Hierarchical 3D Scene Graphs Construction Outdoors

Jon Nyffeler[1] Federico Tombari[2] Daniel Barath[1,2,3]
[1]ETH Zürich [2]Google [3]HUN-REN SZTAKI
jon.nyffeler@gmail.com tombari@in.tum.de dbarath@ethz.ch

Abstract

Understanding and structuring outdoor environments in 3D is critical for numerous applications, including robotics, urban planning, and autonomous navigation. In this work, we propose a pipeline to construct hierarchical 3D scene graphs from outdoor data, consisting of posed images and 3D reconstructions. Our approach systematically extracts and organizes objects and their subcomponents, enabling representations that span from entire buildings to their facades and individual windows. By leveraging geometric and semantic relationships, our method efficiently groups objects into meaningful hierarchies while ensuring robust spatial consistency. We integrate efficient feature extraction, hierarchical object merging, and relationship inference to generate structured scene graphs that capture both global and local dependencies. Our approach scales to large outdoor environments while maintaining efficiency, and we demonstrate its effectiveness on real-world datasets. We also demonstrate that these constructed outdoor scene graphs are beneficial for downstream applications, such as 3D scene alignment. The code is available on GitHub.

1. Introduction

Understanding outdoor 3D environments is a fundamental challenge in computer vision, with applications in robotics, urban planning, and autonomous navigation. Despite its significance, general outdoor 3D scene understanding remains underdeveloped compared to its indoor counterpart [29]. While indoor environments are often well-structured and constrained, outdoor scenes exhibit greater variability, occlusions, and large-scale complexities, making robust 3D scene reconstruction and interpretation considerably more challenging.

3D scene graphs are particularly valuable for various downstream applications [13, 21, 22] due to their compact and structured representation of environments. However, existing scene graph construction methods primarily focus on indoor settings [10] or autonomous driving scenarios [8]. Indoor scene graph approaches leverage structured environments to establish hierarchical relationships between objects and regions [2, 10, 18]. In contrast, autonomous driving methods rely on LiDAR-based object detection and tracking in urban environments, where predefined object categories such as vehicles, pedestrians, and road structures dominate [4, 9]. These approaches are often unsuitable for general outdoor scene graph generation, as they struggle to capture unstructured environments beyond road networks and traffic-related elements. Moreover, many state-of-the-art 3D object detection models fail to generalize to diverse outdoor settings where object categories extend beyond those found in urban datasets.

Figure 1. **Point Cloud of Scene LIN** from the LaMAR dataset [23]. Large object instances (e.g., houses) are distinctly colored, with their components (e.g., windows) shaded in a darker variant of the same color to indicate hierarchical relationships.

A gap exists in research regarding scene graph generation for general outdoor environments. Existing methods either focus on object-level scene understanding without hierarchical relationships or cannot generalize beyond structured urban domains. A framework capable of capturing multi-scale relationships – ranging from entire buildings to facades and individual windows – while maintaining spatial and semantic consistency is essential to bridging this gap.

This work proposes a pipeline for hierarchical 3D scene graph construction from outdoor data, leveraging posed images and 3D reconstruction. Unlike existing hierarchical approaches, *e.g.*, [18, 28], which are often tailored for structured indoor environments or rely on predefined tax-

onomies, our method dynamically constructs hierarchies by identifying multi-scale object relationships within unstructured outdoor scenes. To our knowledge, this is among the first approaches to demonstrate that hierarchical 3D scene graphs can be feasibly and effectively generated in complex outdoor settings using only posed images and reconstructions. Our method extracts objects and their subcomponents, constructing a structured representation that spans multiple abstraction levels. By combining geometric and semantic relationships, our approach enables efficient hierarchical grouping. We demonstrate the effectiveness of our method on real-world outdoor datasets and validate its utility in downstream applications such as 3D scene alignment. Our key contributions are:

- We introduce a hierarchical 3D scene graph representation for outdoor environments, capturing relationships from large-scale structures to fine-grained components.
- We propose an efficient pipeline for generating these scene graphs using posed images and 3D reconstructions, incorporating geometric and semantic constraints.
- We evaluate our approach on real-world outdoor datasets and demonstrate its benefits for downstream applications such as 3D scene alignment.

2. Related Work

3D Scene Graph Generation. Early work on 3D scene graphs primarily focused on indoor environments. Armeni et al. [2] introduced one of the first 3D scene graph frameworks, constructing a graph that spans entire buildings, with nodes representing objects, rooms, and cameras. Their semi-automatic pipeline combined 2D object detectors with multi-view geometric consistency to populate the graph.

Rosinol et al. [18] extended this to dynamic and large-scale settings by building a 3D dynamic scene graph from visual-inertial SLAM data. Their system integrates object detection and human pose estimation into SLAM, producing a layered representation of objects, places, and rooms.

Recent efforts have explored outdoor scene graphs, introducing challenges such as complex semantics and sparse labeled data. Strader et al. [25] propose a method that uses a language-driven spatial ontology to structure outdoor scene graphs. By leveraging large language models, their approach reduces reliance on labeled training data while enabling predictions for unseen concepts.

Parallel research explores learning-based 3D scene graph prediction, particularly in point cloud data. Some methods integrate commonsense knowledge graphs [7] to improve relationship prediction. Overall, scaling scene graph methods from structured indoor environments to unstructured outdoor scenes remains a challenge.

Outdoor Scene Understanding requires semantic interpretation and large-scale geometric reconstruction. Datasets such as KITTI [9], Cityscapes [6], and nuScenes [4] have driven advancements in multi-modal perception and autonomous driving. SemanticKITTI [3] further provides point-wise semantic labels for LiDAR sequences, aiding in point cloud segmentation research.

