Lookalike3D: Seeing Double in 3D

Chandan Yeshwanth **Angela Dai**
Technical University of Munich

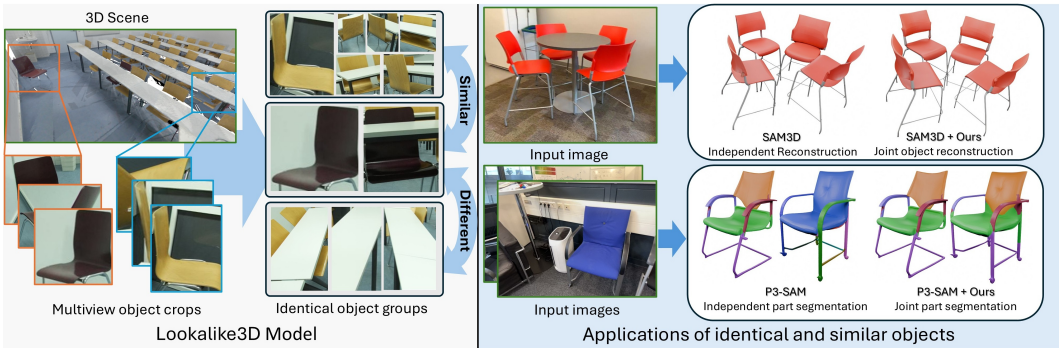

Figure 1: Many real-world scenes naturally contain repeated objects. Lookalike3D proposes to identify such groups of identical or near-identical (similar) objects across multiview images of an indoor 3D scene. This enables holistic reasoning to be exploited in object-focused 3D reconstruction and perception methods. In particular, we demonstrate joint optimization under these global scene constraints using state-of-the-art, pre-trained 3D object reconstruction and part segmentation models.

Abstract

3D object understanding and generation methods in indoor scenes produce impressive results, yet they often overlook a pervasive source of information in real-world scenes: repeated objects. We introduce the task of lookalike object detection in indoor scenes, which leverages repeated and complementary cues from identical and near-identical object pairs. Given an input scene, the task is to classify pairs of objects as identical, similar or different using multiview images as input. To address this, we present Lookalike3D, a multiview image transformer that effectively distinguishes such object pairs through similarity learning by harnessing strong semantic priors from large image foundation models. To support this task, we collected the 3DTwins dataset, containing 76k manually annotated identical, similar and different pairs of objects based on ScanNet++, and show an improvement of 14% IoU over baselines on the task of lookalike object detection. We also demonstrate how our method improves the downstream tasks of joint 3D object reconstruction and part co-segmentation, turning repeated and lookalike objects into a powerful cue for consistent, high-quality 3D scene understanding. Our code, dataset and models will be made publicly available at `https://cy94.github.io/lookalike3d/`.

1 Introduction

3D scene understanding and generation are important building blocks of content creation, design, and virtual reality (VR) applications, all of which rely on clean and consistent 3D assets for fidelity. Recent work on object-level 3D generation and perception has made impressive strides; for instance, 3D object generative methods can produce high-fidelity object-level assets from image inputs [1, 2, 3].

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Other methods show excellent results in making 3D assets decomposable or interactable by inferring object part decomposition [4, 5] and articulation [6, 7, 8]. However, these methods largely ignore a pervasive cue in real-world scenes: repeated objects. Recognizing which objects are identical or near-identical provides rich, complementary information that can significantly improve reconstruction, part understanding, and semantic reasoning.

Current image-to-3D methods are most reliable when objects are fully visible. When objects are partially occluded, these models must hallucinate missing geometry, since they operate primarily on single objects with limited context. However, real indoor scenes often contain multiple instances of the same object, offering complementary visual perspectives. For example, one chair may be seen from the front while another is seen from the side. Despite their ubiquity, repeated objects are rarely leveraged in 3D reconstruction pipelines.

We observe that object pairs in indoor scenes [9, 10, 11] naturally fall into three categories: 1) *identical* pairs that have exactly identical shape and texture, 2) *similar* pairs: distinct objects that differ slightly in shape or texture, 3) *different* pairs that differ significantly in either shape or texture. *Identical* and *similar* pairs contain highly redundant information that can be exploited for joint reconstruction and semantic understanding. Motivated by this, we introduce the task of *lookalike object detection*: given input multiview images of objects in an indoor scene, the task is to classify each pair of objects within the same semantic class as identical, similar, or different.

To solve this task, we formulate lookalike detection as a similarity learning problem. Our method, Lookalike3D, takes multiview images of object pairs and embeds them into a feature space where proximity reflects similarity. We leverage the strong semantic prior of image foundation models such as DINOv2 [12] and propose an alternating attention [13] model paired with a multi-class triplet loss and score alignment loss for similarity learning. Our model can effectively capture fine-grained appearance and geometry cues in multiview images which are not available in coarse 3D point cloud inputs.

We demonstrate the utility of Lookalike3D on two downstream tasks: single-view joint 3D object reconstruction of identical objects from casual camera captures, and part co-segmentation of similar objects. In both cases, we show that jointly optimizing identical or similar pairs of objects produces more consistent and detailed reconstructions or segmentations compared to treating objects independently.

To enable this task, we collected 3DTwins, a large-scale dataset of identical and near-identical objects based on the ScanNet++ dataset [9] of 3D indoor scans containing 76k object pairs spanning identical, similar, and different categories. This dataset enables training, benchmarking, and evaluation of lookalike object detection methods and provides a standard for object matching in 3D indoor scenes.

Our contributions are:

- we introduce the task of lookalike object detection, a task to identify identical and similar object pairs in 3D indoor scenes.
- we present a multiview image transformer-based similarity learning model with alternating attention layers to solve this task.
- we provide a large-scale dataset of identical, similar and different object pairs annotated in real world 3D scans to enable training and benchmarking.
- we demonstrate applications of our method on two downstream tasks: consistent 3D reconstruction of identical objects, and part co-segmentation of similar objects.

2 Related Work

2D Object Re-identification Re-identification is a widely explored computer vision task operating on RGB images. Given a query image of an object, the task is to retrieve the closest images from a database which contain the same object, under varying viewpoint, lighting conditions and occlusions. Most re-identification research has focused on two categories: humans and vehicles. Commonly used benchmarks for vehicle re-identification are VeRi-776 [14] and PKU VehicleID [15], and methods have used self-supervised learning [16], deep metric learning through classification and triplet losses on multiview images [17] or contrastive loss [18] among others. Human re-identification methods [19, 20] use benchmarks such as Market-1501 [21] and solve the task similarly. These methods

are restricted to a single object class and hence do not generalize to diverse and long-tail objects in 3D scenes. Recent work such as VICP [22] has attempted to solve the problem of general object re-identification on a larger set of classes, by similarity learning and in-context prompting. However, these methods are limited to a fixed set of classes and do not utilize multi-view images. Unlike re-identification approaches that output ranked lists, we leverage multi-view data to explicitly classify object pairs as identical, similar, or different.

3D Object Re-identification Analogous to the 2D setting, 3D re-identification methods associate previously seen 3D instances with new ones from point clouds or other 3D inputs. 3D Semantic MapNet [23] matches objects one-to-one via coarse object-level features; Living Scenes [24] and Rescene4D [25] associate or segment instances consistently over time; and RIO [26] addresses object re-localization in 3D scenes. Such object-centric pipelines, as well as broader indoor 3D scene parsing methods for instance segmentation [27] and open-vocabulary recognition [28], use coarse semantics or treat instances independently, preventing grouping of identical or similar objects. In contrast, we use RGB images to group objects via fine-grained appearance and shape cues, producing scene-level groupings that leverage 3D foundation models for joint reconstruction.

2D Image Matching RGB image matching is a related but more general task. Methods such as SuperGlue [29] and ASpanformer [30] predict sparse keypoint matches, while Mast3r [31] and TLFR [32] predict dense pixel matches; CroCov2 [33] and ZeroCo [34] use masked view completion for stereo and dense matching. These methods match multiple views of the same object and depend heavily on surrounding context, often lacking multi-view support and object-level detail. In contrast, Lookalike3D matches different instances of identical and similar objects, is robust to background changes, and effectively uses multi-view input.

