VR and autonomous systems [17, 54–56]. Semantic Scene Graphs (SSGs) go beyond recognizing individual entities (objects and stuff) by reasoning about the relationships among them [61, 66]. They also proved to be a valuable representation for complex scene understanding tasks, such as image captioning [26, 67], generation [13, 24], scene manipulation [10, 11], task planning [27], and surgical procedure estimation [42, 43]. Given the benefits of such representations, scene graph estimation received increasing attention in the computer vision community.

While earlier methods mainly estimate SSGs from images [18, 19, 33, 66, 72], recent approaches have also investigated estimating them from 3D data. Compared to 2D scene graphs, which describe a single image, 3D scene graphs depict the entire 3D scenes, enabling applications requiring a holistic understanding of the whole scene, such as path planning [47], camera localization, and loop closure detection [23]. However, existing 3D methods either require dense 3D geometry of the scenes to estimate 3D scene graphs [1, 23, 61, 64], which limits the use case since dense geometry is not always available, or constrains the scene graph estimation at the image-level [15, 27, 66], which tend to fail inferring relationships among objects beyond the individual viewpoints. A method that estimates 3D scene graphs relies on sparse scene geometry and reasoning about relationships globally has not been explored yet.

In this work, we propose a real-time framework that incrementally estimates a global 3D SSG of a scene simply requiring an RGB sequence as input. The process is illustrated in Fig. 1. Our method simultaneously reconstructs a segmented point cloud while estimating the SSGs of the current map. The estimations are bound to the point map, which allows us to fuse them into a consistent global scene
The segmented map is constructed by fusing entity estimation from images to the points estimated from a sparse Simultaneous Localization and Mapping (SLAM) method [3]. Our network takes the entities and other properties extracted from the segmented map to estimate 3D scene graphs. Fusing entities across frames is non-trivial. Existing methods often rely on dense inputs [38, 58] and struggle with sparse inputs since the points are not uniformly distributed. Estimating scene graphs with sparse input points is also challenging. Sparse and ambiguous geometry renders the node representations unreliable. On the other hand, directly estimating scene graphs from 2D images ignores the relationship beyond visible viewpoints. We aim to overcome the aforementioned issues by proposing two novel approaches. First, we propose a confidence-based fusion scheme which is robust to variations in the point distribution. Second, we present a scene graph prediction network that mainly relies on multi-view images as the node feature representation. Our approach overcomes the need for exact 3D geometry and is able to estimate relationships without view constraints. In addition, our network is flexible and generalizable as it works not only with sparse inputs but also with dense geometry.

We comprehensively evaluate our method on the 3D SSG estimation task from the public 3RScan dataset [60]. We experiment and compare with three input types, as well as 2D and 3D approaches. Moreover, we provide a detailed ablation study on the proposed network. The results show that our method outperforms all existing approaches by a significant margin. The main contributions of this work can be summarized as follows: (1) We propose the first incremental 3D scene graph prediction method using only RGB images. (2) We introduce an entity label association method that works on sparse point maps. (3) We propose a novel network architecture that generalizes with different input types and outperforms all existing methods.

2. Related Work

2.1. 3D Object Localization from Images

Localizing 3D objects from images aims to predict the position and orientation of objects. Existing methods can be broadly divided into two categories: without and with explicit geometrical reasoning.

In the former category, many works focus on estimating 3D bounding boxes by extending 2D detectors with learned priors [28, 37, 41, 70]. When sequential input is available, single view estimations can be fused to estimate a consistent object map [2, 22, 29, 30]. However, the fused results may not fulfill the multi-view geometric constraints. Multi-view approaches estimate oriented 3D bounding boxes from the given 2D detection of views. They mainly focus on minimizing the discrepancies between the projected 3D representation and the detected 2D bounding boxes.

In the latter category, 3D objects are localized with the help of explicit geometric information. Many existing methods treat object detection as spatial landmarks in a map [21, 35, 40, 53, 65, 69], also known as object-level SLAM. Others focus on fusing dense per-pixel predictions to a reconstructed map [6, 16, 38, 44, 71], which is known as semantic mapping or semantic SLAM.