Traditional methods relied on geometric reasoning, such as removing ground planes from LiDAR data and clustering objects before classification. More recent deep learning models predict semantic labels directly from sensor data, improving segmentation accuracy. CNNs process panoramic images for road and object segmentation, while 3D deep learning models handle volumetric data. These advancements form the foundation for scene graph construction, where nodes represent objects or regions and edges encode their relationships.

Object Detection and Segmentation in 3D is essential for building scene graphs. Traditional methods used geometric reasoning, such as clustering point clouds and fitting models [19], but these struggled with fine semantic distinctions.

Deep learning has significantly improved 3D object detection. PointNet [15] introduced direct learning on point clouds, with PointNet++ refining local feature learning. Frustum PointNet [16] combined 2D and 3D data, while voxel-based methods like PointPillars [12] introduced efficient encoding techniques. Hybrid approaches integrate deep networks with geometric constraints to refine predictions. Deep learning now dominates 3D object detection, but geometric reasoning remains useful for improving generalization to unseen scenarios.

Semantic and Spatial Relationship Inference. Modeling relationships of objects is central to 3D scene graph construction. Spatial proximity can infer adjacency or containment, *e.g.*, a car on a road or a tree next to a car. Some frameworks, such as Rosinol et al. [18], construct hierarchical scene graphs by clustering environments into places and rooms. Semantic reasoning further refines relationships. Certain objects have canonical relations, such as a person inside a vehicle. Recent 3D scene graph models incorporate relational labels [10], using training data or external knowledge sources, like large language models. Strader et al.[25] integrated logical rules to enrich scene graphs, improving structured relationship inference.

By combining spatial clustering, semantic understanding, and external knowledge, modern scene graph methods better capture real-world relationships. These techniques support applications such as robotic planning, scene understanding, and question answering.

3. Hierarchical Scene Graph Construction

Our method constructs hierarchical 3D scene graphs from outdoor scenes, leveraging posed images and a 3D mesh reconstruction. The pipeline consists of four main stages: (i) mesh preprocessing, (ii) per-image object proposal and hierarchical merging, (iii) tree pruning and refinement, and

(iv) label/description generation and hyperedge construction. Figure 2 provides an overview of our approach.

3.1. Mesh Preprocessing

The input 3D mesh often requires preprocessing to improve the accuracy and efficiency of subsequent steps. This may involve downsampling the mesh to reduce computational cost, and removing potential artifacts that can occur, *e.g.*, due to noise or the reconstruction method. The specific preprocessing steps can be tailored to the characteristics of the input data, but the goal is to ensure a consistent and clean representation for downstream processing. Let $\mathcal{M}$ denote the preprocessed 3D mesh that we will segment to objects.

3.2. Per-Image Object Proposal and Merging

This step forms the core of our approach. For each image I_i with associated camera pose P_i, we perform the following steps to extract object instances.

2D Object Proposal. We employ a segmentation model (*e.g.*, SAM [11]) to generate a set of 2D mask proposals, denoted as $\{M_{i,j}\}_{j=1}^{N_i}$, where N_i is the number of masks for image I_i. For each image, we sort these masks in descending order of their area, prioritizing larger objects, which are often more likely to represent significant scene structures. Other sorting criteria, such as confidence scores, could also be incorporated if the segmentation network outputs them.

Feature Extraction. For each mask $M_{i,j}$, we extract a feature vector $\mathbf{f}_{i,j} \in \mathbb{R}^d$ using a pre-trained visual encoder (*e.g.*, DINOv2 [14]). To enhance robustness and account for potential inaccuracies in the mask boundaries, we generate multiple crops [26] of the image region defined by the bounding box of $M_{i,j}$, with variations in padding or masking, and then aggregate (*e.g.*, average) the features extracted from these crops to obtain $\mathbf{f}_{i,j}$, *i.e.*, the segment feature.

3D Projection and Refinement. We project each 2D mask $M_{i,j}$ into 3D by casting rays from the camera center through each pixel within the mask and intersecting them with the preprocessed mesh $\mathcal{M}$. This yields an initial point cloud $\mathcal{C}_{i,j}$. We downsample $\mathcal{C}_{i,j}$ using a standard technique like voxel downsampling with a voxel size v. To remove noise and isolate the dominant object, we use a density-based clustering algorithm (DBSCAN [24]) and retain only the largest cluster, resulting in a refined point cloud $\mathcal{C}'_{i,j}$.

Object Initialization and Merging. We process the point clouds $\{\mathcal{C}'_{i,j}\}$, refined in the previous step, sequentially, based on the sorted mask order. For each $\mathcal{C}'_{i,j}$, we determine whether to initialize this object as a new one or to merge it with an existing object instance in the graph. Let $\mathcal{O} = \{O_k\}_{k=1}^{K}$ be the set of existing object instances in the graph, where each object O_k is represented by its point cloud $\mathcal{P}_k$, semantic feature vector $\mathbf{f}_k$ from DINOv2, and the set of image indices $\mathcal{V}_k$ from which it was observed.

