SGAligner: 3D Scene Alignment with Scene Graphs

Sayan Deb Sarkar [1], Ondrej Miksik [2], Marc Pollefeys [1,2], Daniel Barath [1], Iro Armeni [1]

[1]Department of Computer Science, ETH Zurich, Switzerland
[2]Microsoft Mixed Reality & AI Lab, Zurich, Switzerland

sgaligner.github.io

Abstract

*Building 3D scene graphs has recently emerged as a topic in scene representation for several embodied AI applications to represent the world in a structured and rich manner. With their increased use in solving downstream tasks (*e.g.*, navigation and room rearrangement), can we leverage and recycle them for creating 3D maps of environments, a pivotal step in agent operation? We focus on the fundamental problem of aligning pairs of 3D scene graphs whose overlap can range from zero to partial and can contain arbitrary changes. We propose* ***SGAligner****, the first method for aligning pairs of 3D scene graphs that is robust to in-the-wild scenarios (*i.e.*, unknown overlap – if any – and changes in the environment). We get inspired by multi-modality knowledge graphs and use contrastive learning to learn a joint, multi-modal embedding space. We evaluate on the 3RScan dataset and further showcase that our method can be used for estimating the transformation between pairs of 3D scenes. Since benchmarks for these tasks are missing, we create them on this dataset. The code, benchmark, and trained models are available on the project website.*

1. Introduction

Generating accurate 3D maps of environments is a key focus in computer vision and robotics, being a fundamental component for agents and machines to operate within the scene, make decisions, and perform tasks. As such, these maps should be actionable, *i.e.*, containing information (such as objects, instances, their position, and relationship to other elements) that allows agents to perform an action, and represented in an easily scalable, updateable, and shareable structure. Recently, 3D scene graphs [2, 41, 34, 18] have emerged as a topic in scene representation, providing a structured and rich way to represent the world. Not only do they fit the above requirements, but they can also be a lighter-weight [6] and more privacy-aware representation of 3D scenes than the predominantly used 3D point clouds or voxel grids. Hence, 3D scene graphs can be easier and safer to share across agents and humans operating in the same scene [22, 47].

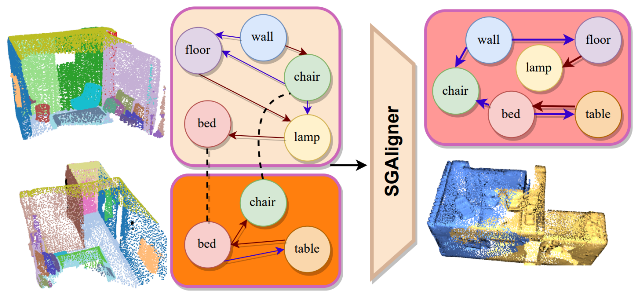

Figure 1. **SGAligner.** We address the problem of aligning 3D scene graphs of environments using multi-modal learning and leverage the output for the downstream task of 3D point cloud registration. Our approach operates directly on 3D scene graph level and it is fast and robust to real-world scenarios.

Given their potential, 3D scene graphs are increasingly used in embodied agents as a representation – commonly built on the fly – to perform robotic navigation [37, 35, 33, 22, 6] and task completion [12, 1, 32, 17, 20]. Since more and more agents are already building 3D scene graphs for downstream tasks, we investigate how to leverage and recycle them for creating 3D maps of the environment – a pivotal step in the agent operation – directly on the scene graph level. Specifically, we examine the fundamental problem of aligning partial 3D scene graphs of an environment that originate from different observations. We focus on real-world scenarios and specifically formulate the problem as follows: given two 3D scene graphs that represent 3D scenes with overlap that can range from zero to partial or full and can contain changes, our goal is to find an alignment across scene graph nodes, if it exists. Interestingly enough, even though entity alignment (*i.e.*, node alignment) is used in knowledge graphs and in linguistics [24, 13, 26, 46, 7, 9, 38, 23], the task of aligning 3D scene graphs of environments has not been explored. An important note is that entity alignment in these domains assumes that there is overlap between graphs and that all inputs contain true (*i.e.*, correct) information.

We propose **SGAligner**, the *first* method for aligning pairs of 3D scene graphs that is robust to in-the-wild scenarios (*i.e*., unknown overlap – if any – and changes in the environment) (see Figure 1). We get inspired by entity alignment methods in multi-modality knowledge graphs [23] and redesign them for our setting. 3D scene graphs represent three main types of information [41, 2, 34]: semantic entities in the scene (*e.g*., object instances) with their 3D geometry (*e.g*., 3D point cloud), their attributes (*e.g*., category, size, and material), and relationships between the entities (*e.g*., relative position and attribute similarity). The main premise is to independently encode each of these modalities with the ultimate objective of learning a joint embedding that can reason about the similarity of two entities (nodes). Given entities matched using a cosine similarity metric, we perform scene graph alignment using the ones with the highest similarity.

We additionally demonstrate our scene graph alignment method on the task of 3D point cloud registration, as well as on the downstream tasks of 3D point cloud mosaicking, finding overlapping 3D scenes, and 3D alignment of a local 3D point cloud on a prior map that may contain changes. Instead of directly computing 3D correspondences on the entire point clouds [14, 30, 44, 3], we use the scene graph alignment as coarse initialization for the registration. We further refine it by computing 3D correspondences [30] on the individual point clouds (*i.e*., object instance point clouds) of each matched entity pair. This is followed by robustly estimating [11] the rigid point cloud transformation using the computed correspondences of all matched pairs.

We evaluate all tasks on the 3RScan [40] dataset, which contains 3D point clouds of indoor scenes captured over time, along with their 3D scene graph annotations [41]. Since 3RScan does not provide partial scene graphs of a scene (captured at the same temporal point) or a benchmark for 3D scene graph alignment or any of the other downstream tasks, we create the data, metrics, and evaluation needed to benchmark all tasks on 3RScan. Our experiments show that our approach reduces the relative translation error of state-of-the-art GeoTransformer [30] by 40% on point cloud registration, while being $3\times$ faster during the overlap check (*i.e*., identifying whether the point clouds to register spatially overlap or not), since it does not need to process the entire point clouds. Detailed ablation studies, along with experiments on the tasks of point cloud mosaicking and of aligning a changed local 3D scene to a prior 3D map, demonstrate the robustness of our approach.

We summarize the contributions of this paper as follows:

- We propose **SGAligner**, the first method for aligning pairs of 3D scene graphs whose overlap can range from zero to partial and that may contain changes.
- We demonstrate the potential of our method on the tasks of 3D point cloud registration, 3D point cloud mosaicking and 3D alignment of a point cloud in a larger map that contains changes.
- We create benchmarks for scene graph alignment and 3D point cloud registration, as well as on the other downstream tasks, on the 3RScan [40, 41] dataset, with data, metrics, and evaluation procedure.

2. Related Work

Multi-Modal Knowledge Graph Alignment. There exists vast literature in the domain of graph matching, with methods focusing on creating single feature vectors to compare on the entire graph-level (*e.g*., [21]). Such methods are not applicable to our case, since the two graphs to align have partial semantic and geometric overlap. A more relevant task is that of multi-modal knowledge graph (KG) alignment [24, 13, 26, 46, 7, 9, 8, 38, 23], which refers to the task of aligning multiple knowledge graphs that represent information from different modalities (*e.g*., text, images, and videos). The goal is to integrate the knowledge from different sources and provide a more comprehensive understanding of the world. EVA [24] leverages visual and other auxiliary knowledge to achieve entity alignment in both supervised and unsupervised settings using a loss formulation inspired by NCA-based text-image matching [25]. More recently, approaches like [7, 23] solve the task by learning a common embedding space for all modalities, where similar entities in the KG have similar embeddings. Both approaches use multiple individual encoders to obtain modality-specific representations for each entity in the KG. However, [23] introduces contrastive learning with intra-modal contrastive loss and inter-modal alignment loss to learn discriminative cross-modal embeddings while [7] only performs common space learning to align the embeddings. Since 3D scene graphs can be also considered as containing multiple modalities (*i.e*., object instances, their 3D geometry, attributes, and in-between relationships), we leverage the above architecture and adapt it for the setting of aligning 3D scene graphs of environments. A point to highlight is that KG alignment methods consider inputs as (partially) overlapping and true, something that in the real-world scenario of creating 3D maps does not hold due to noise.

3D Scene Graphs. Following the success of utilizing 2D scene graphs [19], researchers introduced 3D scene graphs as structured and rich representations to describe real-world scenes [2, 18, 34, 41]. We follow the 3D scene graph structure and dataset presented in [41]. Existing work addresses the task of generating 3D scene graphs with a variety of approaches; from online incremental building [42, 15] to offline based on RGBD images and/or 3D reconstructions [2, 41, 35]. The 3D scene graph representation has been explored in embodied AI for tasks related to navigation and planning [37, 35, 33, 22, 6], task comple-

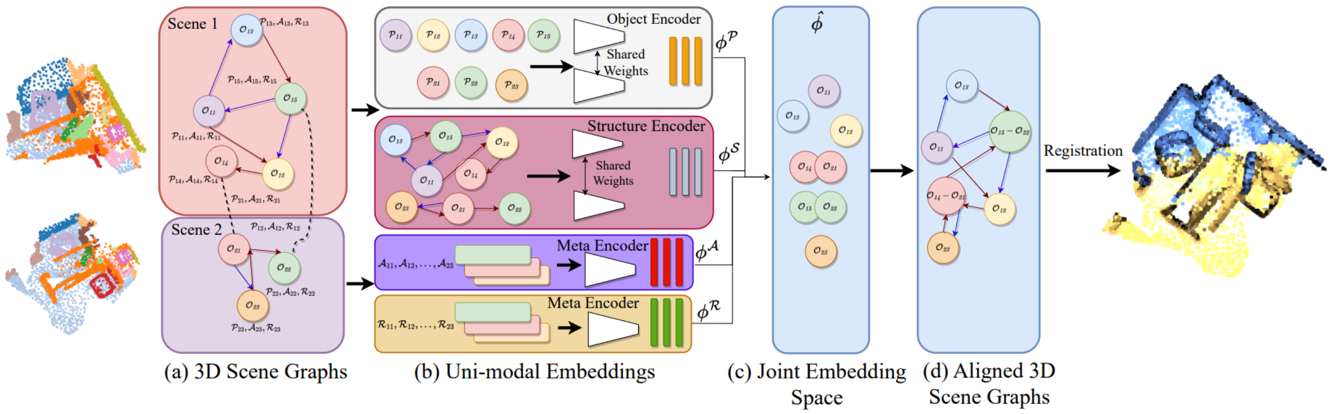

Figure 2. **Overview of SGAligner.** Given two 3D scene graphs with partial overlap (a), we create four uni-modal embeddings (b) that encode object, structure, attribute, and relationship information. These interact with each other to create a joint embedding space (c) where similar nodes are located closely. We use this similarity to match nodes and create the final aligned 3D scene graph (d). We further demonstrate the use of the alignment in 3D point cloud registration.

tion [12, 1, 32, 17, 20], variability estimation [27], and 3D scene manipulation [10]. Although prior work has demonstrated the applicability of 3D scene graphs in an increasing fashion, it has not been investigated for the task of creating 3D maps of scenes. Hydra [15] approaches the latter using hierarchical loop closure detection and 3D scene graph optimization to complete the non-optimised scene graph, that is built on the fly, into a globally consistent and persistent one as new information is captured. On the another hand, our contribution focuses on solving graph matching solely on the object-level. This would enable to leverage, share, and recycle the created graphs for general agent operation.