A major difference between object-level and semantic SLAM is that the former focuses only on foreground objects, while the latter also considers the structural and background information. Specifically, SemanticFusion [36] fuses dense semantic segments from images to a consistent dense 3D map with Bayesian updates. Its map representation provides a dense semantic understanding of a scene ignoring individual instances. PanopticFusion [38] proposes to combine the instance and semantic segmentation from images to a panoptic map. Their approach considers foreground object instances and non-instance semantic information from the background. SceneGraphFusion [64] relies on 3D geometric segmentation [59] and scene graph reasoning to achieve instance understanding of all entities in a scene.

One significant difficulty in instance estimation for semantic SLAM is associating the instances across frames. Existing approaches mainly rely on a dense map to associate predictions by calculating the intersection-over-union (IoU) or the overlapping ratio between the input and the rendered image from the map. However, these methods produce sub-optimal results when the map representation is sparse due to the non-uniform distribution of the map points. We overcome this problem by proposing a confidence-based association.

2.2. 3D Semantic Scene Graph

Estimating 3D scene graphs methods can be divided according to various criteria. From the perspective of the scene graph structure, some methods focus on hierarchical scene graphs [1, 23, 48, 50]. These approaches mainly address the problem of relationship estimation between entities from different hierarchical levels, e.g., object, and room level. The other methods focus on pairwise relationships, e.g., support and comparative relationships, between nodes within a scene [27, 61, 64, 66]. From the perspective of input data, some previous methods rely on RGB input [15, 27] by fusing 2D scene graph predictions to a consistent 3D map. On the other hands, other methods rely on 3D input [1, 23, 48, 50, 61, 64] by using known 3D geometries. Nevertheless, most of the existing methods estimate scene graphs offline [1, 15, 48, 49], while a few works [23, 27, 64] predict scene graph in real-time.

Among all existing work, the pioneering work in 3D scene graph estimation is proposed by [1]. The authors extend the 2D scene graph estimation method from [66] with
temporal consistency across frames and use geometric features from ellipsoids. Kim et al. [27] propose an incremental framework to estimate 3D scene graphs from 2D estimations. Armeni et al. [1] are the first to estimate 3D scene graphs through the hierarchical understanding of the scene. Rosinol et al. [48] build on top of [1] to capture moving agents. This work is subsequently extended to a SLAM system [50]. Wald et al. [61] propose the first 3D scene graph method based on the relationship between objects at the same level, along with 3RScan: a richly annotated 3D scene graph dataset. Wu et al. [64] extend [61] to real-time scene graph estimation with a novel feature aggregation mechanism. Hughes et al. [23] propose to reconstruct a 3D hierarchical scene graph in real-time incrementally. Our method incrementally estimates a flat scene graph with multi-view RGB input and a sparse 3D geometry, which differentiates our work from the previous methods relying on 3D input [61,64], and approaches without geometric understanding [15,27].

3. Method

The proposed framework is illustrated in Fig. 2, which shows how, given a sequence of RGB images, it can estimate a 3D semantic scene graph incrementally. The Incremental Entity Estimation (IEE) front end makes use of the images to generate segmented sparse points. Those are merged into 3D entities and used to generate both an entity visibility graph and a neighbour graph. The Semantic Scene Graph Prediction (SSGP) network uses the entities and both graphs to estimate multiple scene graphs and then fuse them into a consistent 3D SSG.

We define a SSG as $G_s = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ denote a set of entity nodes and directed edges. Each node $v_i \in \mathcal{V}$ is assigned an entity label $l_i \in L$, a set of points $\mathcal{I}_i$, an Oriented Bounding Box (OBB) $b_i$, and a node category $c^\text{node}_i \in C^\text{node}$. Each edge $e_{i,j} \in \mathcal{E}$, connecting node $v_i$ to $v_j$ where $i \neq j$, consists of an edge category $c^\text{edge}_{i,j} \in C^\text{edge}$, $L$, $C^\text{node}$, and $C^\text{edge}$ denote all entity labels, a node category set, and an edge category set, respectively. An OBB $b_i$ is a gravity-aligned 3D bounding box consisting of a boundary dimension $b_i \in \mathbb{R}^3$, a center $\mathbf{c}_i \in \mathbb{R}^3$, and an angle that encodes the rotation along the gravity axis. The OBBs are used to build both graphs and features. The entity visibility graph models the visibility relationship of the entities as a bipartite graph $G_v = (\mathcal{V}, \mathcal{K}, \mathcal{E}_v)$ where $\mathcal{K}$, $\mathcal{E}_v$ denote a set of keyframes and visibility edges, respectively. $G_v$ gives the knowledge of the visibility of entity nodes in keyframes, which is used in computing multi-view visual features in SSG. The neighbour graph encodes the proximity relationship of the entities as an undirected graph $G_p = (\mathcal{V}, \mathcal{E}_p)$, where $\mathcal{E}_p$ is the set of proximity edges. The neighbour graph also serves as the initial graph for the message propagation step in SSG.