We first identify candidate objects for merging by computing a measure of spatial overlap (*e.g.*, 3D bounding box intersection) between $\mathcal{C}'_{i,j}$ and each existing object O_k. To accelerate this process, we utilize the camera frustum defined by P_i to pre-filter objects that are entirely outside the field of view. This step is crucial, as it allows for quasi *linear* time in scene size, otherwise looking through all objects would scale quadratically. For each candidate object O_k deemed sufficiently close to $\mathcal{C}'_{i,j}$, we compute the point cloud intersection $\mathcal{I}_{j,k} = \mathcal{C}'_{i,j} \cap \mathcal{P}_k$ and build the set $\bar{\mathcal{O}}_j$ of all existing objects with non-zero overlap. We calculate a percentage of intersection $p_{j,k}$ as follows:

$$p_{j,k} = \max\left(\frac{|\mathcal{I}_{j,k}|}{|\mathcal{C}'_{i,j}|}, \frac{|\mathcal{I}_{j,k}|}{|\mathcal{P}_k|}\right). \tag{1}$$

We then compute similarity score $s_{j,k}$ of $\mathcal{C}'_{i,j}$ and O_k:

$$\begin{aligned} \sigma_{j,k} &= \text{cosine\_similarity}(\mathbf{f}_{i,j}, \mathbf{f}_k) \\ s_{j,k} &= \alpha_1 p_{j,k} \mathbf{1}_{\{p_{j,k} > \tilde{\tau}_1\}} + \alpha_2 \sigma_{j,k} \mathbf{1}_{\{\sigma_{j,k} > \tilde{\tau}_2\}}. \end{aligned} \tag{2}$$

where $\sigma_{j,k}$ can be interpreted as a semantic similarity. α_i are weights to influence the importance of geometric or semantic similarity and $\mathbf{1}$ is the indicator function ensuring a minimum value.

We select object O_{k^*} with the highest similarity, such that $k^* = \arg\max_k s_{j,k}$. If $s_{j,k^*} > \tau_1$, where τ_1 is a merging threshold, we merge $\mathcal{C}'_{i,j}$ into O_{k^*}. The merged object point cloud, feature vector, and view set are updated as:

$$\mathcal{P}_{k^*} \leftarrow \mathcal{P}_{k^*} \cup \mathcal{C}'_{i,j}, \tag{3}$$

$$\mathbf{f}_{k^*} \leftarrow \frac{|\mathcal{V}_{k^*}| \cdot \mathbf{f}_{k^*} + \mathbf{f}_{i,j}}{|\mathcal{V}_{k^*}| + 1}, \tag{4}$$

$$\mathcal{V}_{k^*} \leftarrow \mathcal{V}_{k^*} \cup \{i\}. \tag{5}$$

We iteratively merge O_{k^*} with other candidate objects from $\bar{\mathcal{O}}_j$ as long as the similarity score exceeds τ_1, recomputing the similarity score after each merge. If $s_{j,k^*} \leq \tau_1$, we initialize a new object O_{K+1} with $\mathcal{P}_{K+1} = \mathcal{C}'_{i,j}$, $\mathbf{f}_{K+1} = \mathbf{f}_{i,j}$, and $\mathcal{V}_{K+1} = \{i\}$, and increment K.

Hierarchical Tree Construction. After initializing or merging an object, we integrate it into a hierarchical object tree. Each object is initially considered a tree of height zero. We define the *root* of a tree as its highest-level node. Let O_{new} denote the newly initialized or merged object.

We first construct a list of candidate tree levels, $\mathcal{L}$, by including all objects left in $\bar{\mathcal{O}}_j$ that weren't already merged to O_{new}, along with all their parent nodes up to the root. In Fig. 3 the levels $\mathcal{L}$ are colored orange.

Next, we compute similarity scores between the root of O_{new} and each node L_m in $\mathcal{L}$. This similarity score, $s'_{new,m}$, uses a modified intersection percentage as:

$$p'_{new,m} = \text{mean}\left(\frac{|\mathcal{I}_{new,m}|}{|\mathcal{P}_{new}|}, \frac{|\mathcal{I}_{new,m}|}{|\mathcal{P}_m|}\right), \tag{6}$$

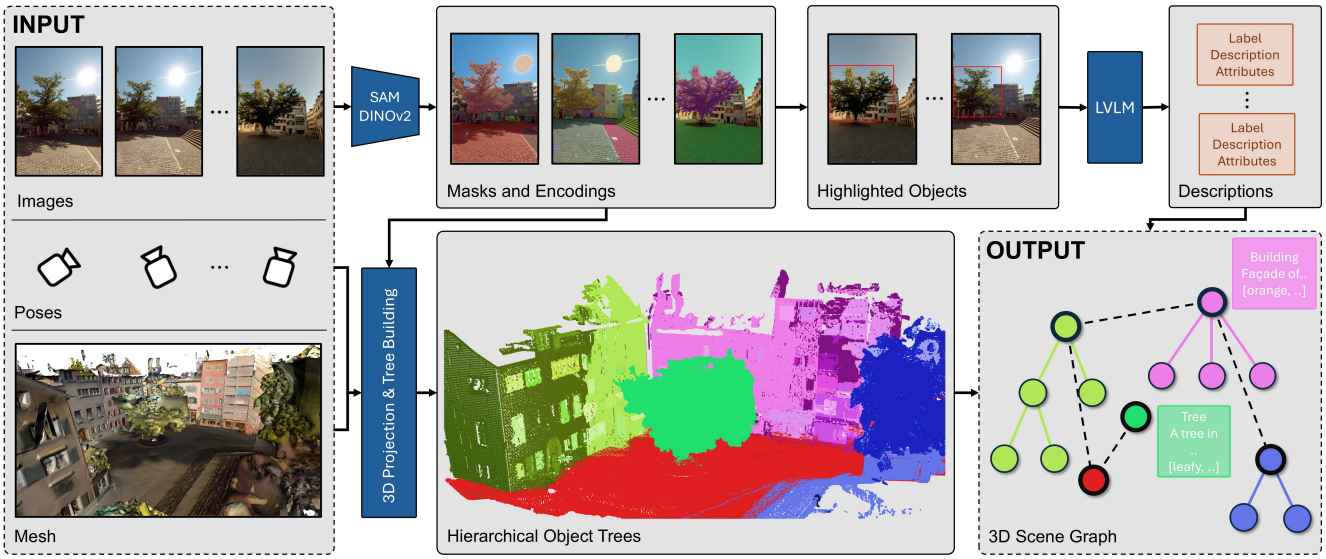


Figure 2. **Overview of the Proposed Method.** Given input images and a mesh, object masks are detected using Segment Anything [11] and projected onto the mesh to obtain object instance segmentation. Images of each object instance are then processed by an LVLM model, which generates labels, descriptions, and attributes. The detected objects are subsequently aggregated into a hierarchical 3D scene graph, enabling a structured and high-level understanding of the scene. The scene graphs are useful in various tasks, *e.g.*, scene alignment [21]