3D Point Cloud Registration. The field of 3D point cloud registration is well-established and active, with approaches mainly being feature-based and end-to-end. We focus our scope on feature-based approaches since they compute hard correspondences and perform more robustly in real-world scenes. Such methods [3, 14, 44, 30] consist of two steps: local (learned) feature extraction and pose estimation with RANSAC [11]. However, they focus on the in-vitro problem of aligning two input point clouds that have some degree of overlap. This does not always hold in a real-world scenario where there is no knowledge of whether there is any overlap or changes between the inputs. Although some of these methods compute matchability scores per estimated 3D point correspondences (*e.g.*, [14, 30]), they assume overlapping input point clouds and do not have the mechanism to discard non-overlapping ones. In addition, when the number of individual point clouds to register becomes large, it requires $O(\mathcal{N}^2)$ complexity to process all possible pairs, while failing to recognize non-aligned pairs. Last, they also require large memory reserve to process input data if the point clouds are large. Our method allows processing all possible pairs faster while identifying non-overlapping pairs before performing the final registration. In addition, it is lightweight and can easily process large scenes, since, as shown in our ablation studies, our method can operate with a very limited number of points per object instance.

3. SGAligner: 3D Scene Graph Alignment

Following standard formulation [41], we define a 3D scene graph $\mathcal{G}$ of a scene s as a pair of sets $(\mathcal{N}, \mathcal{R})$ with nodes $\mathcal{N}$ and edges $\mathcal{R}$. The nodes represent 3D object instances $\mathcal{O}$ in the scan. Each node also contains a set of attributes $\mathcal{A}$ that characterizes the visual (*e.g.*, style and texture), geometric (*e.g.*, shape and symmetry), and state (*e.g.*, closed and empty) characteristics of the object instance, in addition to the object categories. Instance-level point clouds $\mathcal{P}$ are available per node. The edges $\mathcal{R}$ define semantic relationships between the nodes such as `standing on` and `attached to`. Given two graphs $\mathcal{G}_1 = (\mathcal{N}_1, \mathcal{R}_1)$ and $\mathcal{G}_2 = (\mathcal{N}_2, \mathcal{R}_2)$ of scenes s_1 and s_2 respectively, we aim to find the set of objects in the overlapping regions of the two scenes, denoted as entity pairs, $\mathcal{F} = \{(n_1, n_2) \mid n_1 \equiv n_2, n_1 \in N_1, n_2 \in N_2\}$, forming node correspondences.

Our approach follows the network architecture proposed in [23], which we modify from the language domain for the task of 3D scene graph alignment with inputs of varying degree of overlap. The overall architecture is shown in Figure 2. We can formulate scene graphs as a multi-modal knowledge graph – commonly used in entity alignment – where the semantic and geometric information included in scene graphs is treated as the different modalities that are encountered in knowledge graphs. The goal is to learn a joint multi-modal embedding from individual encodings of each modality (uni-modal), in which nodes are closely located if they correspond to the same underlying object instance and belong to different graphs. Specifically, we create uni-modal embeddings using the three main 3D scene graph information types: *object* embedding that encodes

$\mathcal{P}$, *structure* embedding $\mathcal{S}$ that encodes $\mathcal{R}$ in the form of a structured graph, and two *meta* modalities that encode $\mathcal{A}$ and $\mathcal{R}$ between objects in the form of a one-hot vector. To reason about the final entity alignment, similar to [23], we create a joint embedding by combining these uni-modal ones in a weighted manner and perform a joint optimization using knowledge distillation across all embeddings.

3.1. Uni- and Multi-Modal Embeddings

To leverage rich information in 3D scene graphs, we process each modality separately in our framework and create uni-modal embeddings, which are later processed to model complex inter-modal interactions in the joint embedding.

Object Embedding. Point clouds contain rich geometric information about objects. Each of the individual point cloud $\mathcal{P}_i$ of $\mathcal{O}_i$ is the input to the object encoder. We employ the PointNet architecture [29] as the object encoder to extract the visual features $\phi_i^{\mathcal{P}}$ for every node.

Structure Embedding. 3D Scene Graphs contain information on relationships between $\mathcal{O}$, which we leverage to encode the layout of objects in s. We represent this information in the form of a *structure* graph: node features are the relative translation between object instances, and edges are the aforementioned relationships. We calculate relative translation by taking the distance between the object instance consisting of the highest number of relationships and that of any other object instance in the scene. Specifically, we compute distances using the barycenter of the convex hull of the point clouds. We use a Graph Attention Network (GAT) [**?**] to model the structural information in $\mathcal{G}_1$ and $\mathcal{G}_2$ via the structure graph.

We limit the weight matrix to a diagonal matrix, as suggested by [23], to minimize computations and improve the scalability of the model. As per [23], the neighborhood structure embedding $\phi_i^{\mathcal{S}}$ is produced by the last GAT layer, using a two-layer GAT model to aggregate the neighborhood information over several hops.

Meta Embeddings. Along with modeling the geometric and structural properties of the scene, we model the attributes and corresponding relationships per object $\mathcal{O}_i$ in two separate embeddings. We regard the relationship of $\mathcal{O}_i$ with other objects as a bag-of-words feature vector and use it as input to a feed-forward neural network layer to obtain the relational embedding $\phi_i^{\mathcal{R}}$. We adopt the same approach for the attributes of $\mathcal{O}_i$ for simplicity to get the $\phi_i^{\mathcal{A}}$. We call these single-layer networks meta encoders because they are easily interchangeable and extendable to new data.

Joint Embedding. We concatenate each of the previously discussed uni-modal features to a single compact representation $\hat{\phi}_i$ for each object $\mathcal{O}_i$ as follows:

$$\hat{\phi}_i = \oplus_{k \in \mathcal{K}} \left[\frac{\exp(w_k)}{\sum_{j \in \mathcal{K}} \exp(w_j)} h_i^m \right], \tag{1}$$

where $\oplus$ denotes concatenation, $\mathcal{K} = \{\mathcal{P}, \mathcal{S}, \mathcal{R}, \mathcal{A}\}$, and w_m is a trainable attention weight for each modality k. We apply L_2 normalization to each uni-modal feature before the final concatenation. These embeddings are coarse-grained without any interaction between different modalities.

3.2. Contrastive Learning

To model interaction between modalities, we formulate a representation learning framework. A contrastive loss function encourages comparable samples, or aligned entities, to be closer together and dissimilar samples to be farther away in the learned representation space. Using a cross-modal contrastive loss is a typical strategy when working with many modalities, such as in the case of 3D scene graphs, as it encourages samples from various modalities that are semantically related to be closer together in the joint representation space. Inspired by [23], we use the Intra-Modal Contrastive Loss (ICL) and Inter-modal Alignment Loss (IAL) and formulate them similarly.

During training, we assume that $E \subset \mathcal{F}$ is available to us as seed-aligned entity pairs. Formally, for the i^{th} object node $n_1 \in \mathcal{N}_1$, we define $E = \{n_1^i \mid n_2^i \in \mathcal{N}_2\}$, where (n_1^i, n_2^i) is an aligned pair. We define the unaligned pairs within the same graph as $H_1^i = \{n_1^j \mid \forall n_1^j \in \mathcal{N}_1, j \neq i\}$, and aligned pairs across graphs as $H_2^i = \{n_2^j \mid \forall n_2^j \in \mathcal{N}_2, j \neq i\}$ (Figure 3). These two samples define the constrained joint embedding space. We model $\mathcal{L}_k^{ICL}$ to learn intra-modal dynamics for more discriminative boundaries for each modality k in the embedding space. We apply ICL separately on each uni-modal embedding and on the joint concatenated embedding, after L_2 normalization. Each uni-modal embedding is trained individually using ICL and is not intended to interact with others.

Our complete representation is the joint embedding space, and our goal is to learn proper uni-modal encodings that enable node alignment in this joint space. To achieve this, we minimize the bi-directional KL-divergence loss between joint embedding and uni-modal embeddings as the Inter-modal Alignment Loss (IAL). Thereby, we emphasize on aggregating the distribution of various modalities, which narrows the modality gap by learning interactions between various modalities inside each entity. We train our model end-to-end, optimizing both losses as follows:

$$\mathcal{L} = \mathcal{L}_o^{ICL} + \sum_{k \in \mathcal{K}} \alpha_k \mathcal{L}_k^{ICL} + \sum_{k \in \mathcal{K}} \beta_k \mathcal{L}_k^{IAL}, \tag{2}$$

where $\mathcal{K} = \{\mathcal{P}, \mathcal{S}, \mathcal{R}, \mathcal{A}\}$ and o is the joint embedding. Variables α_k and β_k are hyper-parameters that are automatically learned during training. We direct the readers to [23] for a deeper understanding of the loss functions.

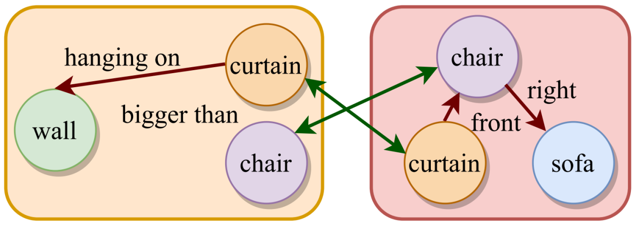

Figure 3. **An example of contrastive learning pairs.** Aligned pairs across graphs are marked with green edges and unaligned pairs within the same graph are marked with brown edges.

3.3. 3D Point Cloud Registration

In this section, we describe how we approach the task of 3D point cloud registration by leveraging our scene graph alignment results. The output of the previously described scene alignment method is the set of matched entity pairs n_1 and n_2, in the scene graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively. For each entity pair n_1^i and n_2^i, we extract 3D point correspondences from $\mathcal{P}_1^i$ and $\mathcal{P}_2^i$. The correspondences are estimated by an off-the-shelf correspondence extraction algorithm (*e.g.*, [30]) by running it on node pairs independently. We collect the point correspondences across all matched entities and then use the robust estimator, *e.g.* RANSAC [11] or one of its recent variants [31, 4, 5], to get the transformation $\mathbf{T} \in \mathrm{SE}(3)$ between the point clouds of the two scenes.

Performing registration on 3D correspondences that stem from node-to-node matches allows for being less sensitive to changes in the point clouds and incorrect matching than state-of-the-art techniques.

Such an approach has two major advantages: (i) it filters non-overlapping scene pairs, which should not be registered, **faster** than state-of-the-art point cloud registration methods and without any need for registration. This is enabled by obtaining object-level (node-to-node) correspondences instead of performing the registration directly on a large-scale 3D point cloud. (ii) It performs better than standard registration methods even on point clouds with low overlap, which we showcase with experiments in the following section.