3.1. Incremental Entity Estimation

During the first step of the IEE front end pipeline, a set of labeled 3D points are estimated from the sequence of RGB images (Sec. 3.1.1). The entity labels are determined using an entity segmentation method on selected keyframes (Sec. 3.1.2). Then, they are associated and fused into a sparse point map (Sec. 3.1.3). Finally, the entities and their properties are extracted using the labeled 3D points (Sec. 3.1.4).

3.1.1 Sparse Point Mapping

We use ORB-SLAM3 [3] to simultaneously estimate the camera poses and build a sparse point map by matching estimated keypoints from sequential RGB frames. To guarantee real-time performance, an independent thread is used to run the local mapping process using the stored keyframes. The same thread additionally takes care of running the entity detector and performing the label mapping process. For each point $p_m \in \mathbb{P}$ in the map, we store its 3D coordinates, an entity label $l_m$, and its confidence score $w_m \in \mathbb{R}_{\geq 0}$. 

Figure 2. Given a sequence of RGB images, we use every frame to reconstruct a sparse point map (a) and every keyframe to estimate the 2D entities (b). (a) and (b) are associated and merged into a single 3D map (c). We asynchronously extract graph properties from the entity map (d) to estimate SSG. Our network computes the geometric, multi-view, and edge features (e). These features are propagated to each other with message passing (f), then used to predict a SSG (g). Then, periodic SSGs are fused to a global 3D SSG (h).
3.1.2 2D Entity Detection

We estimate an entity label mask \( \hat{M}_t(u) \in \hat{L}_t \) and a confidence mask \( \hat{W}_t^c(u) \in [0, 1] \subset \mathbb{R}^c \) with every given keyframe \( k_t \in \mathcal{K} \), where \( u \in \mathbb{R}^2 \) denotes the image coordinates and \( \hat{L}_t \) all entity labels in \( k_t \). Both masks are estimated using a class-agnostic segmentation network which further improves other instance segmentation methods [4, 7, 31] by enabling the discovery of unseen entities [14, 25, 46]. Although segmentation networks provide accurate masks, the estimations are independent across frames. Thus, a label association stage is required to build a consistent label map.

3.1.3 Label Association and Fusion

Inspired by [35, 38, 51, 58], we use the reference mask approach to handle the label inconsistency. It relies on a map reconstruction to solve label consistency by comparing input label mask to rendered mask. Then fuse the associated mask to the global point map.

Label Association. We start by building a reference entity mask \( M_t(u) \in L \) by projecting point entity labels from the sparse point map using the pose of \( k_t \). The consistency-resolved entity mask \( \hat{M}_t(u) \) is estimated by evaluating the corresponding labels on the image mask \( \hat{M}_t(u) \) and the reference mask \( M_t(u) \). This evaluation can be performed by different methods such as using intersection over union statistics [38] or the maximum overlapping ratio between label masks [58]. However, both methods assume that points are uniformly distributed; a premise that fails in most sparse point reconstruction tasks. In such cases, these methods become unstable, as shown in the example provided in the supplementary material. To overcome this problem, we propose to use the maximum mean confidence as the criteria to find the best candidate. First, a confidence mask \( \hat{W}_t^c(u) \) is built by projecting the pixel label confidence using the pose of \( k_t \), then the mean confidence score of a label \( l \in M_t(u) \) and a reference label \( l \in M_t(u) \) is computed by

\[
\mathcal{M}(\tilde{l}, l) = \frac{\sum_{u' \in \Pi(\tilde{l}, l)} \hat{W}_t^c(u')}{{\#(\Pi(\tilde{l}, l))}},
\]

where \( \Pi(\tilde{l}, l) \) gives a set of image coordinates \( u' \in \mathbb{R}^2 \) where \( l \) overlap: \( \{ u' \mid (\hat{M}_t(u') = \tilde{l}) \land (M_t(u') = l) \} \), and \( {\#(\cdot)} \) is the cardinality operator. Then, the mask \( \hat{M}_t(u) \) is generated by replacing the per-pixel entity label \( \tilde{l} \in \hat{M}_t(u) \) with either a reference label \( l \) or a new label \( l_{\text{new}} \not\in L \) depending on:

\[
\tilde{l} = \begin{cases} 
\arg \max_{l} \mathcal{M}(\tilde{l}, l) \quad \text{if} \max_{\tilde{l}} \frac{{\#(\Pi(\tilde{l}, l))}}{{\#(\hat{M}_t(u) = \tilde{l})}} > \tau, \\
\text{otherwise}
\end{cases}
\]

where we filter out a match if the number of overlapped pixel has a low coverage over the total number of label \( l \) on the reference mask with a threshold \( \tau \). In addition, similar to [38], a reference entity label is assigned to only one input entity label. If a reference label has been assigned, we use descending order to search for the next best candidate.

Label Fusion. After the association process, the associated entity labels \( \hat{M}_t'(u) \) are fused to the sparse point map \( \hat{P} \). Since each label on \( \hat{M}_t'(u) \) sources from a map point, the confidence value of a point is updated by

\[
w_{\psi(u)} = \begin{cases} 
\hat{w}_{\psi(u)} + \hat{W}_t^c(u) & \text{if} \ M_t(u) = \hat{M}_t'(u), \\
\hat{w}_{\psi(u)} - \hat{W}_t^c(u) & \text{otherwise}
\end{cases}
\]

where \( \psi(u) \) is the corresponding point index that is projected on the pixel location \( u \) on both \( M_t(u) \) and \( \hat{M}_t'(u) \). In particular, when \( \hat{w}_{\psi(u)} < 0 \), we set the entity label \( l_{\psi(u)} \) to \( \hat{M}_t'(u) \), and the weight \( \hat{w}_{\psi(u)} \) to \( \hat{W}_t(u) \).

3.1.4 Extraction

We use the points belonging to each entity label to compute the 3D OBB \( b_i \) of an entity \( v_i \in \mathcal{V} \). We perform statistical outlier removal (from PCL [52]) to filter out points that could lead to distorted boxes. For the computation, we make use of the minimum volume estimation method [5] assuming gravity alignment.

The entity visibility graph \( \mathcal{G}_v = (\mathcal{V}, \mathcal{K}, \mathcal{E}_v) \) consists of all nodes \( \mathcal{V} \) and keyframes \( \mathcal{K} \) connected by visibility edges \( \mathcal{E}_v \). A visibility edge \( e_{ij} \in \mathcal{E}_v \) exists if entity \( v_i \in \mathcal{V} \) is visible in keyframe \( k_j \in \mathcal{K} \). Specifically, the visibility is determined by checking if any point in node \( v_i \) is visible at \( k_j \).

The neighbour graph \( \mathcal{G}_p = (\mathcal{V}, \mathcal{E}_p) \) consists of nodes \( \mathcal{V} \) and its proximity edges \( \mathcal{E}_p \). A proximity edge \( e_{ij} \in \mathcal{E}_p \) if \( v_i, v_j \in \mathcal{V}, i \neq j \) exists if nodes \( v_i, v_j \) are close in space, which is determined using a bounding box collision detection method. Since the size of the OBBs is not precise, we extend their dimensions by a margin \( \tau^C \) to include additional potential neighbours.

3.2. Semantic Scene Graph Prediction

For every of the scene extractions obtained by the IEE front end, SSGP estimates 3D semantic scene graphs using message passing to jointly update initial feature representations and relationships [15, 61, 64, 66]. In the last step, the network fuses all of them into a consistent global 3D SSG. The initial node features are computed with multiview image features (Sec. 3.2.1), while the initial edge feature is computed with the relative geometric properties of its connected two nodes (Sec. 3.2.2). Both initial features are jointly updated with a GNN along the connectivity given by the neighbour graph (Sec. 3.2.3). The updated node and edge are used to estimate their class distribution (Sec. 3.2.4). We apply a temporal scene graph fusion procedure to combine the predictions into a global 3D SSG (Sec. 3.2.5).
3.2.1 Node Feature

For each node \( v_i \in V \), we compute a multi-view image feature \( v_i \) and a geometric feature \( g_i \). We use the former as the initial node feature \( f^v_i = v_i \) and include the latter with a learnable gate in the message passing step (Sec. 3.2.3).