$$\sigma_{new,m} = \text{cosine\_similarity}(\mathbf{f}_{root(new)}, \mathbf{f}_m)$$
$$s'_{new,m} = \beta_1 p'_{new,m} \mathbf{1}_{\{p'_{new,m} > \tilde{\gamma}_1\}} + \beta_2 \sigma_{new,m} \mathbf{1}_{\{\sigma_{new,m} > \tilde{\gamma}_2\}}, \tag{7}$$

where $\mathcal{P}_{new}$ and $\mathcal{P}_m$ are point clouds of O_{new} and L_m, $\mathcal{I}_{new,m}$ is the intersection between those two and β_i, γ_i are new parameters that allow for a different type of merging. We select the node L_{m^*} with the highest similarity: $m^* = \arg\max_m s'_{new,m}$. If $s'_{new,m^*} > \tau_2$, where τ_2 is the tree merging threshold, we perform one of the following actions based on a set of merging rules:

1. If both O_{new} and L_{m^*} are trees of height zero: Create a new parent O_{parent}, merging the point clouds of O_{new} and L_{m^*}. O_{new} and L_{m^*} become children of O_{parent}.
2. If either O_{new} or L_{m^*} is a tree of height zero: Append the height-zero tree as a child of the other node, merging their respective point clouds.
3. If both O_{new} and L_{m^*} are roots: Append the smaller tree (in terms of point cloud size) as a child of the larger tree.
4. Otherwise: Append O_{new} as a child of L_{m^*}.

Every time two point clouds are merged, we also merge the feature vectors by taking a weighted average according to $|\mathcal{V}_{new}|$ and $|\mathcal{V}_{m*}|$ and concatenate $\mathcal{V}_{new}$ and $\mathcal{V}_{m*}$ to form the merged $\mathcal{V}$. We iteratively repeat this process, taking the root of the updated tree and recomputing similarity scores with respect to $\mathcal{L}$, until all similarity scores are below τ_2. The merging rules are visualized in Fig. 3.

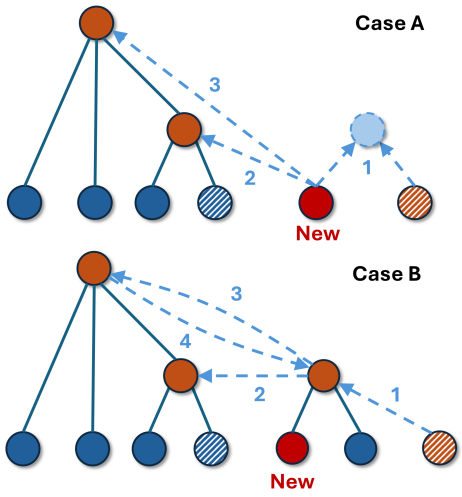


Figure 3. **Merging Rules in Hierarchical Tree Construction.** Nodes with point cloud intersections with the initial object ($\bar{\mathcal{O}}_j$) are represented with stripes, while those involved in similarity score calculations ($\mathcal{L}$) are highlighted in orange. In case A, if the new object is a tree of height zero, it is either appended as a child to the best-matching level of an existing tree or merged with another height-zero tree to form a new tree. In case B, the root of the new object is considered in all tree merging operations and is either appended as a child to another root or has a child appended to it, depending on point cloud size in cases 3 and 4, respectively.

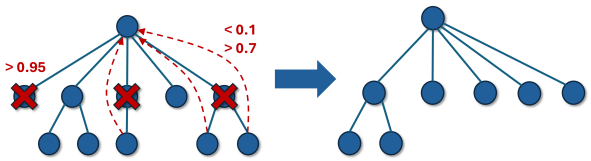


Figure 4. **Tree Pruning and Refinement.** Illustration of node removal and the reassignment of children to their respective parents during the pruning process.

3.3. Tree Pruning and Refinement

After processing all images, we perform a post-processing step to refine the object trees. First, we can remove objects observed by fewer than a minimum number of views (*e.g.*, $|\mathcal{V}_k| < 2$). Then, for each tree, we apply a set of general pruning/simplification rules as follows:

1. **Dominant Child Removal:** Remove leaf nodes that are excessively large relative to their parent (*e.g.*, exceeding a point cloud size ratio threshold). The parent point cloud remains unchanged.
2. **Single Child Parent Removal:** Remove any parent node that has only one child.
3. **Dominant Parent Removal:** Remove any parent node that is excessively large or disproportionately small relative to its parent (*e.g.*, exceeding a point cloud size ratio threshold). The children of the removed node are directly attached to its parent.

Visualization for this step is shown in Fig. 4.

3.4. Label, Attributes and Hyperedge Construction

Finally, we traverse the object trees to extract a complete set of nodes representing objects at different levels of granularity. We generate descriptive labels and attributes for each node and establish hyperedges to capture relationships.