4. Experiments

We evaluate **SGAligner** on the task of 3D scene graph alignment (Section 4.1) and on downstream applications, namely 3D point cloud registration (Section 4.2), 3D point cloud mosaicking (Section 4.3), finding overlapping 3D scenes (Section 4.4), and 3D scene alignment with changes present in the data (Section 4.5). In our experiments we use the 3RScan dataset [40], which consists of 1335 indoor scenes, of which 1178 are used for training and 157 for validation[1]. The dataset contains semantically annotated 3D point clouds per scene, some of which depict the same environment over time. 3D scene graph annotations for 3R Scan are provided in [41]. Although this allows to evaluate the robustness of our method in the case of changed environments, there are no annotations to evaluate in static environments. To enable a thorough evaluation with scenes that range in overlap (from zero to partial), we create sub-scene graphs in single scenes from one temporal point, given their full 3D scene graph provided in the annotations, following standard point cloud registration benchmarks [45]. We make our code and benchmark publicly available.

[1]3D scene graph annotations are not publicly available for the test set, hence we use the validation set of 3RScan as the test set in our experiments.

Dataset Generation. To create the sub-scene graphs, we first generate sub-scenes per scene on the geometry level (*i.e.*, partial point clouds). Scene point clouds in [40] are generated from RGBD frames. To imitate a realistic setting, we create sub-scenes by selecting groups of sequential frames and reconstructing the depicted scene's point cloud. Frames across groups are unique and there is a possibility of 3D spatial overlap in the generated point clouds. We show examples of sub-scenes generated using this approach in Figure 5, alongside the camera trajectory used to capture the corresponding scan in [40]. We generate a total of 6728 sub-scenes from the training scenes and 827 sub-scenes from the validation scenes. We create pairs using the sub-scenes generated from the same scene, such that the spatial overlap in a pair ranges from 10-90%. This results to 11752 pairs for training and 1452 pairs for testing. In Figure 4, we report statistics on the spatial overlap of the generated pairs. For simplicity, hereafter we refer to sub-scenes as *scenes* (s in Section 3).

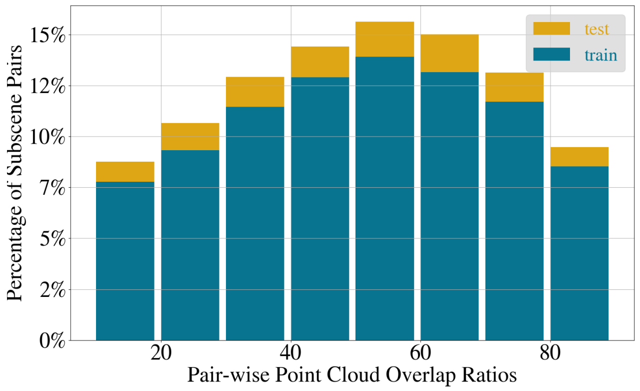

Figure 4. **Overlap statistics for the generated sub-scenes.** We show the percentages of sub-scene pairs per overlap range for both training and test set w.r.t the entire generated dataset.

The individual point clouds of object instances in the generated data will have a varying number of points. To use them as input to PointNet (Section 3.1), we require them to have the same size. We use a size of 512 for training and showcase in the ablation studies that our method can provide similarly robust performance in lower resolutions

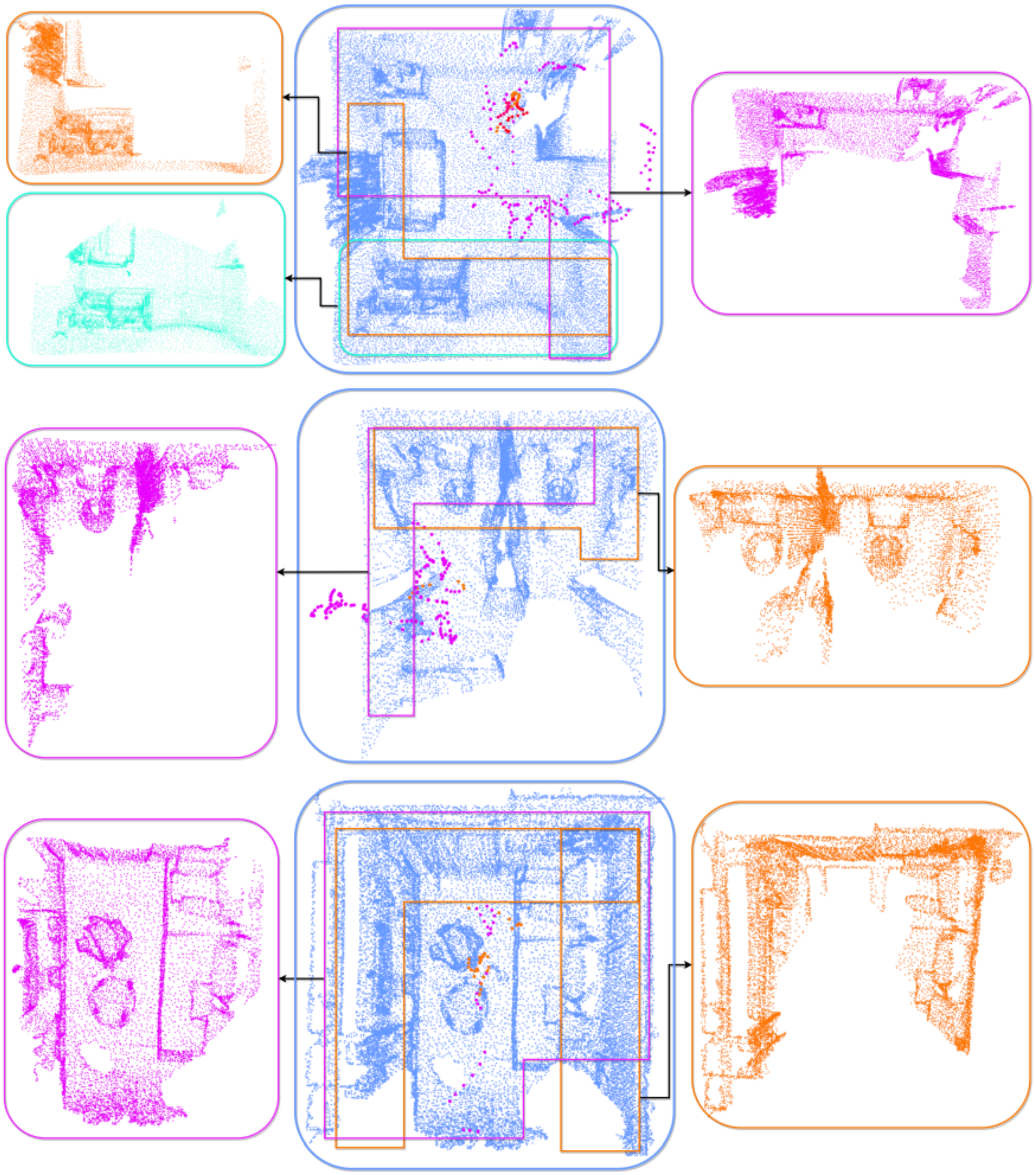

Figure 5. **Visualization of data generation process.** We visualise the creation of sub-scenes given a larger single scene (parent) for which RGB-D frames and camera poses are available. Given a point cloud from [40] (middle column), we create multiple sub-scenes (left and right columns) and showcase the used camera trajectory. Different colors depict the camera trajectory and 3D spatial area of each sub-scene in the parent scene.

(*e.g.*, 64). We use farthest point sampling to downsample each $\mathcal{P}_i$ of object $\mathcal{O}_i$ as

$$\mathcal{P}_i = \{\delta_{k^i} \odot p_k\}_{k=1,|P|}, \quad (3)$$

where δ represents the Kronecker delta, p is a point in $\mathcal{P}$, and $|.|$ is the cardinality of $\mathcal{P}_\rangle$, i.e, the number of points.

4.1. 3D Scene Graph Alignment

In this section, we evaluate **SGAligner** on the 3D scene graph alignment task. We utilize cosine similarity on the joint embedding space to calculate the similarity between two matched entities (nodes) and employ the standard metrics Mean Reciprocal Rank (MRR) and Hits@K, where

K=$\{1, 2, \ldots, 5\}$. MRR denotes the mean reciprocal rank of correct matches. Hits@K denotes the ratio of correct matches appearing within top K matches, based on their cosine similarity ranking. A formal definition of these and all other metrics used is provided in Appendix A.

We compare **SGAligner** to using different modalities, and as a result embeddings, as well as with a baseline from entity alignment in the language domain. For the latter purpose, we adapt the Entity Visual Alignment method (EVA) [24] for 3D scene graphs by replacing the visual encoder with PointNet architecture [29], same as $\mathcal{P}$ in our approach. It is important to note that none of the approaches use iterative learning, instead, 30% of the anchor pairs are used during training following standard formulation. Please note that when employing only the instance level point clouds $\mathcal{P}$ there is no IAL used.

Node Alignment. We first evaluate our method on node alignment, for which results are in Table 1. As evident, our method, even when using a single modality, outperforms EVA [24] with a margin of approximately 2%. Furthermore, and as expected, each modality in our method contributes to improved performance in all metrics. Interestingly, using all modalities provides at K=2 better results than only $\mathcal{P}$ at K=5, and at K=3 it is already better than any of the other combinations at K=5. To further verify the robustness of our method and hence its suitability for real-world applications, we also compare the performance of our method using both ground truth and predicted 3D scene graphs as input during inference (in both cases the network has been trained on ground truth data). We compute the scene graph predictions using the pre-trained network for 3D scene graph generation given 3D point cloud of [41][2]. As shown, although as expected the performance drops, **SGAligner** is still able to provide reasonable matches despite the noise in the input. The success and failure case presented in Figure 6, shows that the existence of the same exact object in multiple replicas creates erroneous matches; when the objects are more discriminative, **SGAligner** provides accurate results.

We further ablate the results of our method (on ground truth data) for scenes of different overlap to evaluate the robustness in cases where few nodes can be matched. Results are reported in Table 2. As expected, the performance drops with lower overlap, however, the gap between the very high and very low overlap is not drastic.