The multi-view image feature is computed by aggregating multiple observations of \( v_i \) on images given by the entity visibility graph. For each view, an image feature is extracted with an image encoding network given the Region-of-Interest (ROI) of the node. The image features are aggregated using a mean operation to the multi-view image feature \( v_i \). Although there are sophisticated methods to compress multi-view image features, such as using gated averaging [15] and learning a canonical representation [63], we empirically found that averaging all the input features [57] yields the best result (see supplementary material). The mean operation also allows incrementally computing the multi-view image feature with a simple moving average. The geometric feature \( g_i \) is computed from the point set \( \mathcal{P}_i \) using a simple point encoder [45].

3.2.2 Edge Feature

For each edge \( e_{i\rightarrow j} \in E_p \), an edge feature \( f^e_{i\rightarrow j} \) is computed using the node properties from its connected two nodes \( v_i \) and \( v_j \) by

\[
\hat{f}^e_{i\rightarrow j} = g_e ([\| v_i - v_j \|, \| v_i - v_j \| \cdot \hat{R}_{i\rightarrow j}]) ,
\]

where \( g_e (\cdot) \) is a Multilayer Perceptron (MLP), \([\cdot] \) denotes a concatenation function, and \( \hat{R}_{i\rightarrow j} \) is a relative pose descriptor which encodes the relative angle between two entities.

The relative pose descriptor is designed to implicitly encode relative angles between two nodes. Using an explicit one is not optimal since OBB estimations do not return the exact pose of an object, which makes explicit pose descriptor not applicable. We instead use the relative geometry properties on a reference frame constructed by the two nodes to implicitly encode the relative pose, as illustrated in Fig. 3. First, we construct a reference frame with the origin the midpoint of the center of two nodes, the x-axis to \( o_j \), the y-axis to the inverse of the gravity direction, and the z-axis the cross product of the x-axis and y-axis. Second, we take maximum and minimum values on each axis of the reference frame to compute the relative pose descriptor as

\[
R_{i\rightarrow j} = \log ( [\| p_i^{\max} \otimes p_j^{\max}, p_i^{\min} \otimes p_j^{\min} ] ) ,
\]

where \( \otimes \) is the Hadamard division, \( p_i^{\max}, p_i^{\min} \in \mathbb{R}^3 \) are the maximum and minimum points on the reference frame for \( \mathbb{R} \in \{ i, j \} \). We use an absolute logarithm ratio to improve the numerical stability.

3.2.3 Message Passing

Given an initial node feature \( f^v_i \) and an edge feature \( f^e_{i\rightarrow j} \), we aggregate the messages from the neighbors for both nodes and edges to enlarge the receptive field and leverage the spatial understanding composition of the environment. We follow [66] by aggregating the messages with a respective GRU unit shared for all nodes and edges. Following, we explain the process taking place in each of the message-passing layers.

First, we incorporate the geometric feature to each node feature using a learnable gate:

\[
\hat{f}^v_i = f^v_i + \sigma ( w^T [f^v_i, g_i]) \sigma (g_i),
\]

where \( \hat{f}^v_i \) is the enhanced node feature, \( \sigma \) denotes a sigmoid function, and \( w^T \) are learnable parameters. The geometric feature may be unreliable, especially when the input geometry is ambiguous or unstable. Thus, we use the learnable gate to learn if the feature should be included. A node message \( m_i \) and an edge message \( m_{i\rightarrow j} \) are computed by

\[
m_i = g_e \left( \left[ \hat{f}^v_i, \max_{j \in \mathcal{N}(i)} \left( \text{FAN} \left( \hat{f}^v_i, \hat{f}^e_{i\rightarrow j}, \hat{f}^e_{j\rightarrow i} \right) \right) \right] \right) ,
\]

\[
m_{i\rightarrow j} = g_e \left( \left[ \hat{f}^v_i, \hat{f}^e_{i\rightarrow j}, \hat{f}^e_{j\rightarrow i} \right] \right) ,
\]

where \( g_e (\cdot) \) and \( g_e (\cdot) \) are MLPs, \( \mathcal{N}(i) \) is the set of indices representing the neighbouring nodes of \( i \), FAN is the feature-wise attention network [64] which weights all neighbour node feature \( \hat{f}^v_j \) using input query \( \hat{f}^v_i \), and key \( \hat{f}^e_{i\rightarrow j} \) given \( j \in \mathcal{N}(i) \).
3.2.4 Class Prediction and Loss Functions

We use the softmax function to estimate the class distribution on both nodes and edges. For multiple predicate estimations, we use the sigmoid function (with a threshold of 0.5) to estimate whether a predicate exists. The network is trained with a cross-entropy loss for classifying both entities and edges. The loss for the edge class is replaced with binary cross entropy for multiple predicates estimation [61].