Labeling and Description. For each object O_i, we select a subset of images $I^* = \{I_k\}_{k \in K^*} \subset \mathcal{V}_i$ (*e.g.*, $|K^*| = 3$) for label and description generation. Image selection is based on a score that combines the number of object points that project in front of the camera view (favoring views that capture as much of the object as possible), the projected 2D mask area, and the mask compactness (*i.e.*, the ratio of the mask area and the area of its bounding box). More formally, we can express such a visibility score as:

$$\text{score}(O_i, I) = \text{num\_visible\_points}(O_i, I) * \text{mask\_area}(O_i, I) * \text{mask\_compactness}(O_i, I).$$

We select $\arg\max\{\text{score}(O_i, I)\}$ as I_{k_0}. We sort I according to $\text{score}(O_i, I)$ and iteratively select additional images, by going through the sorted list and picking image I_k if its pose P_k is sufficiently different from all poses of selected images $\{P_{k_i}\}_i$. We consider pose P_i sufficiently different from pose P_j if either the distance of their camera centers or the angle between their orientations exceed thresholds. This ensures a rich selection of different views on the object, enhancing the description generation.

For each selected image, we visually highlight the object (by drawing a bounding box around the projected object mask) and prompt a vision-language model (VLM) (*e.g.*, GPT-4V [1], though any capable VLM can be used) to generate a label, a description, a list of attributes, and optionally, a confidence score for its prediction. To synthesize the information, the resulting textual outputs are distilled into a single, coherent label, description, and attribute list by a large language model GPT-4 [1]).

Hyperedge Construction. We construct several types of hyperedges to capture different relationships:

1. **Root-to-Root Edges:** We compute a measure of spatial proximity or overlap (*e.g.*, 3D bounding box IoU) between all pairs of root nodes. We then construct a minimum spanning tree based on these values to prune redundant edges, resulting in a set of edges connecting spatially related root nodes.
2. **Parent-Child Edges**: Every object is connected to its parent in the tree (if a parent exists). This directly encodes the hierarchical structure.
3. **Description-Based Hyperedges (Root Nodes Only):** We encode the generated descriptions of all root nodes using a sentence embedding model (SBERT [27]). We then perform DBSCAN-based clustering on these embeddings using cosine similarity, creating hyperedges that group semantically similar objects.
4. **Spatial Hyperedges (Root Nodes Only):** We perform DBSCAN clustering on the 3D centroids of all root nodes. Each cluster forms a hyperedge, grouping spatially proximate objects.

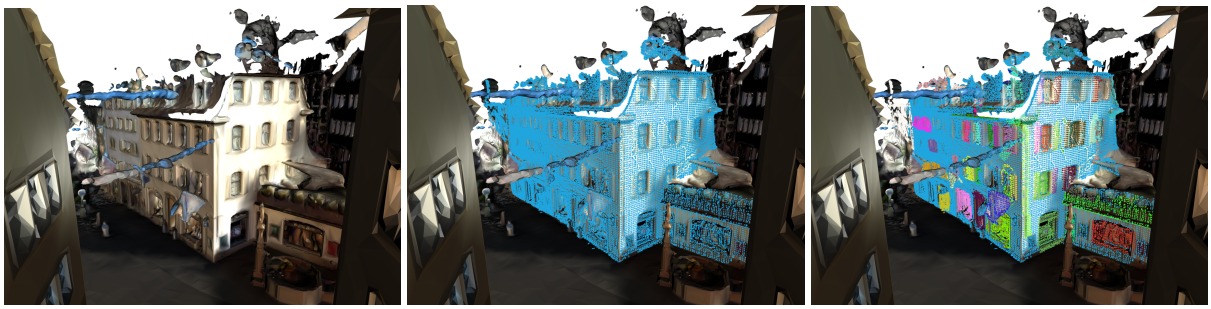
The final output of our method is a hierarchical 3D scene graph, where nodes represent objects at multiple levels of granularity, and edges/hyperedges represent spatial, semantic, and hierarchical relationships.

4. Experiments

In this section, we provide details on the dataset used for scene graph construction together with analysis on the extracted object classes and scene graph structure.

4.1. Dataset and Scene Graph Generation

To evaluate our method, we construct large-scale 3D outdoor scene graphs using the LaMAR dataset [23]. LaMAR is a benchmark dataset designed for augmented reality (AR) localization and mapping, featuring diverse large-scale indoor and outdoor environments captured using AR devices such as HoloLens 2 and iPhones/iPads. The dataset provides multi-sensor data streams, including RGB images,



(a) Processed mesh (b) Large objects (c) Small objects.

Figure 5. **Hierarchical Levels of an Object Tree.** From left to right, we visualize (a) the processed mesh, (b) a root object instance, and (c) its child objects. The hierarchical structure enables representation of both large-scale objects (*e.g.*, entire buildings) and smaller components (*e.g.*, windows, doors) within the same scene graph structure.

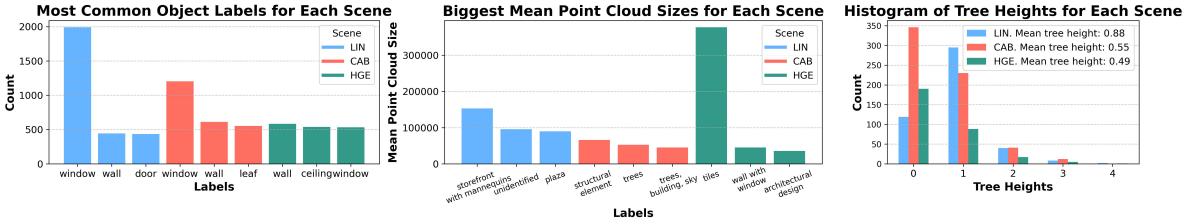


Figure 6. **Analysis of the Generated Scene Graphs.** From left to right, the histograms illustrate (i) the count of the three most common object labels in each scene, (ii) the mean point cloud size for the three largest object labels per scene, and (iii) the distribution of tree heights across scenes. Additionally, we report the mean tree height for each scene.

depth maps, IMU readings, and 3D laser scans, covering over 45,000 square meters of real-world urban and natural environments. The ground-truth alignment pipeline robustly handles heterogeneous device data by aligning AR trajectories to high-precision laser scans, ensuring accurate localization and mapping.