In order to deeper understand the effect of semantic noise to **SGAligner**'s performance, we provide experiments with controlled semantic noise on the test data. Specifically, we consider the following scenarios of noise: (i) only relationships are removed[3]; (ii) only object instances are removed – their corresponding attributes and any relationships that include them are also removed; (iii) both relationships and object instances are removed; (iv) object instances are wrong (*i.e.* they have the wrong semantic label); and (v) both relationships and objects are wrong. Noise is applied to each input scene graph randomly to 15-40% of the modified semantic. For (iv) and (v), we randomly assign any other semantic label, *i.e.*, a chair could be labeled as floor, or relationship, *e.g.*, `standing on` instead of `close by`. Results, including those on the full ground truth dataset (GT) and with predicted 3D scene graphs (Pred.) for reference, are in Table 3. Note that the training set remains the same. As shown, the noise that our method can handle the best is that of missing objects (ii). This means that the relationships between objects can be more important than having information on all objects in the scene, however not by a large margin. What drops the performance drastically is wrong semantic labels for objects and relationships. This means that missing information is less harmful than wrong information in the 3D scene graph, regardless of whether it is about objects or relationships. Hence, it can be useful for autonomous agents building maps to have an understanding of uncertainty in the information. Comparing the results for scenarios (iii) and (iv) with the use of predicted 3D scene

Method	Mean RR ↑	Hits @ ↑ K = 1	K = 2	K = 3	K = 4	K = 5
w/ Ground Truth 3D Scene Graphs						
EVA [24]	0.867	0.790	0.884	0.938	0.963	0.977
$\mathcal{P}$	0.884	0.835	0.886	0.921	0.938	0.951
$\mathcal{P} + \mathcal{S}$	0.897	0.852	0.899	0.931	0.945	0.955
$\mathcal{P} + \mathcal{S} + \mathcal{R}$	0.911	0.861	0.916	0.947	0.961	0.970
SGAligner	**0.950**	**0.923**	**0.957**	**0.974**	**0.9823**	**0.987**
w/ Predicted 3D Scene Graphs						
SGAligner	0.882	0.833	0.881	0.918	0.937	0.951

Table 1. **Evaluation on node matching.** We compare the performance of **SGAligner** for different modality combinations, as well as for ground truth and predicted scene graphs.

Overlap (%)	Mean RR ↑	Hits @ ↑ K = 1	K = 2	K = 3	K = 4	K = 5
10-30	0.872	0.806	0.886	0.927	0.948	0.962
30-40	0.908	0.859	0.917	0.949	0.968	0.978
40-50	0.941	0.908	0.950	0.973	0.980	0.985
50-60	0.960	0.937	0.967	0.982	0.989	0.994
60-	**0.979**	**0.966**	**0.982**	**0.990**	**0.994**	**0.995**
All data	0.950	0.923	0.957	0.974	0.982	0.987

Table 2. **Evaluation on node matching per overlap range.** Even when the spatial overlap between inputs is low, **SGAligner** is able to produce meaningful results. As a note, the average number of overlapping nodes for the 10-30% overlap is 9.3, *i.e.*, K=5 does not exhaustively cover all overlapping nodes.

[2]We refer the reader to Table 2 in [41] for an evaluation on scene graph prediction of this method.

[3]Ground truth annotations do not offer an exhaustive list of relationships and attributes per node. We remove edges from these annotations.

graphs, we observe that for the former the values are significantly lower. This has to be taken into account in case of real-world applications, especially if generalization of the scene graph prediction algorithm is unknown.

	Affected Modules	Mean RR ↑	Hits @ ↑ K = 1	K = 2	K = 3	K = 4	K = 5
(i)	$\mathcal{S}, \mathcal{R}$	**0.906**	**0.858**	0.915	**0.949**	**0.964**	0.974
(ii)	All	**0.924**	**0.878**	**0.942**	**0.968**	**0.977**	**0.985**
(iii)	All	0.902	0.848	**0.918**	**0.949**	**0.964**	**0.975**
(iv)	$\mathcal{P}$	0.661	0.563	0.643	0.699	0.743	0.776
(v)	$\mathcal{P}, \mathcal{S}, \mathcal{R}$	0.587	0.479	0.573	0.632	0.674	0.709
GT	None	0.950	0.923	0.957	0.974	0.982	0.987
Pred.	None	0.882	0.833	0.881	0.918	0.937	0.951

Table 3. **Evaluation on node matching with controlled semantic noise.** *Best* values are in **underlined bold**. *Second best* in **bold**.

3D Scene Graph Alignment. Entity alignment provides pairs of matched nodes. Here, we evaluate how well two scene graphs can be aligned given the predicted node matching. In theory, since the scenes are rigid, two pairs of matched nodes would be enough to perform the alignment, if there was no noise in the matches. Since in reality matches are noisy, we evaluate 3D scene graph alignment with top-2, top-50%, and all of the matched nodes. We choose top matched nodes based on an increasing ranking of their similarity distances. To measure this performance we introduce the *Scene Graph Alignment Recall (SGAR)* metric. Similar to the standard recall metric, we calculate the amount of correct alignments of **SGAligner**. We provide these results in Table 4 for both ground truth and predicted scene graphs. As shown, results with top-2 perform the best in both cases. This means that **SGAligner** can identify at least two matches that are very closely located in the joint embedding space. In addition, SGAR for the top-2 matches is approximately the same for both ground truth and predicted scene graphs, which further shows that our approach can robustly align them.

Input Scene Graphs	Scene Graph Alignment Recall ↑ R@top-2	R@top-50%	R@All
Ground Truth	**0.964**	0.948	0.738
Predicted	**0.963**	0.856	0.450

Table 4. **Evaluation on 3D scene graph alignment.** Top-2 matches are the most robust for correct alignment, since adding matches of decreasing similarity introduce more noise than new information in calculating the alignment. We report for both ground truth and predicted scene graphs.

4.2. 3D Point Cloud Registration

In this section, we evaluate the performance of **SGAligner** for 3D point cloud registration on the same data. We employ the state-of-the-art Geotransformer [30] as a 3D correspondence extraction method, as per Section 3.3. We compare the performance of our approach to standard approaches and use Geotransformer directly on the point clouds of the two scenes to register. Please note that in our case, we use Geotransformer to extract correspondences from individual point clouds on the level of object instances only for the matched nodes. In all cases, we use the Geotransformer model trained on the 3DMatch dataset [45].

We compute the standard metrics of Chamfer distance (CD) as in [43], relative rotation error (RRE), relative translation error (RTE), feature match recall (FMR), and registration recall (RR). RR is calculated with the standard threshold of RMSE $= 0.2$.

Results for 3D point cloud registration for *overlapping scenes* are shown in Table 5. Our method is consistently providing best results in all metrics w.r.t. the standard registration approach, despite the less geometric information in the point clouds we use; specifically on CD, we achieve a 49.4% improvement (K=2). This is an expected behavior since our alignment method, even when it contains incorrect matches, is still providing an initialization to the task and narrows down the search space for correspondences. With respect to self baselines, our method with K=2 performs the best. The drop in K=3 can be attributed to the following: more matches per each node of which one is correct, means that more outliers are provided to RANSAC, which it cannot easily handle. Since the gap of Hits@K between K=2 and K=1 is larger than that between K=3 and K=2, K=2 can still provide a boost of inliers to RANSAC even though it will also increase outliers with respect to K=1. We include examples in Figures 6 and 7.

	Methods	CD ↓	RRE (°) ↓	RTE (cm) ↓	FMR (%) ↑	RR(%) ↑
	GeoTr	0.02247	1.813	2.79	**98.94**	98.49
Ours	K=1	0.01677	**1.425**	2.88	**99.85**	98.79
Ours	K=2	**0.01111**	**1.012**	**1.67**	**99.85**	**99.40**
Ours	K=3	**0.01525**	1.736	**2.55**	**99.85**	**98.81**

Table 5. **3D Point Cloud Registration. SGAligner** with K=2 provides the best results on this task, followed by K=3. GeoTr achieves consistently worse results than any of the **SGAligner** versions. *Best* values are **underlined bold**. *Second best* in **bold**.

We further ablate the results on this task for scenes of different overlap. As shown in Table 6, our approach outperforms the standard one per overlap, and even provides better results for scenes with 30% overlap or higher than the standard approach can do on 60% or higher.

Overlapping vs Non-Overlapping Scenes. In practical and real-world applications, we do not always know if two scenes are overlapping or not. While standard point cloud registration approaches compute a matchability score, they typically train and test only on *overlapping* scenes and do not have the mechanism to discard non-overlapping ones.

Here, we demonstrate how our approach can indirectly

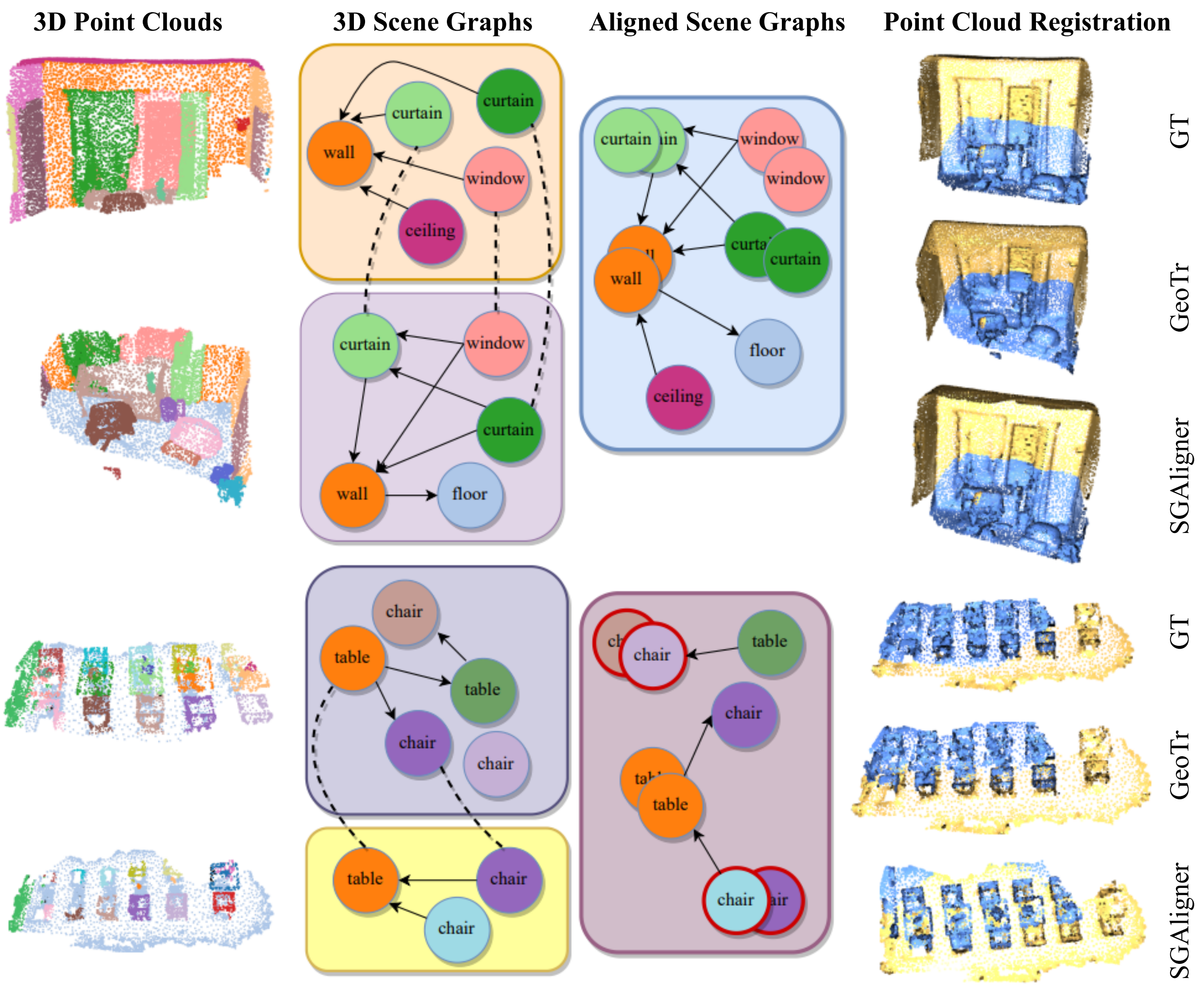

Figure 6. **Qualitative Results.** Given input 3D scene graphs (left), we showcase a success (top) and a failure (bottom) case of **SGAligner** for alignment (middle) and registration (right). The existence of the same exact object in multiple replicas creates erroneous matches (nodes outlined in red) which cannot be recovered in registration. When objects are more discriminative, **SGAligner** provides accurate matching.