3D Point Cloud Registration 3D point cloud registration aims to predict point-level matches between two 3D objects or scenes. Methods such as Lepard [35], Predator [36], RTMM [37] and REGTR [38] are designed to register point clouds of the same object or a complete scene with partial overlap. These methods tend to match based on coarse semantics (chair arm, leg, back) and not on fine-grained appearance details and hence predict confident matches on non-identical objects (e.g., chair leg matching with another different type of chair's leg). In contrast, Lookalike3D can effectively distinguish objects from the same semantic class with different shape or appearance.

Repeated Objects, Co-segmentation, and 3D Reconstruction Neural 3D shape co-segmentation methods [39, 40] jointly segment related shapes via shared part representations. Lookalike3D instead first identifies similar objects and transfers segmentation from a well-segmented source to an undersegmented target. Recent object- and scene-level 3D reconstruction methods [1, 2, 3, 41, 42, 43, 44] reconstruct objects in isolation; we identify *when* multiple instances should be reconstructed jointly. Works leveraging repeats rely on specific input assumptions: JRM [45] needs 3D and 2D inputs and does not explicitly classify pairs; FurnSet [46] takes a single image without classification; Splat and Replace [47] needs user clicks; and SfD [48] requires all instances and masks in one image. JRM and FurnSet also reconstruct each instance separately, yielding slightly different geometries for truly identical objects. Lookalike3D instead explicitly finds identical pairs from multiview input, and optimizes a single joint reconstruction via a 3D foundation model [1] which is guaranteed to be consistent at the scene-level.

3 Method

Problem Formulation. Indoor scenes often contain valuable information in the form of repeated object instances. The task of lookalike object detection is to identify such groups of identical objects and pairs of similar objects in the scene. We take as input a sequence of RGB-D images containing a set of object instances $o_i = (m_i, \{I_{i,j}\})$ where m_i is the binary mask of the 3D object, and $I_{i,j}$ are the multiview crops of the object o_i. The object instances can be obtained from ground truth data or off-the-shelf models that predict object masks on RGB images [49] and aggregate them [50]. RGB-D sequences are required only during training to associate the multiview images together; inference can be performed on a single RGB image.

The task is to predict groups of identical objects $\mathsf{Id}_i = (o_{\mathsf{Id}_{i,1}}, o_{\mathsf{Id}_{i,2}}, \ldots)$ that have identical shape and appearance, and pairs of similar objects $\mathsf{Sim} = ((o_{\mathsf{Sim}_{1,1}}, o_{\mathsf{Sim}_{1,2}}), \ldots)$ that have near-identical shape or appearance. The remaining pairs of objects are classified into Diff (different). The RGB-D setting with a dense trajectory of images is particularly useful for this task as it allows for capturing both

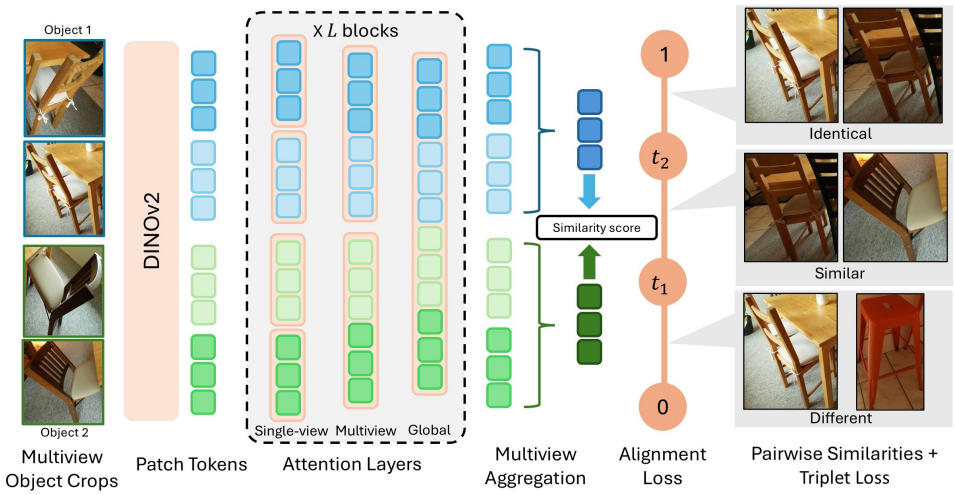

Figure 2: Overview of the Lookalike3D model. Lookalike3D takes as input a single image or multiple posed images each of a pair of objects, passes these through a shared frozen DINOv2 backbone to extract image patch features, and then encodes these patch features with three kinds of attention layers: single-view, multiview and global. The resulting patch features are aggregated and compared to obtain a single similarity measure of the two objects. The model is trained with a triplet loss to separate the identical, similar and different pairs, and an alignment loss to align the similarities to the classification thresholds.

redundant and complementary information about objects, by observing them from different sides and fusing the results.

Overview of Lookalike3D First, Lookalike3D creates pairs of objects (o_1, o_2) in the scene and takes their image crops $I_{1,j}$ and $I_{2,j}$ as input, $j = 1 \dots k$ and k is the maximum number of input views for an object. These images are embedded into patch features using a shared feature extractor [12]. These patch features are then refined using a transformer encoder with self attention layers. The refined features are aggregated over all frames within an object, and compared across objects to compute similarity scores. These scores are used to classify the object pair into one of three categories: Id, Sim and Diff. Object pairs in Id share the exact same shape and appearance, while object pairs in Sim are near-identical. Detecting these two categories enables us to make use of rich information from repeated instances for various downstream applications.

Feature Extractor We use a shared DINOv2 [12] backbone as a semantic matching prior [51] to extract patch features from input image crops independently, giving N_p patch features $p_{i,j}$ for image $I_{i,j}$ which encode high-level object and part semantics. Additionally, we extract intermediate features from layers $(1, 3, 5, 8)$ to encode structural information of the objects [52].

Alternating Attention We utilize the transformer self-attention mechanism to process multiview images [53, 54, 13], and propose passing the DINOv2 patch features $p^0_{i,j}$ through transformer encoder layers at three hierarchical levels: single-view, multiview (object) and global. Single-view layers $\Phi_{\text{frame},l}$ apply self-attention to all tokens within a single object crop. Multiview (object-level) layers $\Phi_{\text{object},l}$ operate on all image tokens from the same object, and global layers $\Phi_{\text{global},l}$ operate on all tokens across both objects. At layer l:

$$\begin{aligned}
\text{Single-view:} \quad & p^l_{i,j} = \Phi_{\text{frame},l}(p^{l-1}_{i,j}) \\
\text{Multiview:} \quad & p^l_{i,j} = \Phi_{\text{object},l}(\{p^l_{i,j}\}), \quad \forall j \in \{1, \dots, k\} \\
\text{Global:} \quad & p^l_{i,j} = \Phi_{\text{global},l}(\{p^l_{i,j}\}), \quad \forall i \in \{1, 2\}, \forall j \in \{1, \dots, k\}
\end{aligned}$$

We add learned frame and object embeddings to the patch tokens in each layer to distinguish between tokens from different frames or different objects. We aggregate the patch features from the final layer across the input views for each object by averaging them to produce a single sequence of patch features for each object f_i. We then flatten the patch features, normalize them and compute the dot product of features of the two objects to obtain a single similarity score $s_{i,j} \in [0, 1]$ for objects i and j. We observe from the global attention scores that the foreground region mask is learned accurately

from the data; therefore, flattening the patch features does not negatively affect performance. The average operation allows Lookalike3D to perform inference on a single RGB image or fewer views than used for training.

Similarity Learning Objectives We use a combination of triplet and score alignment losses for similarity learning. The triplet loss optimizes the relative similarity between identical, similar and different pairs, while the alignment loss ensures that similarity scores remain within defined thresholds. This prevents the need to tune thresholds for diverse and long-tail indoor scenes.