For our experiments, we focus on the three large scenes from LaMAR (LIN, CAB, and HGE). Using these scenes, we generate hierarchical 3D scene graphs, as discussed previously, that capture the relationships between large-scale objects and their subcomponents (*e.g.*, buildings, facades, and windows). Our scene graphs encode both geometric and semantic relationships, allowing for structured reasoning over large-scale environments.

Fig. 5 presents a sample section of the generated 3D scene graph. The left plot highlights a building within the processed mesh. The middle plot visualizes a root scene graph node, with the corresponding building emphasized by coloring its point cloud. The right plot illustrates a lower hierarchical level, showcasing the child nodes of the building, such as windows and doors. Another example is shown in Fig. 1 with large objects denoted by color and smaller parts of each large object with a darker shade of the base color.

Figure 6 presents an analysis of the constructed scene graphs. The left plot shows the most frequently occurring objects in each dataset (*i.e.*, windows). The middle plot reports the average point cloud sizes of the largest objects. Additionally, we provide the distribution of tree heights in the generated scene graphs.

4.2. Scene Alignment using SGAligner

Since no existing outdoor 3D scene graphs are available, we lack a direct baseline for comparison. Instead, we aim to demonstrate the utility of our 3D scene graphs by applying them to the task of scene alignment using SGAligner [21], a cross-modal scene alignment method. SGAligner constructs node embeddings from various modalities of a 3D scene graph, encoding information about *objects*, *relationships*, and *attributes*. Specifically, it learns to represent individual object instances as high-dimensional embedding vectors. Given two scene graphs with overlapping spatial regions, SGAligner optimizes these embeddings so that corresponding nodes are placed close together, enabling accurate object-to-object matching between scenes.

This matching allows for scene alignment by identifying multiple object correspondences and applying point cloud registration to their respective point clouds while ignoring the rest of the scene. Assuming that most objects remain unchanged between the two scans, the rigid transformation that aligns the majority of matched objects in a shared coordinate system is considered the final scene transformation.

We adapt SGAligner for large-scale outdoor settings by

modifying its node embeddings. Specifically, we:

- Increase the PointNet-based downsampling from 512 to 1024 points per object.
- Replace standard graph attention network (GAT) node embeddings with sentence encodings of object descriptions using SBERT.
- Introduce Hypergraph Attention Network (HGN) embeddings, where each hyperedge embedding is computed as the mean of its associated node embeddings.

We perform node alignment on the root nodes only, due to the large number of nodes. An example of object matching between two scenes can be seen in the supplementary material. While we focus on alignment in this work, the same object-level embeddings can support a broader range of downstream tasks, including visual and text-based localization (e.g. SceneGraphLoc [13] or "Where Am I?" [5]).

Generating Sub-Scenes for Alignment. Note that while our method for scene graph generation is entirely training-free, SGAligner requires supervised training on pairs of scenes with spatial overlap. We synthetically create sub-scenes from the large LaMAR scenes (i.e. our generated scene graphs) to generate training data. We sample a point uniformly at random from a sphere in the center of the scene and construct a plane going through that point of origin. The orientation of the plane is also randomly sampled, whilst ensuring, keeping the vertical anlge of the plane within a threshold. All objects on one side of the plane are discarded. If an object is intersected by the plane, only the portion on one side is retained. Edges between objects are preserved only if both objects remain in the sub-scene, and hyperedges are retained if at least one of their objects is included. The process is repeated twice to generate a pair of sub-scenes with certain overlap, where overlap percentage refers to the ratio of shared points between two sub-scene point clouds relative to the total number of points in their union. A histogram on overlap can be seen in Fig. 7.

To evaluate generalization, we train on one scene (*e.g.*, LIN) and test node alignment on the other two (HGE and CAB). Each dataset split consists of 1000 training pairs, 200 validation pairs, and 250 test pairs per scene. We also analyze the impact of different modalities during training.

Following the evaluation protocol of SGAligner, we measure object retrieval recall to assess whether objects can be correctly matched to their corresponding pairs among all objects in the environment. We report recall at $K \in \{1, 2, 3, 4, 5\}$, where, for example, $K = 5$ measures how often the correct object appears among the top five predictions of the SGAligner model trained on our scenes.

Evaluation Results. Table 1a presents the retrieval recall when training on scene LIN and testing on CAB and HGE. The first column specifies the modalities used during training: point cloud ($\mathcal{P}$), structure ($\mathcal{S}$), hyperedge ($\mathcal{H}$), and attributes ($\mathcal{A}$). Across all modality combinations, the recall rate approaches 100%, demonstrating the informativeness of the constructed 3D scene graphs. In all cases, the mean retrieval recall (RR) exceeds 93%, highlighting the robustness of our method.

Modalities	Mean RR ↑	Hits @↑				
		K = 1	K = 2	K = 3	K = 4	K = 5
$\mathcal{P}$	0.963	0.955	0.961	0.967	0.969	0.972
$\mathcal{P}+\mathcal{S}$	0.939	0.931	0.936	0.942	0.945	0.948
$\mathcal{P}+\mathcal{H}$	0.967	0.959	0.966	0.971	0.973	0.975
$\mathcal{P}+\mathcal{S}+\mathcal{H}$	0.943	0.932	0.941	0.949	0.952	0.956
$\mathcal{H}+\mathcal{A}$	0.996	0.994	0.997	0.998	0.998	0.998

(a) Trained on LIN.