	Overlap (%)	**CD** ↓	**RRE** ↓ (°)	**RTE** (cm) ↓	**FMR** (%) ↑	**RR** (%) ↑
GeoTr.	10-30	0.09788	8.830	13.56	94.57	92.25
	30-60	0.00584	0.156	0.24	97.28	97.36
	60-	0.00177	0.048	0.07	99.47	99.31
Ours	10-30	0.05160	5.660	8.48	99.23	95.35
	30-60	**0.00127**	**0.045**	**0.05**	**99.68**	**98.34**
	60-	<u>**0.00046**</u>	<u>**0.018**</u>	<u>**0.02**</u>	<u>**99.92**</u>	<u>**99.93**</u>

Table 6. **3D Point Cloud Registration per overlap.** For our method we use the best performing (K=2). **SGAligner** for overlap $\geq 30\%$ is performing better than GeoTr for $\geq 60\%$. *Best* values are <u>**underlined bold**</u>. *Second best* in **bold**.

provide a solution to this problem, without any additional supervision. We formulate it as to how well a method can identify whether a pair of scenes overlaps. For the state-of-the-art Geotransformer, we compute a scene-level *matchability* μ by averaging over all correspondence matchability scores and consider two scenes as overlapping if $\mu \geq 0.2$. For our approach, we compute a scene-level *alignment* score ξ by averaging the total number of matched nodes (for K=1) and consider two scenes as overlapping if $\xi \geq 0.2$.

To evaluate this task, we create a new test set that includes overlapping and non-overlapping scenes from the data we generate. Specifically, we use all 1452 of the overlapping pairs in the test set we previously created, and sample 1452 non-overlapping pairs from the rest of the sub-scenes created in the data generation process. Please note that these pairs can be from same sub-scene or different sub-scene, captured at same or different temporal point. The training set remains the same and contains *only* overlapping pairs. We compute the precision, recall, F1-score, and time in milliseconds required to make the overlap decision for all pairs. Results are in Table 7. Our approach runs 3 times faster than Geotransfomer since it does not process the en-

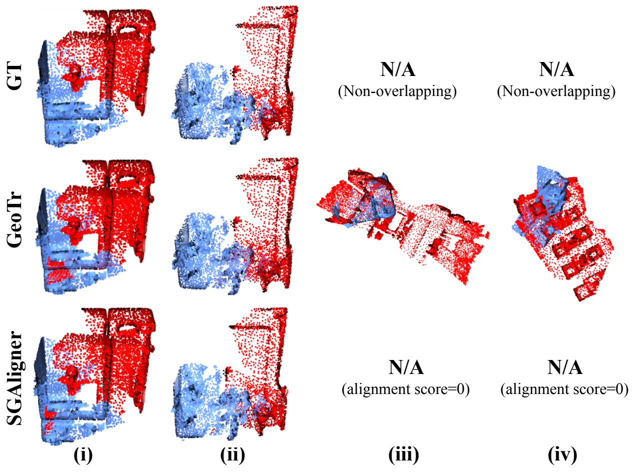

Figure 7. **Registration Results for SGAligner and [30]**. Columns (i)-(ii) represent registration on overlapping pairs. (iii)-(iv) show incorrect results of [30] for **non-overlapping** pairs, where our *alignment score* is zero (*i.e.*, there is no registration output to report for **SGAligner**). Since these pairs are not overlapping, there is also no ground truth registration.

tire point cloud, which can be computationally demanding. In addition, it leads to comparative performance while being able to identify more overlapping pairs correctly.

We choose μ and ξ based on the best performing value of μ for Geotransformer. Specifically, we observed that a higher value (0.4) leads to no true positives and a lower one (0.1) leads to lower precision (66.94%) since it identifies more false positives. This is understandable given that it was not trained on this data. However, (0.2) for a metric measuring average point correspondence similarity is low. For ξ we observe only a slight decrease in precision when using 0.2 instead of the best performing value (0.4). In contrast, our approach does not require such difficult-to-set thresholding scheme.

Method	**Prec.** (%) ↑	**Recall** (%) ↑	**F1** (%) ↑	**Average Time Per Scene** (ms) ↓
GeoTr [30]	**99.63**	80.98	89.34	442.50
Ours	92.03	**90.94**	**91.48**	**139.64**

Table 7. **Overlap Check for Point Cloud Registration.** Our method performs comparably or better than GeoTr, while being about ×3 faster to perform the check.

4.3. 3D Point Cloud Mosaicking

In this section, we aim at reconstructing the 3D scene from a set of partial point clouds observing parts of the scene. We proceed by selecting one of the point clouds as the origin and then estimating the absolute pose (*i.e.*, 3D translation and rotation) between the origin and each remaining point cloud in the set. One can imagine this procedure as 3D point cloud mosaicking. Please note that there is no guarantee that all partial clouds overlap with the one chosen as origin. While this could be alleviated by forming all possible pairs and forcing global consistency, solving the 3D mosaicking problem falls outside the scope of this paper. We only aim to demonstrate the potential of the proposed algorithm for this problem.

We perform the pairwise registration for all pairs using the method described in Section 3.3. In Table 8, we report results and compare with Geotransformer [30], using the same reconstruction paradigm. The evaluation metrics we use focus on the geometric aspects of accuracy and completeness [39] of the resulting reconstruction, as well as on precision, recall, and F1-score of registered point clouds. **SGAligner** has higher performance on 3 out of 5 metrics, and is mainly affected in completion and precision. The performance drop is due to the fact that for some scenes with low overlap, **SGAligner** fails to perform node alignment and, hence, registration with the incorrect alignments fails. In these scenes, GeoTr has a better complete context of the entire scene unlike the only object-based context in our approach. Furthermore, it is interesting to note that our K=1 method performs better than K=2. This is expected since there are more spurious matches from K=2 that lead to worse performance. Two success cases are shown in Figure 8 and a failure in Figure 9 (the graphs are shown simplified for visualisation purposes and do not represent the entire available graph).

Methods		**Acc** ↓	**Comp** ↓	**Precision** ↑	**Recall** ↑	**F1-Score** ↑
GeoTr [30]		0.20721	**0.03835**	**0.94180**	0.79118	0.83667
Ours	K=1	**0.00944**	0.09347	0.90873	**0.97444**	**0.93575**
	K=2	0.01215	0.10641	0.89234	0.97042	0.92345

Table 8. **Point cloud mosaicking from multiple point clouds. SGAligner** performs best in 3 out of the 5 metrics, with K = 1. The drop in the other 2 is due to a few low overlap pairs where node alignment fails, hence, registration too. *Best* values in **bold**.

4.4. Finding Overlapping Scenes

We discussed that **SGAligner** provides less than $O(N^2)$ computation complexity when addressing the task of registering multiple 3D scenes for which we have no knowledge of whether they overlap or not. Another approach to avoid full registration on all pairs (standard registration methods), is to use a retrieval-based approach. We consider the following approach as baseline: (i) extract local 3D keypoints for all available 3D point clouds [48]; (ii) generate a 3D descriptor per extracted keypoint [36]; (iii) accumulate the 3D keypoint descriptors into a global descriptor for each point cloud [16], and (iv) perform kNN search to rank global descriptors based on the queried one. Similarly here, this experiment serves as a demonstration of the potential of **SGAligner** and does not aim to solve the task.

Specifically, given a point cloud, we extract keypoints

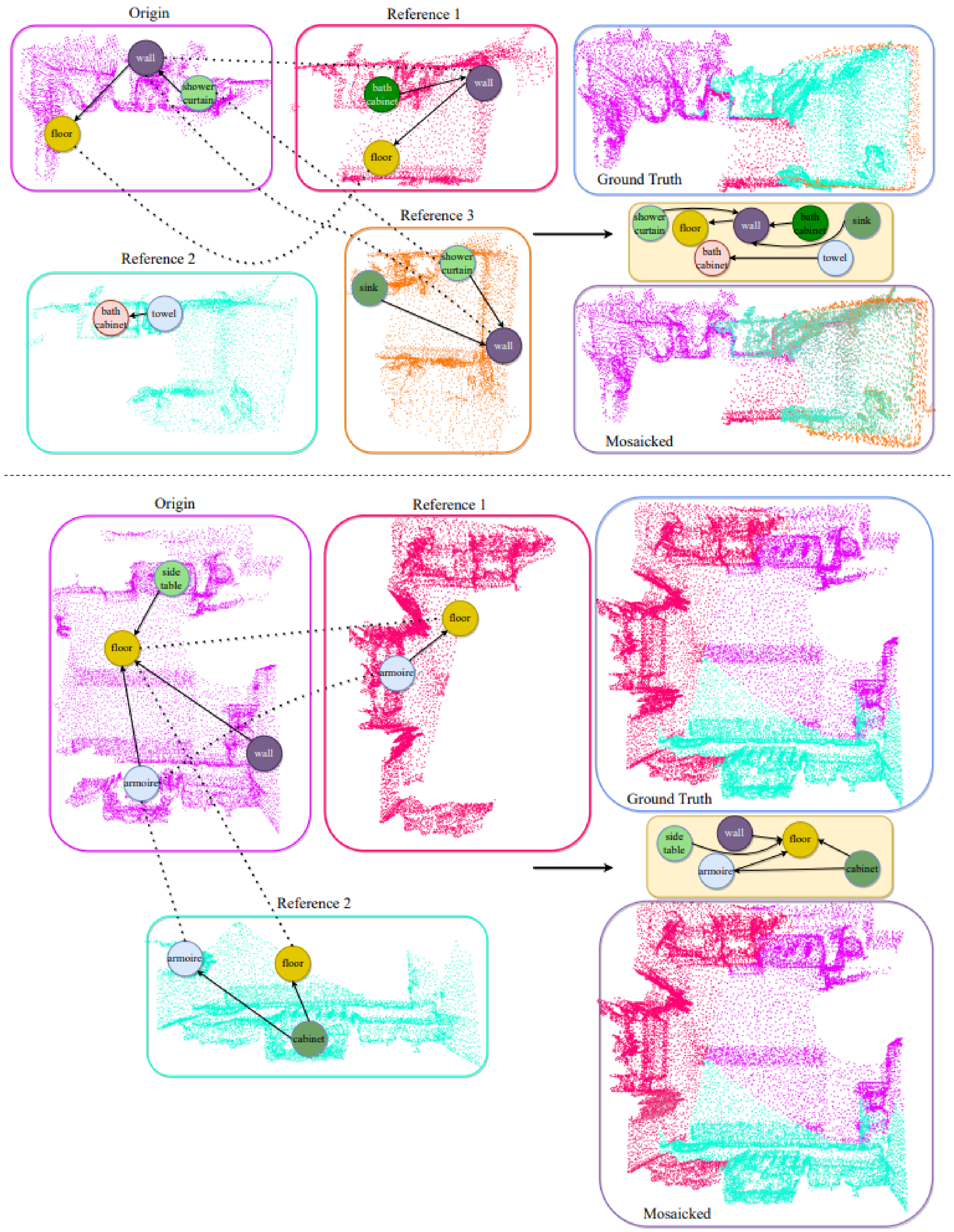

Figure 8. **Qualitative Results on Point Cloud Mosaicking.** Given partial point clouds of a scene and the corresponding 3D scene graphs, we showcase two example results on Point Cloud Mosaicking and the creation of a unified scene graph using the methodology discussed in Section 4.3. The solid lines show the relationships between objects and dashed lines represent the ground truth entity pairs $\mathcal{F}$.