Triplet Loss We use a multi-class triplet loss with margins [55, 19] for the similarity learning task of separating the three categories of pairs in the feature space: Id, Sim and Diff. For a given anchor pair of ground truth (GT) Id objects, we encourage its similarity to be higher than that of corresponding Sim or Diff pairs, resulting in two loss terms. The third term encourages similar pairs to be closer together in the feature space than different pairs. Since the triplet loss uses feature distances instead of similarities, we use the feature distance $d_{i,j} = 1 - s_{i,j}$. The combined loss is:

$$\mathcal{L}_{\text{triplet}} = \sum_{a \in \mathcal{A}} \left[d_{a,o_{\mathsf{Id}}} - d_{a,o_{\mathsf{Sim}}} + \alpha_1 \right]_+ + \left[d_{a,o_{\mathsf{Sim}}} - d_{a,o_{\mathsf{Diff}}} + \alpha_2 \right]_+ + \left[d_{a,o_{\mathsf{Id}}} - d_{a,o_{\mathsf{Diff}}} + \alpha_3 \right]_+$$

where $[x]_+ = \max(0, x)$ denotes the hinge function, $\mathcal{A}$ is the set of anchor objects and o_{Id}, o_{Sim} and o_{Diff} represent ground truth identical, similar and different objects corresponding to the anchor object a, and α_i denote the margins between the respective classes. We select hard negatives [19] for each loss term and find this essential for separating the three classes; for example, the first loss term compares the maximum Id distance with the minimum Sim distance.

Classification and Score Alignment Loss While the triplet loss $\mathcal{L}$ separates Id, Sim and Diff in the feature space, we need to set a hard threshold to classify pairs into one of these three types. Hence, we use two thresholds t_1 and t_2 in the interval $[0, 1]$ such that a pair (o_i, o_j) is labeled Id if $s_{i,j} \geq t_2$, Sim if $t_1 \leq s_{i,j} < t_2$, and Diff otherwise.

We encourage similarity scores to lie within these regions via a signed-distance loss against the ground-truth thresholds for the identical and different bins:

$$\mathcal{L}_{\text{align}} = \sum_{(o_i,o_j)} \begin{cases} (t_2 - s_{i,j}) + (1 - s_{i,j}) & \text{if } (o_i, o_j) \in \mathsf{Id} \\ (s_{i,j} - t_1) + s_{i,j} & \text{if } (o_i, o_j) \in \mathsf{Diff} \end{cases}$$

Scores for similar pairs are aligned to the Sim bin by the combination of $\mathcal{L}_{\text{triplet}}$ and $\mathcal{L}_{\text{align}}$. The overall loss is the sum of triplet and alignment losses $\mathcal{L} = \mathcal{L}_{\text{triplet}} + \mathcal{L}_{\text{align}}$. Finally, identical object pairs are transitively grouped together, while similar object pairs are kept as-is.

Implementation Details We set $(t_1, t_2) = (0.33, 0.66)$ within the similarity range $[0, 1]$. The top 5 views covering each object are selected. Inference takes on average only 34ms/sample with a batch size of 256, allowing evaluation on scenes with hundreds of object pairs. Other details are in the appendix, Sec. C.

3.1 Downstream Applications of Lookalike3D

Lookalike3D produces groups of identical objects and pairs of similar objects. We demonstrate applications of these on two downstream tasks: 3D object reconstruction and part co-segmentation.

3.1.1 Joint 3D Object Reconstruction

We use the identical objects detected by Lookalike3D for joint 3D object reconstruction. Run independently, SAM 3D produces reconstructions of identical objects that vary in shape and scale. Given a single image I containing k identical objects with masks $\{M_i\}_{i=1}^k$, we adapt SAM 3D to produce one coherent scene-level reconstruction fusing cues from all k instances.

SAM 3D's geometry and texture/refinement models originally condition on a single mask via cross-attention; we extend both to k masks: at each flow-transformer layer l, we condition on each M_i separately to obtain intermediate features $\{f_{l,i}\}$, average them into a single f_l in canonical space, and pass it to layer $l{+}1$. The sparse structure generator yields a single 64^3 shape and the layout decoder a single rotation, scale, and translation; the SLAT sampler then denoises the shape into SLAT features under the same per-mask conditioning, which the SLAT decoder converts to a final object mesh. To place the joint object in the scene, we use the median per-instance scale and reuse each instance's individually predicted rotation and translation. Further details are in the appendix (Sec. D).

Category	Train	Val	Overall
Identical Pairs	32,270	2,496	34,766
Similar Pairs	5,468	735	6,203
Different Pairs	33,320	2,450	35,770
Total Pairs	**71,058**	**5,681**	**76,739**
Unique Instances	22,285	1,563	23,848
Object Classes	867	148	1,015

Table 1: Statistics of the 3DTwins dataset



Figure 3: Examples of 3DTwins object pairs

We experiment with both a single-frame setting where all objects are observed in a single image, as well as a multi-frame setting where we choose the best frame covering each object.

3.1.2 Part Co-segmentation

We exploit predicted similar pairs from Lookalike3D for part transfer, since similar objects share largely the same part structure (e.g., chair leg, armrest, seat, back). P3-SAM [5] provides high-quality 3D part decomposition but under-segments objects with ambiguous geometry (e.g., fusing a chair's back and seat). We address this by reconstructing both objects with SAM 3D, aligning them via point-to-point ICP, and transferring part IDs to the target via k-NN majority vote from the source.

4 Dataset

We introduce a dataset of identical, similar, and different object pairs based on ScanNet++ [9]. We leverage paired 3D and RGB-D data, where dense image trajectories provide rich multiview information and 3D geometry enables consistent association across views. During training we use only the multiview images. A single human annotator labels each same-class pair as identical/similar/different using multiview images and 3D meshes, giving 76k annotated pairs. Identical pairs match exactly in shape and texture; similar pairs differ in at least one of these. Dataset statistics and examples are in Tab. 1 and Fig. 3; details are in the appendix (Sec. B).

5 Experiments

Evaluation Setting and Metrics We evaluate lookalike object detection on the 3DTwins validation set (corresponding to the ScanNet++ validation set, since the test set is hidden) using both ground-truth 3D instances and those predicted by MaskClustering [50]. To address dataset imbalance, we report Intersection-over-Union (IoU) per category (identical, similar, different) and overall.

5.1 Baselines

We compare our method against state-of-the-art 2D and 3D matching baselines which can operate on diverse indoor objects and fine-grained details, and can effectively distinguish similar and dissimilar regions of images and point clouds respectively. Re-identification methods are not directly comparable as they require large class-specific datasets, whereas our setting involves over 100 object classes.

2D matching baselines include Telling Left from Right (TLFR) [32], which combines Stable Diffusion [56] and DINOv2 [12] features; SuperGlue [29], ASpanformer [30], and Mast3r [31], which predict sparse keypoint or dense pixel matches; and Qwen3-VL-4B-Instruct [57], which we prompt to classify image pairs. 3D matching baselines include Lepard [35] and Predator [36]. We also compare against three DINO baselines: DINOv2-feats, which directly compares last-layer features from a pretrained DINOv2, DINO+MLP, which concatenates DINO features and classifies them three-way into our three classes, and DINO+triplet which learns feature embeddings using a triplet loss without cross attention.

For each baseline, we set two equally spaced thresholds on the number of matches, details are in the appendix (Sec. C.3). Since 2D matching methods operate on single views, we adapt them to the multiview setting by averaging match counts over 5 image pairs per object pair.

We also compare against three object-centric baselines: OpenShape [58] and ULIP-2 [59] feature embeddings, as well as the object encoder from Living Scenes [24] (zero-shot and fine-tuned on 3DTwins).

	Method	ScanNet++ GT Inst.				ScanNet++ Pred. Inst.			
		Id.	Sim.	Diff.	Overall	Id.	Sim.	Diff.	Overall
3D Methods	Lepard [35]	0.06	0.13	0.47	0.22	0.05	0.10	0.24	0.13
	Predator [36]	0.52	0.06	0.08	0.22	0.42	0.06	0.03	0.17
2D Methods	DINOv2-feats [12]	0.02	0.14	0.37	0.18	0.03	0.15	**0.37**	0.18
	SuperGlue [29]	0.03	0.07	0.44	0.18	0.11	0.02	0.09	0.07
	ASpanformer [30]	0.06	0.06	0.44	0.18	0.10	0.00	0.10	0.07
	Mast3r [31]	0.17	0.08	0.47	0.24	0.26	0.02	0.02	0.10
	TLFR [32]	0.29	0.12	0.33	0.24	0.16	0.08	0.05	0.09
	Qwen3-VL [57]	0.60	0.16	0.53	0.43	0.22	0.06	0.06	0.12
Object-centric	OpenShape (ZS) [58]	0.46	0.13	0.30	0.30	0.30	0.11	0.16	0.19
	ULIP-2 (ZS) [59]	0.36	0.12	0.22	0.23	0.30	0.12	0.11	0.18
	Living Scenes (ZS) [24]	0.29	0.10	0.24	0.21	0.21	0.10	0.13	0.15
	Living Scenes (FT) [24]	0.46	0.14	0.55	0.38	0.33	0.11	0.24	0.23
Learned Methods	DINO+MLP	0.45	0.17	0.60	0.41	0.31	0.15	0.28	0.25
	DINO+triplet	0.59	0.05	0.00	0.21	0.47	0.05	0.00	0.17
Ours	Lookalike3D	**0.65**	**0.18**	**0.65**	**0.49**	**0.50**	**0.15**	0.29	**0.31**

Table 2: Quantitative comparison on the 3DTwins dataset (ScanNet++). IoU for Identical (Id.), Similar (Sim.), and Different (Diff.) pairs, on GT and predicted instances. ZS: zero-shot, FT: finetuned. Best in **bold**, second best underlined.