Modalities	Mean RR ↑	Hits @↑				
		K = 1	K = 2	K = 3	K = 4	K = 5
$\mathcal{P}$	0.944	0.931	0.941	0.95	0.954	0.959
$\mathcal{P}+\mathcal{S}$	0.865	0.845	0.862	0.873	0.88	0.885
$\mathcal{P}+\mathcal{H}$	0.938	0.92	0.94	0.951	0.957	0.962
$\mathcal{P}+\mathcal{S}+\mathcal{H}$	0.855	0.832	0.849	0.863	0.873	0.882
$\mathcal{H}+\mathcal{A}$	0.941	0.929	0.939	0.947	0.951	0.954

(b) Trained on CAB.

Modalities	Mean RR ↑	Hits @↑				
		K = 1	K = 2	K = 3	K = 4	K = 5
$\mathcal{P}$	0.942	0.93	0.94	0.948	0.953	0.958
$\mathcal{P}+\mathcal{S}$	0.892	0.879	0.887	0.896	0.9	0.904
$\mathcal{P}+\mathcal{H}$	0.944	0.933	0.942	0.95	0.955	0.959
$\mathcal{P}+\mathcal{S}+\mathcal{H}$	0.896	0.884	0.892	0.9	0.904	0.908
$\mathcal{H}+\mathcal{A}$	0.971	0.963	0.97	0.976	0.979	0.982

(c) Trained on HGE.

Table 1. **Evaluation of Node Matching Across Different Training Scenes.** Each subtable shows the performance when trained on a specific scene and evaluated on the two remaining ones. The reported scores represent the average retrieval performance across both testing scenes.

Similar trends are observed in Tables 1b and 1c, confirming the consistency of the generated scene graphs across different environments. These results demonstrate the suitability of our hierarchical scene graph representation for accurate object retrieval and scene alignment.

We also present a summary of the mean reciprocal ranks over all model types when the model is trained on one

Train \ Test	**CAB**	**HGE**	**LIN**
CAB	—	0.933	0.884
HGE	0.951	—	0.907
LIN	0.961	0.962	—

Table 2. **Mean Reciprocal Rank.** Comparison of matching performance across different training and testing scene combinations, averaged over all models.

Method	LIN			CAB			HGE		
	RRE↓	RTE↓	Time (s)	RRE↓	RTE↓	Time (s)	RRE↓	RTE↓	Time (s)
Baseline	32.049	20.727	41.995	31.283	14.453	14.794	30.037	14.14	2.644
SGAligner + FPFH	0.279	2.241	**8.283**	0.075	1.552	**9.384**	0.879	2.516	**5.981**
SGAligner + GeoTr.*	**7.448e-05**	**5.864e-05**	85.245	**5.196e-05**	**2.712e-05**	57.989	**9.414e-05**	**4.636e-05**	30.683

Table 3. **Results of point cloud registration.** Mean relative rotation error (RRE; in degrees), mean relative translation error (RTE; in meters), and average runtime (in seconds) for the baseline method (FPFH features [20] on the entire scene) and SGAligner [21] (using FPFH and GeoTransformer [17] matches) trained on the proposed outdoor 3D scene graphs of the three test scenes. * Out of the 1500 scenes, the official code of GeoTransformer crashed on 53.

dataset and tested on another in Table 2. While minor variations exist, all training and testing combinations yield stable results, demonstrating the robustness of our method across a diverse range of scenes.

4.3. Point Cloud Matching for Scene Alignment

We perform 3D point cloud matching using a baseline method as well as our prefiltering strategy, which leverages object-to-object matching to refine alignment. We evaluate the following three approaches:

- Direct point cloud matching and registration using Open3D's fast global registration algorithm [30], based on FPFH features [20].
- Independent point cloud registration applied to retrieved object-to-object matches using the same FPFH-based method [20]. The final transformation is estimated from *all* these object point correspondences by RANSAC.
- Independent point cloud registration on the retrieved object-to-object matches using GeoTransformer [17]. The final transformation is estimated from *all* these object point correspondences by RANSAC.

For the direct point cloud method we downsample each subscene to 30,000 points using a voxel size of 0.6 meters to ensure computational efficiency, given the large scene sizes. We do not apply GeoTransformer to match the entire scene directly, as done in SGAligner, since it fails entirely on these large outdoor environments due to scalability issues.

Metrics. We evaluate alignment accuracy using the mean relative rotation error (RRE, in degrees) and mean relative translation error (RTE, in meters). Additionally, we report the average runtime in seconds for each method.

Results. Table 3 presents the results of point cloud registration for each method and each scene. We train the model on each scene and test on the two others. For each test scene, we average the results we achieved when training on the other two scenes and report it in the table. The baseline method, which applies FPFH features to the entire scene, performs significantly worse than the other approaches, resulting in orders-of-magnitude higher RRE and RTE.

In contrast, integrating SGAligner with the proposed scene graphs leads to substantial improvements, even when using standard FPFH features on object-to-object matches. When leveraging object matches estimated by SGAligner with GeoTransformer, we achieve near-perfect alignment, with both translation and rotation errors in the 10^{-5} range. These results highlight the effectiveness of scene graphs for improving outdoor scene alignment. Across all cases, the baseline method struggles to achieve accurate alignment, whereas SGAligner, when combined with the proposed scene graphs, consistently improves performance using both FPFH and GeoTransformer correspondences.

Additionally, Fig. 7 shows that SGAligner with FPFH achieves low RTE at minimal overlap, with error dropping quickly, unlike the slower decline seen in traditional FPFH.

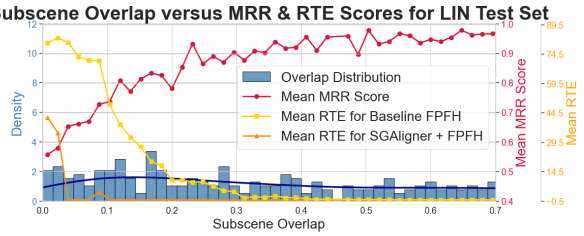


Figure 7. **Sub-Scene Overlap Analysis.** Distribution of sub-scene overlap percentage and according test scores for LIN.