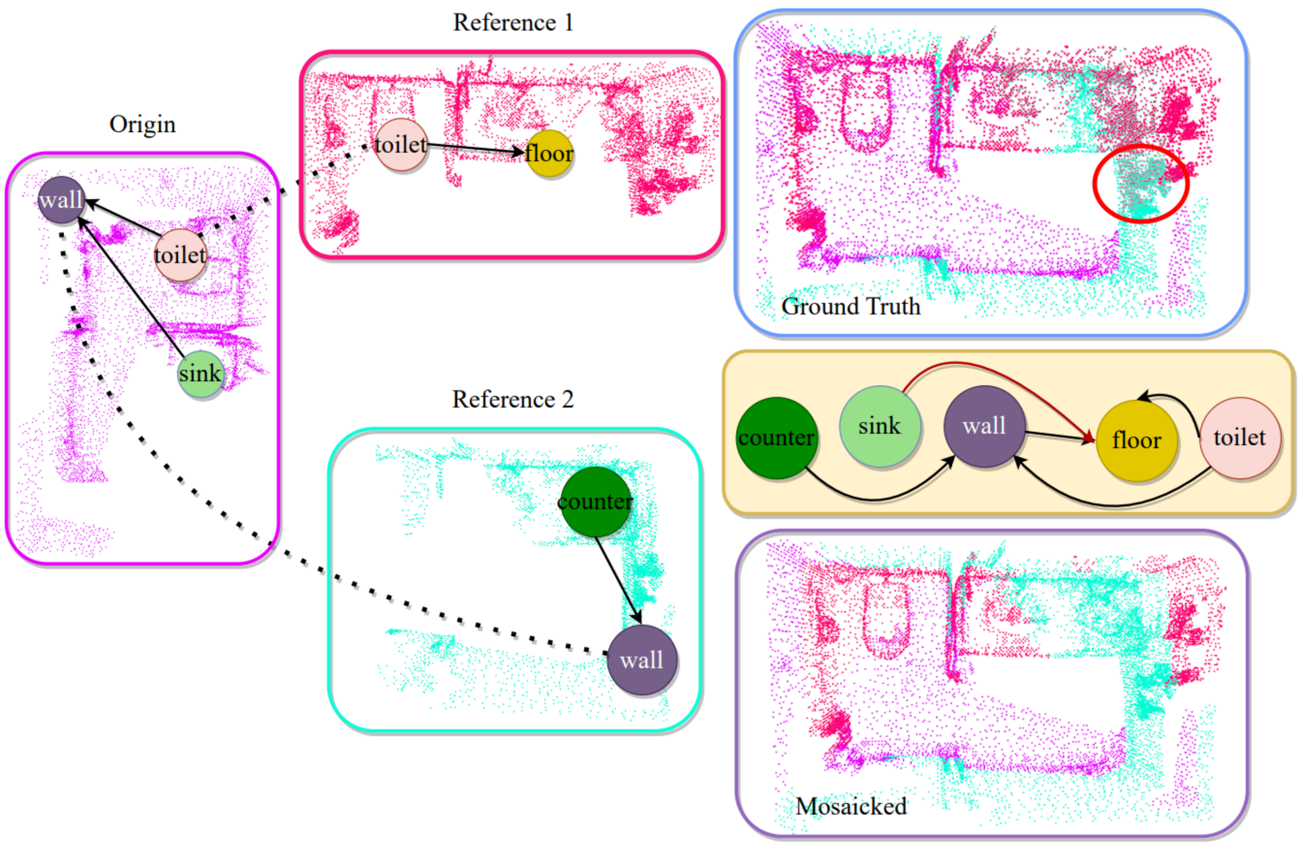

Figure 9. **Failure on Point Cloud Mosaicking.** Similar to the success cases in Fig. 8, we also showcase a failure case of our approach on point cloud mosaicking. The solid lines show the relationships between objects and dashed lines represent the ground truth entity pairs $\mathcal{F}$. The red circle on the ground truth point cloud depicts the area where the failure is the most visible and red arrow demonstrates the misalignment of **SGAligner** between a *sink* and a *floor*.

from the entire scene based purely on geometry and without any notion of object-ness or semantics. We randomly select 500 keypoints and their descriptors per scene, which we use to train [16] so as to generate optimized global descriptors. During inference and given a query point cloud and its corresponding global descriptor, we perform a kNN search to identify the closest neighbors of it in the rest of the point clouds. We evaluate on Mean Reciprocal Rank (MRR) and compare with **SGAligner**.

Results are shown in Table 9. We evaluate on different point cloud densities, ranging from using the full point cloud density offered in [40] to random subsampling for 10, 20, 30, and 50%. Please note that in **SGAligner**, we do not use the entire scene, only objects in the scene graph. As described in the main paper, we downsample object point clouds using farthest point sampling to 512 points. Any further subsampling takes place on the 512 points. We follow the same protocol for **SGAligner** and perform evaluation per subsampled scene level.

As expected, the retrieval-based method is performing well when there is a dense point cloud, since our method employs a limited amount of points per object instance. However, VLAD+KNN cannot retain a robust performance when density decreases, already reaching lower performance than **SGAligner** at 10% subsampling. In contrast, our approach is barely affected by a changing density since it already operates on lower-resolution point clouds. This showcases that the topological information encoded in 3D scene graphs can lead to more robust results when dealing with common failure cases in global descriptors (*i.e.*, changes in point cloud density).

4.5. Aligning 3D Scenes with Changes

In this section, we investigate the task of aligning a new 3D scene (target) on a prior 3D map (source), where the new scene can overlap fully or partially with the prior map and may contain changes (both geometric and semantic). Specifically, we investigate the following scenarios: (i) aligning a *local* 3D scene on a *larger* prior map that contains no changes – here overlap of the local scene with the map is 100%; (ii) aligning a *local* 3D scene on a *larger* prior map that contains changes; and (iii) aligning a *local*

Subsampling %	VLAD + KNN	SGAligner
0	<u>**0.557**</u>	0.383
10	0.316	**0.356**
20	0.276	**0.343**
30	0.222	**0.339**
50	0.162	**0.312**

Table 9. **Mean Reciprocal Rank (↑) comparison with a retrieval-based approach.** *Best* results per subsampling level are in **bold**. **SGAligner** is robust to point clouds of lower density, showcasing the potential for light-weight and privacy-preserving communication between agents. *Overall best* in <u>**underlined bold**</u>.

3D scene on a *local* prior map that contains changes. We approach this as a 3D scene graph alignment task. For (i) we use as large maps the 3D scene graphs of the entire scenes offered in [40, 41]. For local 3D scenes we use the data generated above. The results in Table 10 show that performance depends on the size of the prior map, whether there is full or partial overlap, and on the existence of temporal differences. They also demonstrate that our method can handle the alignment of maps that showcase temporal changes, even if not explicitly trained for this purpose.

	Mean RR ↑	Hits @ ↑ K = 1	K = 2	K = 3	K = 4	K = 5	No. of Pairs
(i)	0.970	0.952	0.976	0.989	0.993	0.995	827
(ii)	0.934	0.907	0.933	0.960	0.966	0.972	110
(iii)	0.886	0.833	0.894	0.928	0.946	0.957	2262

Table 10. **Alignment of a local 3D scene to a prior 3D map with differences in overlap and changes.** Although the harder the scenario (iii) the lower the performance, our method can handle temporal changes even if not specifically trained for this.

4.6. Ablation Studies

We provide ablation experiments on **SGAligner** to further understand the performance of the node matching task.

Intra-graph alignment. To further validate how our model performs on aligning nodes between two 3D scene graphs (source-reference) with no/partial overlap, we formulate the *Intra-Graph Alignment Recall (IGAR)* metric. It measures the fraction of the nodes in the source graph that are aligned (K=1) with nodes in the same source graph or, in other words, how many node matches out of total are self-aligned. We provide these results in Table 11. We do not explicitly model **not** self-matching nodes within the same graph, yet, *IGAR* values show that our method rarely performs this.

$$IGAR = \frac{1}{M}\sum_{M}\frac{|pred\{n^i \equiv n^j\}|}{|\mathcal{F}|}, n \in \mathcal{N} \qquad (4)$$

where, $i \neq j, pred\{n^i \equiv n^j\}$ is the set of nodes in the graph which **SGAligner** aligned with nodes in the same graph, $\mathcal{F}$ is the set of ground truth anchor pairs, $\mathcal{N}$ is the set of objects in a single graph, and M is the total number of graphs.

Method	IGAR ↓ (%)
$\mathcal{P}$	16.9
$\mathcal{P} + \mathcal{S}$	16.5
$\mathcal{P} + \mathcal{S} + \mathcal{R}$	13.1
SGAligner	**8.2**

Table 11. **Evaluation on node self-alignment. SGAligner** has not been explicitly modeled to not create self-matches but still is able to differentiate between nodes from the same and different graphs.

Commonly confused classes. We compute a confusion matrix to identify which object categories are most frequently misaligned during entity alignment and if our method fails on certain semantic classes (*e.g.*, *chair*, *table*, etc). In Figure 10, we show the confusion matrix on all 4 module combinations of **SGAligner**. As expected, the object encoder module $\mathcal{P}$, although performing well, confuses the most *wall* and *floor* classes. This is due to the fact that purely on a semantic level, without encoding any positional/structural information, these classes are similar. We can further observe that **SGAligner** is robust to certain classes like *pillow*, *tv*, and *lamp*, but classes like *wall*, *floor*, and *fridge* are easy to misalign, albeit less than in $\mathcal{P}$.