5.2 Comparison to State of the Art

Quantitative results on the 3DTwins dataset are shown in Tab. 2. Lookalike3D outperforms all baselines on both GT and predicted instances, improving overall IoU by 0.06 and 0.13 respectively. Baselines either rely on coarse semantics (DINOv2, Predator), depend on contextual cues (2D matchers, Lepard), or miss small-object details (Qwen3-VL); all degrade significantly on predicted instances and lack mechanisms to leverage multi-view information. Lookalike3D instead integrates fine-grained shape and appearance across views, remaining robust across all three pair categories. Additionally, we evaluate generalization on a pseudo-GT dataset of identical and different samples based on ScanNet [10] in Tab. 3 (details in appendix, Sec. C.6). Lookalike3D outperforms all baselines. Qualitative results are shown in Fig. 4. Lookalike3D correctly recognizes identical groups of objects even in cluttered scenes with occlusion, while baselines tend to classify these as similar or different.

Method	ScanNet GT Inst.		
	Id.	Diff.	Overall
Lepard [35]	0.41	0.49	0.45
Predator [36]	0.71	0.17	0.44
DINOv2-feats [12]	0.32	0.52	0.42
SuperGlue [29]	0.41	0.70	0.55
ASpanformer [30]	0.65	0.69	0.67
Mast3r [31]	0.45	0.78	0.61
TLFR [32]	0.35	0.61	0.48
Qwen3-VL [57]	0.55	0.65	0.60
Lookalike3D	**0.79**	**0.85**	**0.82**

Table 3: Evaluation on ScanNet GT instances. Best in **bold**, second best underlined.

5.3 Downstream Applications

5.3.1 3D Object Reconstruction

We jointly reconstruct the identical pairs from Lookalike3D using SAM 3D. A qualitative comparison is shown in Fig. 5a. While SAM 3D produces plausible reconstructions of identical objects, these vary in shape due to limited scene context. Lookalike3D enables SAM 3D to produce consistent reconstructions across identical objects.

Method	CD↓	NC↑	F@1↑	F@5↑
SAM 3D (individual)	0.0763	0.528	0.164	0.452
SAM 3D + Lookalike3D (single)	0.0746	0.531	0.168	0.458
SAM 3D + Lookalike3D (multi)	**0.0484**	**0.548**	**0.179**	**0.528**

Table 4: Reconstruction quality of SAM 3D individual vs. SAM 3D + Lookalike3D joint reconstruction on ScanNet++. Best in **bold**.

Quantitative results on 28 ScanNet++ scenes (348 instances, 77 groups) against the ground truth point cloud are in Tab. 4 (details in appendix, Sec. D.1). Joint reconstruction using Lookalike3D achieves lower Chamfer distance, normal consistency and higher F-scores.

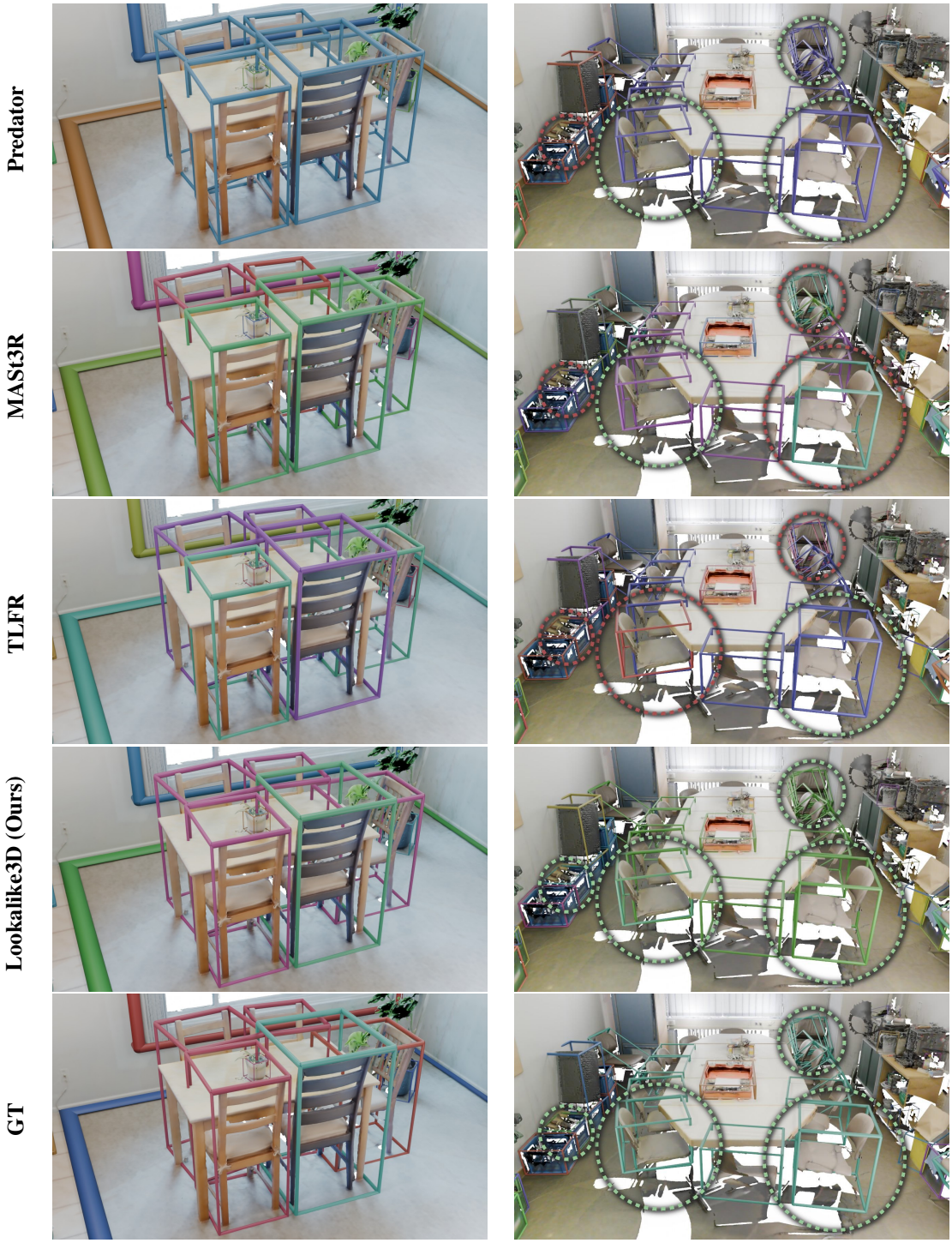

Figure 4: Identical object groups identified by Lookalike3D and baselines. Each group is indicated by a unique color which is randomized across scenes and methods. Correct and incorrect predictions are circled in green and red, respectively (right column). Unlike baselines that frequently misclassify or over-group instances, Lookalike3D leverages multi-view images and fine-grained appearance for accurate object grouping.



(a) Joint 3D reconstruction using SAM 3D on Lookalike3D outputs. Individual reconstructions vary in shape and size, while joint prediction fuses cues across instances into a single consistent reconstruction. Insets show original viewpoints.



(b) Part segmentation transfer across similar objects. P3-SAM undersegments objects with ambiguous geometry (e.g., fused chair arms and legs); we use similar pairs from Lookalike3D to transfer segmentation from a well-segmented source to the target, yielding more detailed results.