3.2.5 Fusion

Multiple predictions on the same nodes and edges are fused to ensure temporal consistency. We use the running average approach [8] to fuse predictions [64]. For each entity and edge, we store the full estimated probability estimation $\mu$ and a weight $\rho^t \in \mathbb{R}_{>0}$ at time $t$. Given a new prediction, we update the previously stored $\mu^{t-1}$ and $\rho^{t-1}$ as

$$
\mu^t = \frac{\mu^t \cdot \rho^t + \mu^{t-1} \cdot \rho^{t-1}}{\rho^t + \rho^{t-1}},
$$

where $\rho_{\text{max}}$ is the maximum weight value.

4. Evaluation

We evaluate our method on the task of 3D semantic scene graph estimation (Sec. 4.2) and incremental label association (Sec. 4.3). In addition, we provide ablation studies on the proposed network (Sec. 4.4), and a runtime analysis of our pipeline (Sec. 4.5).

4.1. Implementation Details

In all experiments, we use the default ORB-SLAM3 [3] setup provided by the authors\(^1\) for our IEE front end. For the 2D entity detection, we use EntitySeg [46] with a ResNet50 [20] backbone pretrained on COCO [32] and fine-tuned on the 3RScan [60] training split. For multi-view feature extraction, we use a ResNet18 [20] pretrained on ImageNet [9] without fine-tuning. The point encoder is the vanilla PointNet without learned feature transformation [45]. Regarding hyperparameters, we set $\tau$ to 0.2, $\tau^C$ to 0.5 meters, $\rho_{\text{max}}$ to 100, and the number of message passing layers to 2.

We use the ground truth pose to guide the scene reconstruction because (i) our focus lands on entity detection and scene graph estimation, and (ii) the provided image sequence from 3RScan [60] has a low frame rate (10 Hz), severe image blur, and jittery motion.

4.2. 3D Semantic Scene Graph estimation

For the input types, we compare all methods with the input of ground truth segmentation [61] (GT), geometric segmentation [59] (Dense) and sparse segmentation (Sparse).

For the baseline methods, we compare ours with two 2D methods (IMP [66], and VGfM [15]), and two 3D methods (3DSSG [61] and SGFN [64]).

**Baseline Methods.** We will briefly discuss baseline methods here. Check supplementary for further details. IMP [66] computes a node feature using the image feature cropped from the ROI of the node in an image and computes an edge feature using the union of two ROIs from its connected nodes. Both features are jointly updated with prime-dual message passing and learnable message pooling. VGfM [15] extends IMP by adding geometric features and temporal message passing to handle sequential estimation. 3DSSG [61] extends the ROI concept in IMP [66] in

---

\(^1\)https://github.com/UZ-SLAMLab/ORB_SLAM3.git
3D by replacing ROIs to 3D bounding boxes. The node and edge features are jointly updated with a graph neural network with average message pooling. SGFN [64] improves 3DSSG by replacing the initial edge descriptor with the relative geometry properties between two nodes and introducing an attention method to handle dynamic message aggregation, which enables incremental scene graph estimation.

Implementation. For all methods, we follow their implementation details and train on 3RScan dataset [60] from scratch until converge, using a custom training and test split since the scenes in the original test do not have ground truth scene graphs provided. For IMP [66], since it is a single image prediction method, we adopt the voting mechanism as in [27] to average the prediction over multiple frames. Since ours and other 2D baseline methods rely on image input, we generate a set of keyframes by sampling all input frames using their poses for the GT and Dense inputs (check supplementary material for further details). To ensure diversity in the viewpoint, we filter out a frame if its pose is too similar to any selected frames with the threshold values of 5 degrees in rotation and 0.3 meters in translation.