Robustness to missing positional information. We evaluate the performance of **SGAligner** on node alignment when all the relationships encoding positional information between the nodes, such as `left` and `standing on`, are missing. Results are in Table 12. As expected, the structure module $\mathcal{S}$ suffers from this compared to the full ground-truth experiment, since the number of edges encoded in the neighborhood of an entity gets reduced. However, overall, our method does not show a drastic drop in performance. This can be attributed to the fact that we do not discriminate between different types of relationships in our encoders. This is an important robustness characteristic, especially when working with predicted scene graphs where the relationships could be missing or incorrectly labelled. This also shows that once trained with full ground truth, our method is able to handle missing data during inference which would be useful for an autonomous agent.

Robustness to point cloud resolution. To validate the robustness of our method, we provide an ablation study of **SGAligner** on node alignment while varying the resolution of the point cloud, $\mathcal{P}_i$, of each object $\mathcal{O}_i$. The results are in Table 13. Please note that the default setting is 512 points per object. Since we operate on a node-to-node level instead of raw metric level, **SGAligner** is able to retain performance on node alignment while the number of points per object goes down from 512 to 64, with a drop in

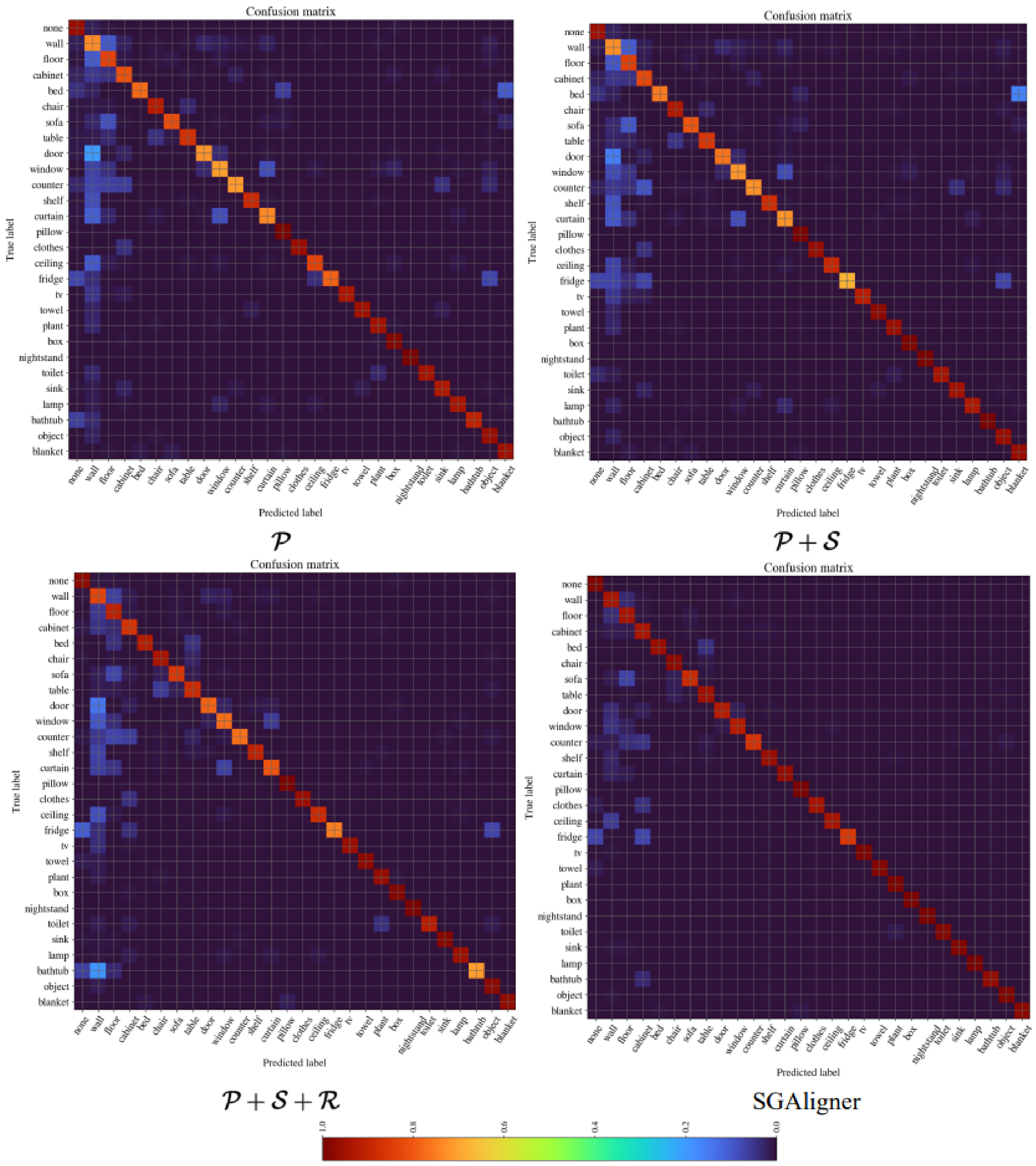

Figure 10. **Object Confusion Matrices** of the 4 module combinations of **SGAligner**: object encoder ($\mathcal{P}$); object and structure encoders ($\mathcal{P}+\mathcal{S}$); object, structure and relationship encoders ($\mathcal{P}+\mathcal{S}+\mathcal{R}$); and the proposed method with all modules (**SGAligner**). High values indicate that an object (denoted on y-axis) is often recognized as the object denoted on x-axis – everything but the diagonal should be 0.

Modalities	Mean RR ↑	Hits @ ↑ K = 1	K = 2	K = 3	K = 4	K = 5
$\mathcal{P}$	0.884	0.835	0.886	0.921	0.938	0.951
$\mathcal{P}+\mathcal{S}$	0.880	0.830	0.882	0.918	0.936	0.948
$\mathcal{P}+\mathcal{S}+\mathcal{R}$	0.893	0.844	0.898	0.933	0.949	0.959
SGAligner	**0.948**	**0.921**	**0.952**	**0.971**	**0.979**	**0.985**

Table 12. **Evaluation on node matching with missing positional relationships.** We compare the performance of **SGAligner** for different modality combinations – module $\mathcal{S}$ suffers the most since layout information is an important part of its graph structure.

MRR of about 3%. This is an interesting robustness characteristic, especially when working with embodied robots on low compute applications. Potentially, our module can be plugged into 3D scene graph construction and optimisation pipelines such as Hydra [15] and Kimera [35] since it can help align 3D data with low point cloud density, typical for real-time applications.

PC res.	Mean RR ↑	Hits @ ↑ K = 1	K = 2	K = 3	K = 4	K = 5
64	0.923	0.892	0.915	0.936	0.952	0.967
128	0.934	0.906	0.929	0.948	0.962	0.971
256	0.946	0.914	0.941	0.968	0.976	0.982
512	**0.950**	**0.923**	**0.957**	**0.974**	**0.982**	**0.987**

Table 13. **Evaluation on node matching with different point cloud resolutions.** We compare the performance of **SGAligner** varying the number of points per object, abbreviated as *PC res.*, provided as input to the object encoder, $\mathcal{P}$. The drop in performance is small considering that we drastically downsample up to $\times 8$ the initial resolution.

5. Conclusion

We presented SGAligner, the first method capable of aligning 3D scene graphs directly on the graph level, that is robust to the in-the-wild scenarios, such as unknown overlap between scenes or changing environments. We demonstrated that aligning the scenes directly on the scene graph level can improve downstream tasks (*e.g.*, point cloud alignment) in terms of accuracy and speed. We believe our work could unlock agents to leverage this emerging scene representation for creating 3D maps of the environment, further using it for and sharing it with downstream tasks.

References

[1] Christopher Agia, Krishna Murthy Jatavallabhula, Mohamed Khodeir, Ondrej Miksik, Vibhav Vineet, Mustafa Mukadam, Liam Paull, and Florian Shkurti. Taskography: Evaluating robot task planning over large 3d scene graphs. In *Conference on Robot Learning*, pages 46–58. PMLR, 2022. 1, 3

[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*, pages 5664–5673, 2019. 1, 2

[3] Xuyang Bai, Zixin Luo, Lei Zhou, Hongbo Fu, Long Quan, and Chiew-Lan Tai. D3feat: Joint learning of dense detection and description of 3d local features. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6359–6367, 2020. 2, 3

[4] Daniel Barath and Jiří Matas. Graph-cut ransac. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6733–6741, 2018. 5

[5] Daniel Barath, Jana Noskova, Maksym Ivashechkin, and Jiri Matas. Magsac++, a fast, reliable and accurate robust estimator. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1304–1312, 2020. 5

[6] Yun Chang, Luca Ballotta, and Luca Carlone. D-lite: Navigation-oriented compression of 3d scene graphs under communication constraints. *arXiv preprint arXiv:2209.06111*, 2022. 1, 2

[7] Liyi Chen, Zhi Li, Yijun Wang, Tong Xu, Zhefeng Wang, and Enhong Chen. Mmea: entity alignment for multi-modal knowledge graph. In *Knowledge Science, Engineering and Management: 13th International Conference, KSEM 2020, Hangzhou, China, August 28–30, 2020, Proceedings, Part I 13*, pages 134–147. Springer, 2020. 1, 2

[8] Liyi Chen, Zhi Li, Tong Xu, Han Wu, Zhefeng Wang, Nicholas Jing Yuan, and Enhong Chen. Multi-modal siamese network for entity alignment. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 118–126, 2022. 2

[9] Bo Cheng, Jia Zhu, and Meimei Guo. Multijaf: Multi-modal joint entity alignment framework for multi-modal knowledge graph. *Neurocomputing*, 500:581–591, 2022. 1, 2

[10] Helisa Dhamo, Fabian Manhardt, Nassir Navab, and Federico Tombari. Graph-to-3d: End-to-end generation and manipulation of 3d scenes using scene graphs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16352–16361, 2021. 3

[11] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communication of ACM*, 1981. 2, 3, 5

[12] Samir Yitzhak Gadre, Kiana Ehsani, Shuran Song, and Roozbeh Mottaghi. Continuous scene representations for embodied ai. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14849–14859, June 2022. 1, 3

[13] Hao Guo, Jiuyang Tang, Weixin Zeng, Xiang Zhao, and Li Liu. Multi-modal entity alignment in hyperbolic space. *Neurocomputing*, 461:598–607, 2021. 1, 2

[14] Shengyu Huang, Zan Gojcic, Mikhail Usvyatsov, Andreas Wieser, and Konrad Schindler. Predator: Registration of 3d point clouds with low overlap. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021. 2, 3, 17, 18

[15] N. Hughes, Y. Chang, and L. Carlone. Hydra: A real-time spatial perception system for 3D scene graph construction and optimization. 2022. 2, 3, 15

[16] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3304–3311. IEEE, 2010. 10, 12

[17] Ziyuan Jiao, Yida Niu, Zeyu Zhang, Song-Chun Zhu, Yixin Zhu, and Hangxin Liu. Sequential manipulation planning on scene graph. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8203–8210. IEEE, 2022. 1, 3

[18] Ue-Hwan Kim, Jin-Man Park, Taek-Jin Song, and Jong-Hwan Kim. 3-d scene graph: A sparse and semantic representation of physical environments for intelligent agents. *IEEE transactions on cybernetics*, 50(12):4921–4933, 2019. 1, 2