Figure 5: Downstream applications of Lookalike3D.

5.3.2 Part Co-segmentation

We apply predicted similar pairs to 3D part segmentation using P3-SAM. While P3-SAM produces accurate parts, it undersegments objects with ambiguous geometry, such as fused chair legs and arms. We transfer part segmentation from a source object with clearer geometry to a similar target object. A qualitative comparison of individual and joint part segmentation is shown in Fig. 5b.

Method	cIoU ↑	fIoU ↑	Acc ↑	BF ↑	PCE ↓
P3-SAM (individual)	0.43	0.29	0.53	0.09	3.47
P3-SAM + Ours	**0.50**	**0.36**	**0.65**	**0.11**	**2.61**

Table 5: Part segmentation of P3-SAM individual vs. P3-SAM + Lookalike3D. cIoU/fIoU: coarse/fine mIoU, Acc: point accuracy, BF: part boundary F-score, PCE: part count error. Best in **bold**.

By leveraging similar object pairs we can improve part segmentation on undersegmented objects. We also evaluate quantitatively on 36 objects common to 3DTwins and the Articulate3D [60] dataset, shown in Tab. 5. Fine mIoU is evaluated on the original annotations, while coarse mIoU is from merging smaller parts together as P3-SAM predicts limited small parts. We also measure the accuracy of labels per point, F-Score on the part boundary and the error in the number of predicted parts.

5.4 Ablation Studies

We ablate the different components of our method to show their contribution towards the full model in Tab. 6. Each row is ablated independently from the full model by changing only one hyperparameter or model component. All ablations are run with GT instances.

Model/Data	Id.	Sim.	Diff.	Overall
Triplet → InfoNCE loss	0.36	0.14	0.56	0.35
w/o alignment loss	0.63	0.14	0.65	0.47
w/o DINO aggr. feat.	0.62	0.09	0.58	0.43
Single-view attn. only	0.62	0.08	0.60	0.43
Single-view & object-level attn.	0.62	0.07	0.63	0.44
Single-view & global attn.	0.62	0.12	0.61	0.45
1 input view	0.60	0.13	0.55	0.43
3 input views	**0.66**	0.14	0.65	0.48
Full model (*Ours*)	0.65	**0.18**	**0.65**	**0.49**

Table 6: Ablation study, IoU per pair type (identical, similar, different, overall). The full model uses triplet and alignment losses, aggregated DINO features, single-view, multiview, and global attention, and 5 input views.

Effect of alternating attention layers Lookalike3D aggregates information across multiple views of an object via a hierarchical attention (image-object-global). Removing any level primarily degrades performance on the *similar* class (Tab. 6), as identifying near-identical objects requires comparing minor details across all views.

Effect of each loss function Our triplet loss compares distances between identical, similar, and different pairs, while the alignment loss grounds predictions relative to classification thresholds. Unlike InfoNCE [61], which lacks explicit margins and causes class overlap, the triplet loss effectively separates all three categories (Tab. 6). The alignment loss further pushes Id and Diff scores into their respective bins, indirectly aligning Sim.

Effect of increasing input views Lookalike3D leverages multiple views of the input pair for fine-grained shape and appearance cues. Increasing the number of views from $1 \rightarrow 3 \rightarrow 5$ increases IoU, particularly on the similar category (Tab. 6).

Effectiveness of aggregated DINOv2 features DINOv2 intermediate layers provide fine-grained structural features [52] needed to distinguish objects; using only last-layer semantic features reduces performance on the similar and different classes (Tab. 6).

Multi-annotator agreement To test the consistency of the annotation process, we re-annotated 23 scenes (1513 pairs) with four new annotators, of which 1016 pairs were labeled by two or more annotators. We obtain a Cohen's κ of 0.83 (almost-perfect) and Fleiss' κ of 0.85 with over 90% raw agreement between the annotations, so the annotation is reproducible and not specific to the single annotator. We re-evaluated our best model on this subset against both label sets (new labels aggregated by majority vote), on GT and predicted instances. The IoU changed by only 0.012 overall, indicating the original annotations are consistent with our guidelines.

Confusion matrix and analysis We show the row-normalized confusion matrix of predictions in Tab. 7. Similar pairs are more likely to be confused with the identical class, since they often differ in small details such as the logo on a monitor, text on a poster, or articulated segments of a table lamp. For this reason, we believe that similar pairs are genuinely challenging to detect from image features alone, including the DINOv2 features our method builds on.

	Predicted		
GT	Id.	Sim.	Diff.
Id.	**0.83**	0.08	0.10
Sim.	0.41	**0.26**	0.33
Diff.	0.16	0.06	**0.78**

Table 7: Confusion matrix of Lookalike3D predictions.

Limitations Our method's performance is limited by the patch resolution of DINOv2 features on small or distant objects, and by camera coverage on objects with subtle distinguishing details.

Broader impact Our work targets better understanding and generation of indoor 3D scenes, enabling research in 3D computer vision and assistive technologies for embodied agents and robots. We do not expect any negative environmental, societal, or ethical effects.

6 Conclusion

We introduced lookalike object detection in indoor scenes, the Lookalike3D model for similarity learning, and the 3DTwins dataset. Our method leverages complementary geometry and appearance cues from repeated instances to improve joint 3D reconstruction and part co-segmentation. We hope Lookalike3D inspires further research into object-level scene understanding.

Acknowledgments

This project was supported by the ERC Starting Grant SpatialSem (101076253).

References

[1] Xingyu Chen, Fu-Jen Chu, Pierre Gleize, Kevin J Liang, Alexander Sax, Hao Tang, Weiyao Wang, Michelle Guo, Thibaut Hardin, Xiang Li, et al. Sam 3d: 3dfy anything in images. *arXiv preprint arXiv:2511.16624*, 2025.

[2] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21469–21480, 2025.

[3] Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, Ruicheng Wang, Zelong Lv, Yu Deng, Hongyuan Zhu, Yue Dong, Hao Zhao, Nicholas Jing Yuan, and Jiaolong Yang. Native and compact structured latents for 3d generation. *Tech report*, 2025.

[4] Minghua Liu, Mikaela Angelina Uy, Donglai Xiang, Hao Su, Sanja Fidler, Nicholas Sharp, and Jun Gao. Partfield: Learning 3d feature fields for part segmentation and beyond. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9704–9715, 2025.

[5] Changfeng Ma, Yang Li, Xinhao Yan, Jiachen Xu, Yunhan Yang, Chunshi Wang, Zibo Zhao, Yanwen Guo, Zhuo Chen, and Chunchao Guo. P3-sam: Native 3d part segmentation. *arXiv preprint arXiv:2509.06784*, 2025.

[6] Chuhao Chen, Isabella Liu, Xinyue Wei, Hao Su, and Minghua Liu. Freeart3d: Training-free articulated object generation using 3d diffusion. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, pages 1–13, 2025.

[7] Ruining Li, Yuxin Yao, Chuanxia Zheng, Christian Rupprecht, Joan Lasenby, Shangzhe Wu, and Andrea Vedaldi. Particulate: Feed-forward 3d object articulation. *arXiv preprint arXiv:2512.11798*, 2025.

[8] Daoyi Gao, Yawar Siddiqui, Lei Li, and Angela Dai. Meshart: Generating articulated meshes with structure-guided transformers. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2025.

[9] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12–22, 2023.

[10] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017.

[11] Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Tal Dimry, Yuri Feigin, Peter Fu, Thomas Gebauer, Brandon Joffe, Daniel Kurz, Arik Schwartz, et al. Arkitscenes: A diverse real-world dataset for 3d indoor scene understanding using mobile rgb-d data. *arXiv preprint arXiv:2111.08897*, 2021.

[12] 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.

[13] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5294–5306, 2025.

[14] Xinchen Liu, Wu Liu, Tao Mei, and Huadong Ma. A deep learning-based approach to progressive vehicle re-identification for urban surveillance. In *European conference on computer vision*, pages 869–884. Springer, 2016.

[15] Hongye Liu, Yonghong Tian, Yaowei Wang, Lu Pang, and Tiejun Huang. Deep relative distance learning: Tell the difference between similar vehicles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2167–2175, 2016.

[16] Ming Li, Xinming Huang, and Ziming Zhang. Self-supervised geometric features discovery via interpretable attention for vehicle re-identification and beyond. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 194–204, 2021.