Evaluation Metric. We report the overall recall (Recall) as used in many scene graph work [34, 61, 64, 66, 68] but with the strictest top-k metric with $k = 1$ as in [62]. In addition, we report the mean recall ($m$Recall) which better indicates model performance when the input dataset has a severe data imbalance issue (see supplementary material for the class distribution). Moreover, since different segmentation methods may result in different number of segments, we map all predictions on estimated segmentation back to ground truth. This allows us to compare the reported numbers across different segmentation methods. We report the Recall of relationship triplet estimation (Rel.), object class estimation (Obj.) and predicate estimation (Pred.), and the $m$Recall of object class estimation and predicate estimation.

Results. Following the evaluation scheme in [64] and in [61], we report two evaluations in Tbl. 1 and Tbl. 2, respectively. The former one maps the node classes to 20 NYUv2 labels [39] to suppress the severe class imbalance in the data as discussed in [62] and estimates a single predicate out of seven support relationship types plus the “same part” relationship to handle over-segmentation. The latter uses 160 node and 26 edge classes with multiple predicate estimation.

In Tbl. 1, overall, it can be seen that all image-based methods (IMP, VGfM, Ours) outperforms points-based methods (3DSSG, SGFN) in almost all object prediction metrics, while the methods based on 3D edge descriptor (3DSSG, SGFN, Ours) tend to have better predicate estimation. This suggests that the 2D representations from images are more representative than 3D, and 3D edge descriptors are more suitable for estimating support types of predicates. By comparing IMP [66] and VGfM [15], it can be seen that the effect of the geometric feature and the temporal message passing is mainly reflected in the $m$Recall metrics. However, it deteriorates the performance when the input is sparse. A possible reason is that the geometric feature is relatively unstable, which decreases the network performance. It is also reflected in the two 3D methods, i.e. 3DSSG and SGFN, where they failed to perform in classifying objects with sparse segmentation while giving a similar performance in the predicate classification. Among all methods, our method outperforms all baselines among all input types and all metrics, apart from the predicate estimation, which has a slightly worse result on some input types. In addition, we report ours using the proposed incremental estimation pipeline, denoted as Ours(i). The incremental estimation process improves slightly in object estimation. The same behavior is also reported in [64]. We show a qualitative result using our full pipeline in Fig. 4.

In Tbl. 2, our method outperforms all other methods in the relationship and object estimation in Recall and object estimation in $m$Recall. 3DSSG [61] has the best results in predicate estimations. This suggests that union 3D bounding boxes are more suitable when estimating multiple predicates.

4.3. Incremental Label Association
We evaluate our label association method in the task of incremental entity segmentation, which aims to estimate accurate class-agnostic segmentation given sequential sensor input, with two baseline methods, i.e. InSeg [58] and PanopticFusion [38].

Baseline Methods. Both baseline methods use reference map approaches as mentioned in Sec. 3.1.3. InSeg [58] considers only the overlapping ratio between labels on an estimated mask and a reference mask. PanopticFusion uses IoU [38] as the evaluation method and limits one reference label can only be assigned to one query label.
Table 3. Evaluation of different label association methods in the task of incremental entity estimation in 3RScan dataset [60].

<table>
<thead>
<tr>
<th>Method</th>
<th>AOS (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>InSeg [58]</td>
<td>38.6</td>
</tr>
<tr>
<td>PanopticFusion [38]</td>
<td>35.9</td>
</tr>
<tr>
<td>Ours</td>
<td>39.6</td>
</tr>
</tbody>
</table>

Implementation. For all methods, we use our IEE pipeline with different label association methods on all training scenes in the 3RScan dataset [60].

Evaluation Metric. We use the Average Overlap Score (AOS) as the evaluation metric [59]. It measures the ratio of the dominant segment of its corresponding ground truth instance. We use the nearest neighbour search to find the ground truth instance label of a reconstructed point. Since our method reconstructs a sparse point map, using the ground truth points as the denominator does not reflect the performance. We instead calculate the score over the sum of all estimated points within the ground truth instance as:

\[
AOS = \sum_i \frac{\max_j \text{Overlap}(S_i, P_j)}{\#(S_i)},
\]

where \(S_i\) is the set of all estimated points with ground truth instance label \(i\).

Results. The evaluation result is reported in Tbl. 3. Our approach achieves the highest AOS, which is 1% higher than InSeg [58] and 3.7% higher than PanopticFusion [38]. The use of a confidence-based approach handles better the label consistency and thus improves the final AOS score. We provide an example of how our method improves temporal consistency under non-uniform distributed points in the supplementary material.