5. Conclusion

We presented a method for generating hierarchical 3D scene graphs from large-scale outdoor environments using posed images and 3D reconstructions. Our approach systematically extracts and organizes objects into structured hierarchies, capturing relationships from entire buildings to their subcomponents, such as facades and windows. By leveraging geometric and semantic relationships, we ensure spatial consistency and meaningful scene representation. Experimental results on the LaMAR dataset demonstrate the robustness and informativeness of our scene graphs in the downstream application of cross-modal 3D scene alignment. Our method provides a scalable framework for structured outdoor 3D scene understanding, with applications in robotic navigation, AR localization, and large-scale mapping. The code and scene graphs are available at GitHub.

Acknowledgement. The work has been supported by the ETH Zurich Career Seed Award.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 5

[2] Iro Armeni, Zhi-Yang He, JunYoung Gwak, Amir R. Zamir, Martin Fischer, Jitendra Malik, and Silvio Savarese. 3d scene graph: A structure for unified semantics, 3d space, and camera. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1991–2000, 2019. 1, 2

[3] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. SemanticKITTI: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9297–9307, 2019. 2

[4] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yin Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11621–11631, 2020. 1, 2

[5] Jiaqi Chen, Daniel Barath, Iro Armeni, Marc Pollefeys, and Hermann Blum. "where am i?" scene retrieval with language. In *Computer Vision – ECCV 2024*, pages 201–220, Cham, 2025. Springer Nature Switzerland. 7

[6] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3213–3223, 2016. 2

[7] Linfang Ding, Guohui Xiao, Albulen Pano, Mattia Fumagalli, Dongsheng Chen, Yu Feng, Diego Calvanese, Hongchao Fan, and Liqiu Meng. Integrating 3d city data through knowledge graphs. *Geo-spatial Information Science*, pages 1–20, 2024. 2

[8] Tobias Fischer, Lorenzo Porzi, Samuel Rota Bulo, Marc Pollefeys, and Peter Kontschieder. Multi-level neural scene graphs for dynamic urban environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21125–21135, 2024. 1

[9] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012. 1, 2

[10] Qiao Gu, Ali Kuwajerwala, Sacha Morin, Krishna Murthy Jatavallabhula, Bipasha Sen, Aditya Agarwal, Corban Rivera, William Paul, Kirsty Ellis, Rama Chellappa, et al. Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5021–5028. IEEE, 2024. 1, 2

[11] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023. 3, 4

[12] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12697–12705, 2019. 2

[13] Yang Miao, Francis Engelmann, Olga Vysotska, Federico Tombari, Marc Pollefeys, and Dániel Béla Baráth. Scenegraphloc: Cross-modal coarse visual localization on 3d scene graphs. In *European Conference on Computer Vision*, pages 127–150. Springer, 2024. 1, 7

[14] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 3

[15] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017. 2

[16] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 918–927, 2018. 2

[17] Zheng Qin, Hao Yu, Changjian Wang, Yulan Guo, Yuxing Peng, Slobodan Ilic, Dewen Hu, and Kai Xu. Geotransformer: Fast and robust point cloud registration with geometric transformer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(8):9806–9821, 2023. 8

[18] Antoni Rosinol, Arjun Gupta, Marcus Abate, Jingnan Shi, and Luca Carlone. 3d dynamic scene graphs: Actionable spatial perception with places, objects, and humans. In *Robotics: Science and Systems (RSS)*, 2020. 1, 2

[19] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–4, 2011. 2

[20] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE international conference on robotics and automation*, pages 3212–3217. IEEE, 2009. 8

[21] 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*, pages 21927–21937, 2023. 1, 4, 6, 8

[22] Sayan Deb Sarkar, Ondrej Miksik, Marc Pollefeys, Daniel Barath, and Iro Armeni. Crossover: 3d scene cross-modal alignment. *arXiv preprint arXiv:2502.15011*, 2025. 1

[23] Paul-Edouard Sarlin, Mihai Dusmanu, Johannes L Schönberger, Pablo Speciale, Lukas Gruber, Viktor

Larsson, Ondrej Miksik, and Marc Pollefeys. Lamar: Benchmarking localization and mapping for augmented reality. In *European Conference on Computer Vision*, pages 686–704. Springer, 2022. 1, 5

[24] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS)*, 42(3):1–21, 2017. 3

[25] Jared Strader, Nathan Hughes, William Chen, Alberto Speranzon, and Luca Carlone. Indoor and outdoor 3d scene graph generation via language-enabled spatial ontologies. *IEEE Robotics and Automation Letters*, 9(2):1232–1239, 2024. 2

[26] Ayça Takmaz, Elisabetta Fedele, Robert W Sumner, Marc Pollefeys, Federico Tombari, and Francis Engelmann. Openmask3d: Open-vocabulary 3d instance segmentation. *arXiv preprint arXiv:2306.13631*, 2023. 3

[27] Bin Wang and C-C Jay Kuo. Sbert-wk: A sentence embedding method by dissecting bert-based word models. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2146–2157, 2020. 5

[28] Abdelrhman Werby, Chenguang Huang, Martin Büchner, Abhinav Valada, and Wolfram Burgard. Hierarchical open-vocabulary 3d scene graphs for language-grounded robot navigation. In *First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024*, 2024. 1

[29] Hongyan Zhi, Peihao Chen, Junyan Li, Shuailei Ma, Xinyu Sun, Tianhang Xiang, Yinjie Lei, Mingkui Tan, and Chuang Gan. Lscenellm: Enhancing large 3d scene understanding using adaptive visual preferences. *arXiv preprint arXiv:2412.01292*, 2024. 1

[30] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018. 8