[17] Dechao Meng, Liang Li, Xuejing Liu, Yadong Li, Shijie Yang, Zheng-Jun Zha, Xingyu Gao, Shuhui Wang, and Qingming Huang. Parsing-based view-aware embedding network for vehicle re-identification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7103–7112, 2020.

[18] Liping Lu, Zihao Fu, Duanfeng Chu, Wei Wang, and Bingrong Xu. Clip-senet: Clip-based semantic enhancement network for vehicle re-identification. *IEEE Transactions on Intelligent Transportation Systems*, 27(1):1267–1278, 2025.

[19] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017.

[20] Nicolai Wojke and Alex Bewley. Deep cosine metric learning for person re-identification. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 748–756. IEEE, 2018.

[21] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *Proceedings of the IEEE international conference on computer vision*, pages 1116–1124, 2015.

[22] Zhizhong Huang and Xiaoming Liu. Generalizable object re-identification via visual in-context prompting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22539–22550, 2025.

[23] Vincent Cartillier, Neha Jain, and Irfan Essa. 3d semantic mapnet: Building maps for multi-object re-identification in 3d. *arXiv preprint arXiv:2403.13190*, 2024.

[24] Liyuan Zhu, Shengyu Huang, Konrad Schindler, and Iro Armeni. Living scenes: Multi-object relocalization and reconstruction in changing 3d environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 28014–28024, 2024.

[25] Emily Steiner, Jianhao Zheng, Henry Howard-Jenkins, Chris Xie, and Iro Armeni. Rescene4d: Temporally consistent semantic instance segmentation of evolving indoor 3d scenes. *arXiv preprint arXiv:2601.11508*, 2026.

[26] Johanna Wald, Armen Avetisyan, Nassir Navab, Federico Tombari, and Matthias Nießner. Rio: 3d object instance re-localization in changing indoor environments. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7658–7667, 2019.

[27] Jonas Schult, Francis Engelmann, Alexander Hermans, Or Litany, Siyu Tang, and Bastian Leibe. Mask3d: Mask transformer for 3d semantic instance segmentation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8216–8223. IEEE, 2023.

[28] 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.

[29] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020.

[30] Hongkai Chen, Zixin Luo, Lei Zhou, Yurun Tian, Mingmin Zhen, Tian Fang, David Mckinnon, Yanghai Tsin, and Long Quan. Aspanformer: Detector-free image matching with adaptive span transformer. In *European conference on computer vision*, pages 20–36. Springer, 2022.

[31] Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In *European conference on computer vision*, pages 71–91. Springer, 2024.

[32] Junyi Zhang, Charles Herrmann, Junhwa Hur, Eric Chen, Varun Jampani, Deqing Sun, and Ming-Hsuan Yang. Telling left from right: Identifying geometry-aware semantic correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3076–3085, 2024.

[33] Philippe Weinzaepfel, Thomas Lucas, Vincent Leroy, Yohann Cabon, Vaibhav Arora, Romain Brégier, Gabriela Csurka, Leonid Antsfeld, Boris Chidlovskii, and Jérôme Revaud. Croco v2: Improved cross-view completion pre-training for stereo matching and optical flow. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17969–17980, 2023.

[34] Honggyu An, Jin Hyeon Kim, Seonghoon Park, Jaewoo Jung, Jisang Han, Sunghwan Hong, and Seungryong Kim. Cross-view completion models are zero-shot correspondence estimators. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 1103–1115, 2025.

[35] Yang Li and Tatsuya Harada. Lepard: Learning partial point cloud matching in rigid and deformable scenes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5554–5564, 2022.

[36] 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*, pages 4267–4276, 2021.

[37] Benjamin Thérien, Chengjie Huang, Adrian Chow, and Krzysztof Czarnecki. Object re-identification from point clouds. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 8377–8388, 2024.

[38] 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*, pages 6677–6686, 2022.

[39] Zhiqin Chen, Kangxue Yin, Matthew Fisher, Siddhartha Chaudhuri, and Hao Zhang. Bae-net: Branched autoencoder for shape co-segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8490–8499, 2019.

[40] Chenyang Zhu, Kai Xu, Siddhartha Chaudhuri, Li Yi, Leonidas J Guibas, and Hao Zhang. Adacoseg: Adaptive shape co-segmentation with group consistency loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8543–8552, 2020.

[41] Yawar Siddiqui, Duncan Frost, Samir Aroudj, Armen Avetisyan, Henry Howard-Jenkins, Daniel DeTone, Pierre Moulon, Qirui Wu, Zhengqin Li, Julian Straub, et al. Shaper: Robust conditional 3d shape generation from casual captures. *arXiv preprint arXiv:2601.11514*, 2026.

[42] Mingyu Dong, Chong Xia, Mingyuan Jia, Weichen Lyu, Long Xu, Zheng Zhu, and Yueqi Duan. Replicateanyscene: Zero-shot video-to-3d composition via textual-visual-spatial alignment. *arXiv preprint arXiv:2604.10789*, 2026.

[43] Chong Xia, Kai Zhu, Zizhuo Wang, Fangfu Liu, Zhizheng Zhang, and Yueqi Duan. Simrecon: Simready compositional scene reconstruction from real videos. *arXiv preprint arXiv:2603.02133*, 2026.

[44] Armen Avetisyan, Christopher Xie, Henry Howard-Jenkins, Tsun-Yi Yang, Samir Aroudj, Suvam Patra, Fuyang Zhang, Duncan Frost, Luke Holland, Campbell Orme, et al. Scenescript: Reconstructing scenes with an autoregressive structured language model. In *European Conference on Computer Vision*, pages 247–263. Springer, 2024.

[45] Qirui Wu, Yawar Siddiqui, Duncan Frost, Samir Aroudj, Armen Avetisyan, Richard Newcombe, Angel X Chang, Jakob Engel, and Henry Howard-Jenkins. Jrm: Joint reconstruction model for multiple objects without alignment. *arXiv preprint arXiv:2603.25985*, 2026.

[46] Paul Dobre, Xin Wang, and Hongzhou Yang. Furnset: Exploiting repeats for 3d scene reconstruction. *arXiv preprint arXiv:2604.20093*, 2026.

[47] Nicolás Violante, Andreas Meuleman, Alban Gauthier, Fredo Durand, Thibault Groueix, and George Drettakis. Splat and replace: 3d reconstruction with repetitive elements. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pages 1–12, 2025.

[48] Tianhang Cheng, Wei-Chiu Ma, Kaiyu Guan, Antonio Torralba, and Shenlong Wang. Structure from duplicates: Neural inverse graphics from a pile of objects. *arXiv preprint arXiv:2401.05236*, 2024.

[49] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[50] Mi Yan, Jiazhao Zhang, Yan Zhu, and He Wang. Maskclustering: View consensus based mask graph clustering for open-vocabulary 3d instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 28274–28284, 2024.

[51] Junyi Zhang, Charles Herrmann, Junhwa Hur, Luisa Polania Cabrera, Varun Jampani, Deqing Sun, and Ming-Hsuan Yang. A tale of two features: Stable diffusion complements dino for zero-shot semantic correspondence. *Advances in Neural Information Processing Systems*, 36:45533–45547, 2023.

[52] Shir Amir, Yossi Gandelsman, Shai Bagon, and Tali Dekel. Deep vit features as dense visual descriptors. *arXiv preprint arXiv:2112.05814*, 2(3):4, 2021.

[53] Haotong Lin, Sili Chen, Junhao Liew, Donny Y Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views. *arXiv preprint arXiv:2511.10647*, 2025.

[54] Qing Wang, Jiaming Zhang, Kailun Yang, Kunyu Peng, and Rainer Stiefelhagen. Matchformer: Interleaving attention in transformers for feature matching. In *Proceedings of the Asian conference on computer vision*, pages 2746–2762, 2022.

[55] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[56] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[57] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[58] Minghua Liu, Ruoxi Shi, Kaiming Kuang, Yinhao Zhu, Xuanlin Li, Shizhong Han, Hong Cai, Fatih Porikli, and Hao Su. Openshape: Scaling up 3d shape representation towards open-world understanding. *Advances in neural information processing systems*, 36:44860–44879, 2023.

[59] Le Xue, Ning Yu, Shu Zhang, Artemis Panagopoulou, Junnan Li, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, et al. Ulip-2: Towards scalable multimodal pre-training for 3d understanding. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 27081–27091. IEEE, 2024.