4.4. Ablation Study

We ablate our network with two components, i.e. geometric descriptor \(g_i\) and relative pose descriptor \(R_i \rightarrow j\). The experiment setup is the same as in [64], which makes the ablation comparable to Tbl. 1. The result is reported in Tbl. 4. More ablation studies are in the supplementary material.

Our vanilla network without \(g_i\) and \(R_i \rightarrow j\) outperforms baselines in most of the metrics. With \(g_i\), there is a consistent improvement on all metrics except the Pred. in mRecall. Compared to VGFm [15] in Tbl. 1, VGFm [15] fails to improve the performance of IMP [66] with sparse input. Our gated geometric feature aggregation improves its baseline with sparse input, bringing a more consistent performance gain than VGFm [15]. The \(R_i \rightarrow j\) improves the mRecall performance with \(GT\) and Dense inputs but decreases the model Recall performance. This behavior suggests that \(R_i \rightarrow j\) helps handle class imbalance issues. The combination of both components achieves the best Recall. However, the model tends to focus on dominant classes, resulting in slightly worse performance in mRecall.

Table 4. Ablation study on the proposed network. We ablate the proposed gated geometric feature \((g_i)\) and the relative pose descriptor \((R_i \rightarrow j)\) using the same experiment setup as in Tbl. 1.

<table>
<thead>
<tr>
<th>Method</th>
<th>Recall(%)</th>
<th>mRecall</th>
</tr>
</thead>
<tbody>
<tr>
<td>(g_i) (R_i \rightarrow j) Rel. Obj. Pred.</td>
<td>61.9 76.4 95.6</td>
<td>74.3 69.2</td>
</tr>
<tr>
<td></td>
<td>62.9 77.9</td>
<td>95.9</td>
</tr>
<tr>
<td></td>
<td>60.4 76.3</td>
<td>95.0</td>
</tr>
<tr>
<td></td>
<td>✓ ✓</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>Dense</td>
<td>30.2 54.0</td>
<td>88.5</td>
</tr>
<tr>
<td></td>
<td>33.9 56.4</td>
<td>89.7</td>
</tr>
<tr>
<td></td>
<td>✓ ✓</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>Sparse</td>
<td>28.7 52.7</td>
<td>88.2</td>
</tr>
<tr>
<td></td>
<td>✓ ✓</td>
<td>✓ ✓</td>
</tr>
<tr>
<td></td>
<td>✓ ✓</td>
<td>✓ ✓</td>
</tr>
</tbody>
</table>

Table 5. Runtime [ms] of the different components of our method.

4.5. Runtime

We report the runtime of our system on 3RScan [60] sequence 4acaebcc-6c10-2a2a-858b-29c7e4fb410d in Tbl. 5. The analysis is done with a machine equipped with an Intel Core i7-8700 3.2GHz CPU with 12 threads and a NVidia GeForce RTX 2080ti GPU.

5. Conclusion

We present a novel method that estimates 3D scene graphs from RGB images incrementally. Our method runs in real-time and does not rely on depth inputs, which could benefit other tasks, such as robotics and AR, that have hardware limitations and real-time demand. The experiment results indicate that our method outperforms others in three different input types. The provided ablation study demonstrates the effectiveness of our design. Our vanilla network, without any geometric input and relative pose descriptor, still outperforms other baselines. Our method provides a novel architecture for estimating scene graphs with only RGB input. The multiview feature is proven to be more powerful than existing 3D methods. Our method can be further improved in many directions. In particular, using semi-direct SLAM methods such as SVO [12] might improve the handling of untextured regions where feature-based methods often fail. In addition, the multiview image encoder could be replaced with a more powerful encoder to improve the scene graph estimation with a computational penalty.
References


[52] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, May 9–13 2011.


on Intelligent Robots and Systems (IROS), pages 1419–1426. IEEE, 2017. 1
[58] Keisuke Tateno, Federico Tombari, and Nassir Navab. Real-Time and Scalable Incremental Segmentation on Dense SLAM. In IEEE Conf. Intelligent Robots and Syst., 2015. 2, 4, 7, 8
[66] Danfei Xu, Yuke Zhu, Christopher Choy, and Li Fei-Fei. Scene Graph Generation by Iterative Message Passing. In CVPR, 2017. 1, 2, 4, 5, 6, 7, 8