[19] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123:32–73, 2017. 2

[20] Xinghang Li, Di Guo, Huaping Liu, and Fuchun Sun. Embodied semantic scene graph generation. In *Conference on Robot Learning*, pages 1585–1594. PMLR, 2022. 1, 3

[21] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. Graph matching networks for learning the similarity of graph structured objects. In *International conference on machine learning*, pages 3835–3845. PMLR, 2019. 2

[22] Yongwei Li, Yalong Ma, Xiang Huo, and Xinkai Wu. Remote object navigation for service robots using hierarchical knowledge graph in human-centered environments. *Intelligent Service Robotics*, 15(4):459–473, 2022. 1, 2

[23] Zhenxi Lin, Ziheng Zhang, Meng Wang, Yinghui Shi, Xian Wu, and Yefeng Zheng. Multi-modal contrastive representation learning for entity alignment. *arXiv preprint arXiv:2209.00891*, 2022. 1, 2, 3, 4, 17

[24] Fangyu Liu, Muhao Chen, Dan Roth, and Nigel Collier. Visual pivoting for (unsupervised) entity alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4257–4266, 2021. 1, 2, 7

[25] Fangyu Liu, Rongtian Ye, Xun Wang, and Shuaipeng Li. Hal: Improved text-image matching by mitigating visual semantic hubs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:11563–11571, 04 2020. 2

[26] Ye Liu, Hui Li, Alberto Garcia-Duran, Mathias Niepert, Daniel Onoro-Rubio, and David S Rosenblum. Mmkg: multi-modal knowledge graphs. In *The Semantic Web: 16th International Conference, ESWC 2019, Portorož, Slovenia, June 2–6, 2019, Proceedings 16*, pages 459–474. Springer, 2019. 1, 2

[27] Samuel Looper, Javier Rodriguez-Puigvert, Roland Siegwart, Cesar Cadena, and Lukas Schmid. 3d vsg: Long-term semantic scene change prediction through 3d variable scene graphs. *arXiv preprint arXiv:2209.07896*, 2022. 3

[28] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. 17

[29] 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*, pages 652–660, 2017. 4, 7

[30] Zheng Qin, Hao Yu, Changjian Wang, Yulan Guo, Yuxing Peng, and Kai Xu. Geometric transformer for fast and robust point cloud registration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022. 2, 3, 5, 8, 10, 17

[31] Rahul Raguram, Ondrej Chum, Marc Pollefeys, Jiri Matas, and Jan-Michael Frahm. Usac: A universal framework for random sample consensus. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):2022–2038, 2012. 5

[32] Zachary Ravichandran, J Daniel Griffith, Benjamin Smith, and Costas Frost. Bridging scene understanding and task execution with flexible simulation environments. *arXiv preprint arXiv:2011.10452*, 2020. 1, 3

[33] Zachary Ravichandran, Lisa Peng, Nathan Hughes, J. Daniel Griffith, and Luca Carlone. Hierarchical representations and explicit memory: Learning effective navigation policies on 3d scene graphs using graph neural networks. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 9272–9279, 2022. 1, 2

[34] Antoni Rosinol, Arjun Gupta, Marcus Abate, Jingnan Shi, and Luca Carlone. 3d dynamic scene graphs: Actionable spatial perception with places, objects, and humans. *arXiv preprint arXiv:2002.06289*, 2020. 1, 2

[35] Antoni Rosinol, Andrew Violette, Marcus Abate, Nathan Hughes, Yun Chang, Jingnan Shi, Arjun Gupta, and Luca Carlone. Kimera: From slam to spatial perception with 3d dynamic scene graphs. *The International Journal of Robotics Research*, 40(12-14):1510–1546, 2021. 1, 2, 15

[36] 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. 10

[37] Gabriel Sepulveda, Juan Carlos Niebles, and Alvaro Soto. A deep learning based behavioral approach to indoor autonomous navigation. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 4646–4653. IEEE, 2018. 1, 2

[38] Yinghui Shi, Meng Wang, Ziheng Zhang, Zhenxi Lin, and Yefeng Zheng. Probing the impacts of visual context in multimodal entity alignment. In *Web and Big Data: 6th International Joint Conference, APWeb-WAIM 2022, Nanjing, China, November 25–27, 2022, Proceedings, Part II*, pages 255–270. Springer, 2023. 1, 2

[39] Jiaming Sun, Yiming Xie, Linghao Chen, Xiaowei Zhou, and Hujun Bao. NeuralRecon: Real-time coherent 3D reconstruction from monocular video. *CVPR*, 2021. 10

[40] Johanna Wald, Armen Avetisyan, Nassir Navab, Federico Tombari, and Matthias Nießner. Rio: 3d object instance relocalization in changing indoor environments. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7658–7667, 2019. 2, 5, 6, 12, 13

[41] Johanna Wald, Helisa Dhamo, Nassir Navab, and Federico Tombari. Learning 3d semantic scene graphs from 3d indoor reconstructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3961–3970, 2020. 1, 2, 3, 5, 7, 13

[42] Shun-Cheng Wu, Johanna Wald, Keisuke Tateno, Nassir Navab, and Federico Tombari. Scenegraphfusion: Incremental 3d scene graph prediction from rgb-d sequences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7515–7525, 2021. 2

[43] Zi Jian Yew and Gim Hee Lee. Rpm-net: Robust point matching using learned features. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11824–11833, 2020. 8, 18

[44] Zi Jian Yew and Gim Hee Lee. Regtr: End-to-end point cloud correspondences with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022. 2, 3

[45] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *CVPR*, 2017. 5, 8

[46] Qingheng Zhang, Zequn Sun, Wei Hu, Muhao Chen, Lingbing Guo, and Yuzhong Qu. Multi-view knowledge graph embedding for entity alignment. *arXiv preprint arXiv:1906.02390*, 2019. 1, 2

[47] Ruohan Zhang, Dhruva Bansal, Yilun Hao, Ayano Hiranaka, Jialu Gao, Chen Wang, Roberto Martín-Martín, Li Fei-Fei, and Jiajun Wu. A dual representation framework for robot learning with human guidance. In *6th Annual Conference on Robot Learning*. 1

[48] Yu Zhong. Intrinsic shape signatures: A shape descriptor for 3d object recognition. In *2009 IEEE 12th international conference on computer vision workshops, ICCV workshops*, pages 689–696. IEEE, 2009. 10

A. Appendix

In this section, we provide additional details about implementation and evaluation metrics used in the paper.

A.1. Implementation Details

Inspired by MCLEA [23], we use a multi-layered GAT with 2 layers and each hidden unit being 128-dimensional. All modules output a 100-dimensional embedding and the joint embedding, being a weighted concatenation, is 400-dimensional. We use $\mathcal{T}_1$ for ICL loss as 0.1 and $\mathcal{T}_2$ for IAL loss as 1.0. We train our model for 50 epochs on an NVIDIA GeForce RTX 3060 Ti 8GB GPU with a batch size of 4 using AdamW [28] optimizer and a learning rate of 0.001.

A.2. Evaluation Metrics

The evaluation metrics that we use to assess performance in Section 4 are formally defined in this section.

A.2.1 Alignment Metrics

Inspired by works in multi-modal entity alignment [23], we define the alignment metrics as follows:

Hits @ K represents the fraction of true anchor entities present in the top k predictions:

$$H_k(r_1, ..., r_n) = \frac{1}{n}\sum_{i=1}^{n} I[r_i \leq k] \quad (5)$$

where, $I[x \leq y] = 1$ when $x \leq y$ else 0 and $k \in [1, 2, 3, 4, 5]$.

Mean Reciprocal Rank (MRR) corresponds to the arithmetic mean over the reciprocals of ranks of true triples.

$$MRR(r_1, ..., r_n) = \frac{1}{n}\sum_{i=1}^{n} r_i^{-1} \quad (6)$$

A.2.2 Registration Metrics

Feature Matching Recall (FMR) [14][30] measures the fraction of point cloud pairs for which, based on the number of inlier correspondences, it is likely that accurate transformation parameters can be recovered with a robust estimator such as RANSAC. It should be noted that FMR simply verifies whether the inlier ratio (IR) is higher than a threshold $\mathcal{T}$ = 0.05. It does not examine if the transformation can actually be inferred from those correspondences, which is not always the case because of the possibility that their geometric arrangement is (almost) degenerate, such as when they are situated closely together or along a straight edge.

$$FMR = \frac{1}{M}\sum_{i=1}^{M}[\![IR_i > \mathcal{T}]\!] \quad (7)$$

where M is the number of all point cloud pairs.

Registration Recall (RR) is the fraction of registered point cloud pairs for which the transformation error is smaller than 0.2m. The transformation error is the root mean squared error of the ground truth correspondence $\mathcal{H}^*$ after applying the predicted transformation $\mathbf{T}_{P\to Q}$.

$$RMSE = \sqrt{\frac{1}{|\mathcal{H}^*|}\sum_{({p^*}_{x_i},{q^*}_{y_i})\in\mathcal{H}^*}||T_{P\to Q}({p^*}_{x_i}) - {q^*}_{y_i}||_2^2} \quad (8)$$

$$RR = \frac{1}{M}\sum_{i=1}[\![RMSE_i < 0.2m]\!] \quad (9)$$

Relative Rotation Error (RRE) is the geodesic distance in degrees between estimated and ground-truth rotation matrices.

$$RRE = arccos(\frac{trace(R^T \cdot \bar{R} - 1)}{2}) \quad (10)$$

Relative Translation Error (RTE) is the euclidean distance between estimated and ground-truth translation vectors.

$$RTE = ||t - \bar{t}|| \quad (11)$$

We compute mean RRE and RTE between all the registered point cloud pairs.

Chamfer Distance measures the quality of registration. Following [43], [14], we use the *modified* Chamfer distance metric :

$$\bar{CD}(P,Q) = \frac{1}{|P|}\sum_{p\in P} min_{q\in Q_{raw}}||{T_P}^Q(p) - q||_2^2 + \frac{1}{|Q|}\sum_{q\in Q} min_{p\in P_{raw}}||q - {T_P}^Q(p)||_2^2 \quad (12)$$

where, P_{raw} and Q_{raw} are $raw/clean$ source and target point clouds respectively.

A.2.3 Mosaicking Metrics

The definition of full 3D point cloud mosaicking metrics is provided in Table 14.

Metric	Definition
Acc	$mean_{p\in P}(min_{p^*\in P^*}\|\|p - p^*\|\|)$
Comp	$mean_{p^*\in P^*}(min_{p\in P}\|\|p - p^*\|\|)$
Precision	$mean_{p\in P}(min_{p^*\in P^*}\|\|p - p^*\|\| < 0.05)$
Recall	$mean_{p^*\in P^*}(min_{p\in P}\|\|p - p^*\|\| < 0.05)$
F1-Score	$\frac{2*precision*recall}{precision+recall}$

Table 14. **3D Mosaicking Metric Definitions.** p and p^* are ground truth and mosaicked point clouds respectively.