[60] Anna-Maria Halacheva, Yang Miao, Jan-Nico Zaech, Xi Wang, Luc Van Gool, and Danda Pani Paudel. Articulate3d: Holistic understanding of 3d scenes as universal scene description. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5633–5644. IEEE, 2025.

[61] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[62] Lu Qi, Jason Kuen, Weidong Guo, Tiancheng Shen, Jiuxiang Gu, Jiaya Jia, Zhe Lin, and Ming-Hsuan Yang. High-quality entity segmentation. *arXiv preprint arXiv:2211.05776*, 2022.

Appendix

A Comparison with Single-view Baselines

The 2D matching methods that we compare against are designed to operate in a single view setting, hence we report their single view results on the image with the most coverage, compared with our full method in Tab. 8. Single-view matching methods capture limited object information, while Lookalike3D effectively combines multiview information and outperforms all baselines.

		GT Instances				Pred. Instances			
	Method	Id.	Sim.	Diff.	Overall	Id.	Sim.	Diff.	Overall
2D Methods	SuperGlue [29]	0.05	0.07	0.44	0.19	0.02	0.05	0.24	0.10
	ASpanformer [30]	0.07	0.06	0.43	0.19	0.04	0.03	0.24	0.10
	Mast3r [31]	0.16	0.07	0.45	0.23	0.10	0.06	0.24	0.13
	TLFR [32]	0.25	0.11	0.33	<u>0.23</u>	0.18	0.11	0.17	0.15
	Qwen3-VL [57]	0.28	0.15	0.65	0.36	0.27	0.1	0.26	0.21
Ours	Lookalike3D	**0.65**	**0.18**	**0.65**	**0.49**	**0.50**	**0.15**	**0.29**	**0.31**

Table 8: Lookalike3D compared with 2D matching baselines evaluated on single images using IoU (%) for Identical (Id.), Similar (Sim.), and Different (Diff.) pairs. We report results on both Ground Truth (GT) instances and Predicted instances to evaluate robustness. The best results are **bolded** and the second best are <u>underlined</u>.

B Dataset Details

B.1 Data Preparation

We rasterize the ScanNet++ [9] 3D instance annotations onto the undistorted DSLR images to obtain high-quality object crops. We calculate the visibility of an object o in image I as the fraction of the total vertices on o that are visible in I, and use the top 10 crops for each object during annotation. Object pairs are created within semantic classes. Pairs are created using only the objects in the same scene. We generate all intra-class object pairs, excluding structural elements (walls, floors), small items (pens, files), and objects without distinct 3D segmentation (pipes).

B.2 Annotation Interface

We built a web annotation interface where an annotator is shown pairs of objects and can classify each pair as identical, similar or different, and additionally select similarity types for similar pairs (described below). An example pair is shown in Fig. 6. Identical objects are iteratively grouped together; initially each object is in its own group and identical pairs are accumulated together by merging groups, which greatly reduces the time required to annotate them. Blurred or unclear pairs can be marked as unknown. The annotator can also view the full scene context of the object, which is often helpful to view repeated objects that are close together in the scene.

The annotator can also view the 3D mesh of the object as shown in Fig. 7, which is useful when the shape or dimensions of the objects are not clear from the images.

B.3 Similarity Types

We annotated each similar object pair with one or more *similarity types*, indicating in what aspects the objects differ. Possible values for the similarity type are:

1. *shape*: objects with similar shapes, and possibly different textures
2. *articulated*: identical objects in different states of articulation, such as rotated office chairs and monitors adjusted to different heights
3. *mirrored*: identical objects that are left-right mirrored and hence cannot be rigidly transformed to superimpose one on the other

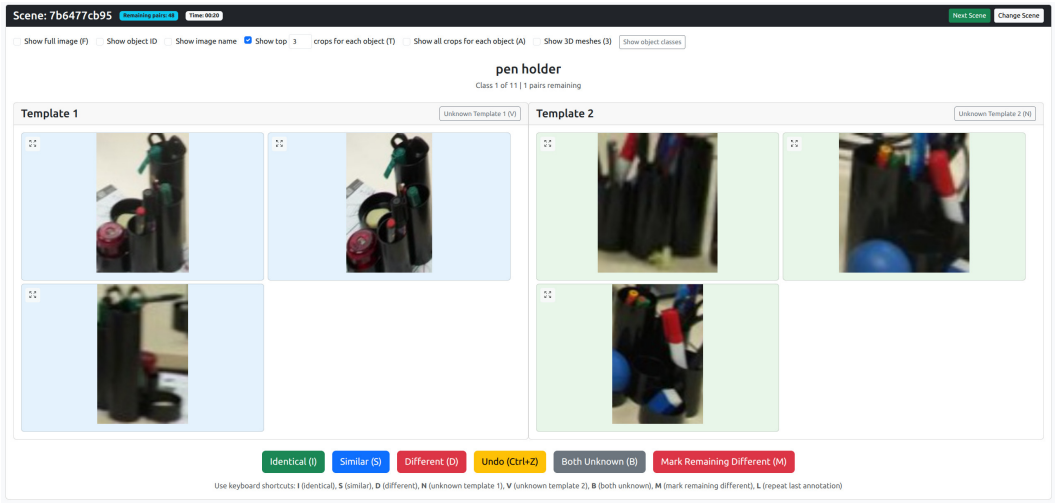

Figure 6: Web interface of our annotation tool. The annotator has access to multiple views of each object as well as their full views for scene context. Here we show the top 3 views of each object by coverage.

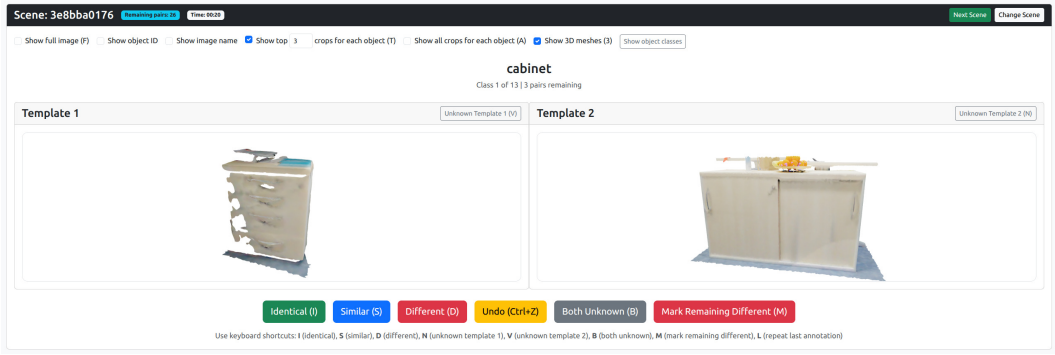

Figure 7: 3D view of object meshes during annotation. Note that 3D information is only available during annotation and not during model training.

4. *deformable*: identical objects that are deformed into different 3D shapes, such as pillows and blankets
5. *texture/color*: objects with identical shapes and different textures or colors.

B.4 Annotation Guidelines

We ensure the quality of our annotations through the following annotation guidelines. *Identical pairs* must be unambiguously identified through the 2D images, or verified using the 3D mesh when required. *Similar pairs* must share the semantic structure, such as a pair of chairs which both have rotating wheels, or a pair of height adjustable tables at different heights, articulated table lamps, trash cans of different colors, pillows with different deformations, and have at least 90% shape overlap. *Different pairs* do not have shared semantic structure and differ significantly, such as a wooden dining chair without arms, and a leather office chair with wheels and arms. We first annotated 50 randomly selected scenes to build the annotation guidelines, and then annotated from scratch according to the guidelines to create the dataset of 856 train and 50 validation scenes based on ScanNet++.

C Implementation Details

C.1 Model Details

We use a DINOv2-small model which remains frozen throughout training. Intermediate DINO features from layers $[1, 3, 5, 8]$ and the last layer features are concatenated together along the feature

dimension and then projected to the original feature dimension of 384 before being passed to the alternating attention layers.

Our model consists of one alternating attention block, which contains 1 layer each of single-view, multiview and global attention. The attention layers use an internal embedding dimension of 256, with 8 attention heads. We add cosine positional embeddings to the patch features at each layer in our model.

C.2 Lookalike3D Training

We use the Adam optimizer with a learning rate of 0.0001. We apply random rotation, horizontal flip, color jitter, and crop augmentations during training. Images are padded to a square shape and then resized to the 224×224 input dimensions of DINO. We sample batches to maximize the number of similar and different pairs for a given anchor identical pair, and train with a batch size of at most 128 pairs. Training takes about 4 hours on an Nvidia A6000 GPU.

We set the triplet loss margins α_i to $(0.4, 0.4, 0.8)$ for Id-Sim, Sim-Diff and Id-Diff respectively. The triplet loss requires each identical anchor pair to have a corresponding negative (similar or different) pair such that their similarities can be optimized jointly. Since each annotated identical pair does not necessarily have a similar or different pair with a common anchor object, we randomly sample *different* pairs to create a more complete dataset. For an annotated identical pair of objects (a, b) from semantic class S, we sample additional pairs (a, c) and (b, d) where $c, d \notin S$.

C.3 Baselines

For each baseline, we set two thresholds t_1, t_2 from the histogram of predicted match counts so that at least 95% of image pairs fall within range, and keep only matches with at least 70% confidence where applicable; these are analogous to our similarity thresholds $t_1 = 0.33, t_2 = 0.66$.

C.4 MaskClustering Instance Segmentation

We obtain class-agnostic instance predictions from MaskClustering, by combining Cropformer [62] masks from multiview DSLR images subsampled by a factor of 5. Predicted instances are associated with GT instances based on a minimum IoU threshold on 0.5.

C.5 Evaluation

We evaluate intersection-over-union (IoU) separately for each of three pair types, and over all pair types. Evaluation on GT instances allows us to separate out the accuracy of instance prediction from the accuracy of Lookalike3D.

When evaluating on predicted instances, we rasterize the predicted instance segmentation onto DSLR images and create crops as in the GT setting. GT pairs that are missing in the predictions are classified into an *unknown* class, hence including the effect of instance segmentation on lookalike object detection.

C.6 ScanNet evaluation

To assess cross-dataset generalization we construct a pair-classification benchmark over the ScanNet validation split. We select 50 scenes from the ScanNet validation set, and use the per-frame 2D instance masks, posed RGB frames (`scannet_frames_25k`), and the 3D mesh for each scene. For every annotated object instance, we extract a 3D point cloud and a set of 2D image crops by tightly bounding the object's pixels in each frame where its instance mask is visible.

Object-pair samples are then assembled into two classes. *Identical* pairs consist of distinct camera views of the *same* physical object within a single scene; for each instance we draw five distinct viewpoints per side for multi-view evaluation. When fewer views are available, we repeat the input views. *Different* pairs consist of two object instances that share the *same* semantic label but originate from *different ScanNet scenes*, where scene identity is the `sceneXXXX` prefix (so re-scans such as `sceneXXXX_00` and `sceneXXXX_01` are treated as the same scene and are excluded from cross-room pairing). This avoids the trivial case of distinguishing objects of different classes, and isolates the harder problem of telling apart genuinely distinct physical instances of the same category. The final

benchmark contains $1{,}588$ identical and $1{,}588$ different pairs ($3{,}176$ total) drawn from 37 distinct semantic labels.

Since Lookalike3D outputs identical, similar and different labels for pairs of objects while the ScanNet evaluation dataset has only identical and different samples, we map predicted similar labels to either different or identical using a 0.5 similarity threshold.

D SAM 3D Joint Reconstruction Details

Given identical 2D input objects, SAM 3D Objects produces 3D reconstructions that vary in shape and scale, making them inconsistent. We make use of predicted identical object pairs to jointly predict a single clean reconstruction that combines aspects of each individual object, by conditioning SAM 3D on a single image input and multiple object masks.

Specifically, we use a single RGB image I containing all of the identical objects as input to SAM 3D, and the masks M_i of each of these objects in I. Each object may have a different degree of visibility; our joint reconstruction method combines these together into a single coherent 3D object.

SAM 3D consists of a geometry model that predicts the initial dense object shape, and a texture and refinement model that refines the dense shape into a set of sparse voxels and predicts object texture.

Geometry Model The geometry model denoises a set of layout tokens into the object rotation, translation and scale, and a 16^3 set of layout tokens densely representing the low resolution shape of the object. It consists of a sparse structure generator and a sparse structure decoder. The sparse structure generator is conditioned on I and M_i through cross-attention. Since the model originally conditions on a single object mask, we inject multiple masks M_i of objects o_i into each flow transformer layer by conditioning on each M_i separately. This gives intermediate features $\{f_{l,i}\}$ corresponding to each mask in each layer l. We average over these features to obtain a single intermediate representation f_l that represents all the objects o_i in the canonical space, and use this as input to the $(l+1)^{th}$ layer. The final output of the sparse structure generator is a $16^3 \times 8$ feature vector representing the joint low-resolution shape of the object. This is upsampled to 64^3 using the existing sparse structure decoder. The layout tokens are similarly decoded through the layout decoder, and the final outputs of the geometry model are a single rotation, scale and translation, and the single 64^3 high-resolution geometry of the jointly reconstructed object.

Texture and Refinement Model The multi-mask conditioning in the texture and refinement model is carried out in a similar manner. The high-resolution 64^3 shape is converted to sparse coordinates which are denoised to SLAT (structured latent) features. The input to the SLAT sampler is a set of noise tokens at these sparse coordinates. The SLAT sampler is then conditioned on M_i and I, by injecting these as conditions into every layer of the sparse flow transformer and averaging the intermediate outputs. The output is a single set of SLAT features. These SLAT features are decoded with the SLAT decoder to obtain the mesh of the jointly reconstructed object.

In this way, we adapt SAM 3D to generate a single 3D object representing a combination of the objects o_i, effectively combining appearance and shape cues from identical objects with varying degrees of occlusion.

We use the median scale of the individually predicted objects as the scale of the jointly reconstructed object, and apply the rotation and translation of the individual objects to the joint object to transform them in the scene. Hence, the outputs from our joint reconstruction do not vary either in shape or scale, while improving upon the 3D geometry of the individual reconstructions.

D.1 Evaluation

We construct the per-group consensus ground truth by SDF averaging. For each (scene, image) group, we extract the per-instance GT sub-meshes from the labelled ScanNet++ scan (faces whose vertices all carry the object's instance id), center each to its origin, and ICP-align (point-to-point, rigid) every other instance to the first. We then evaluate signed distances of all aligned meshes on a shared 64^3 voxel grid covering their union AABB, average the resulting SDFs, and recover the consensus surface via marching cubes. Predictions are likewise mean-centered and ICP-aligned to the consensus before metrics are computed, isolating shape quality from pose and per-instance scan-coverage noise. The

same SDF-averaging operation is applied on both the GT side (over per-instance GT meshes) and on our prediction side (over per-instance SAM 3D outputs, yielding the joint mesh), making the comparison apples-to-apples.

E Part Co-segmentation

Alignment using point-to-point ICP between the correctly segmented source and the undersegmented objects is typically stable since SAM 3D outputs canonical coordinates. In case of alignment failures, we fall back to binned vertical-axis rotations as initialization for ICP.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract describes the task, similarity learning method, dataset, and downstream applications in joint 3D reconstruction and part segmentation, which are the main contributions of this paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations and failure cases of this work are discussed in the Experiments section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: There are no theoretical proofs in the paper.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The method is described in detail, and further implementation details are provided in the main paper as well as in the supplementary material.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The proposed dataset is included in the submission, and code will be released upon acceptance.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental setting is described in the Experiments section and under Implementation Details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Due to the computational cost of training and the deterministic nature of the evaluation, as well as following previous related work which reports results without error bars, we report results from a single run.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The hardware used and runtime are described in the Implementation Details section in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We confirm that the paper conforms with the NeurIPS Code of Ethics. We do not capture any human data, and the datasets used are publicly available and widely used in this field of research.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: These are discussed under 'Broader impact' in the Experiments section.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not anticipate any risks from the release of our code, data and models.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Our dataset is based on ScanNet++, and our model is based on existing works, all of which have been referenced. Their licenses are properly respected.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Our dataset is included in the submission, and contains sufficient documentation to use it along with existing publicly available documentation of the ScanNet++ dataset.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were not used for the core methodology of this paper.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.