Supplementary for Material Magic Wand: Material-Aware Grouping of 3D Parts in Untextured Meshes

A. Part Segmentation

We obtain parts by computing connected components of each mesh after merging duplicated vertices (vertices having the exact same coordinates). This segmentation choice reflects how artist-created meshes naturally decompose into fine-grained, meaningful components. As shown in Figure 2, most segments exhibit clear structural or semantic meaning (e.g., individual wheel spikes, boat ropes, bedsheet fringes). We exclude scanned objects from our dataset since they do not exhibit such decomposition. However, our method could be applied to scanned objects given appropriate part segmentation. All baselines reported in the paper use the same set of parts.

B. Data balancing

Objaverse presents several challenges when used directly for supervision (Table Ablation results in the main paper). First, many meshes exhibit strong class imbalance, where a few material identities account for the majority of parts within a shape. Figure 3(a) shows such a case, where most parts share a single material label. Second, meshes with a large number of material identities often display a pronounced long-tail distribution, with part counts per material following a power-law decay. As shown in Figure 3(b), although the mesh contains 76 materials, most appear only in a few parts. Third, meshes with very large numbers of parts can dominate training if sampled naively. For example, Figure 3(d) shows a mesh with 56,194 parts, many of which are near-duplicate geometries. Such meshes inflate dataset size without contributing proportional diversity and can lead to dataset-level bias if each part is sampled uniformly.

To address geometric redundancy, we merge highly similar parts using a vertex-distribution histogram matching procedure (section C), yielding a more compact and diverse part set. We further apply data balancing strategies to mitigate both within and across mesh imbalance, resulting in a more uniform distribution over both meshes and material identities. To do so, we apply the following constraints while sampling:

- Each material identity within a mesh should have at least 8 training samples. If the number of deduplicated parts for a material is below 8, we first include additional instances of the same geometry from different locations in the mesh, as these occur under different spatial context and serve as natural augmentation. If this is still insufficient (e.g., Fig 3(c)), we render multiple additional viewpoints of the same part, treating each viewpoint as an additional training sample.
- We limit each material from each mesh to a maximum of

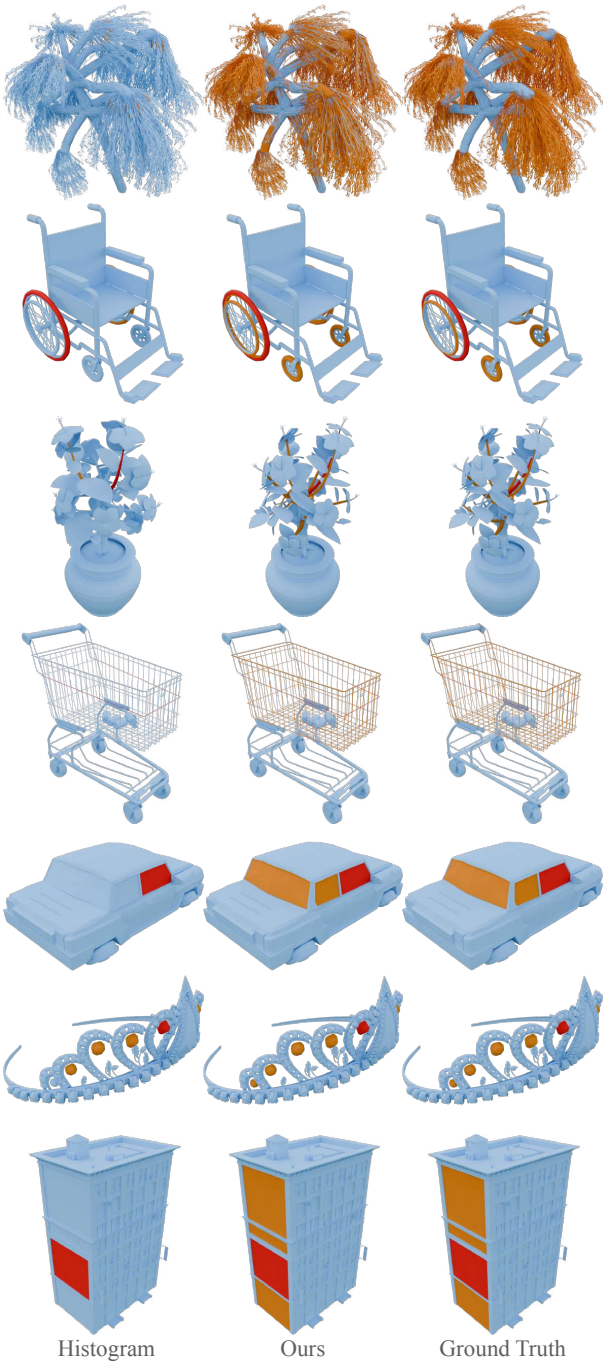

Figure 1. **Qualitative results of Histogram Matching approach.** Histogram matching successfully retrieves near-identical components (e.g., duplicated branches or similar jewels) but fails to capture structurally similar yet non-identical parts, such as the rear car window or varied tree leaves.

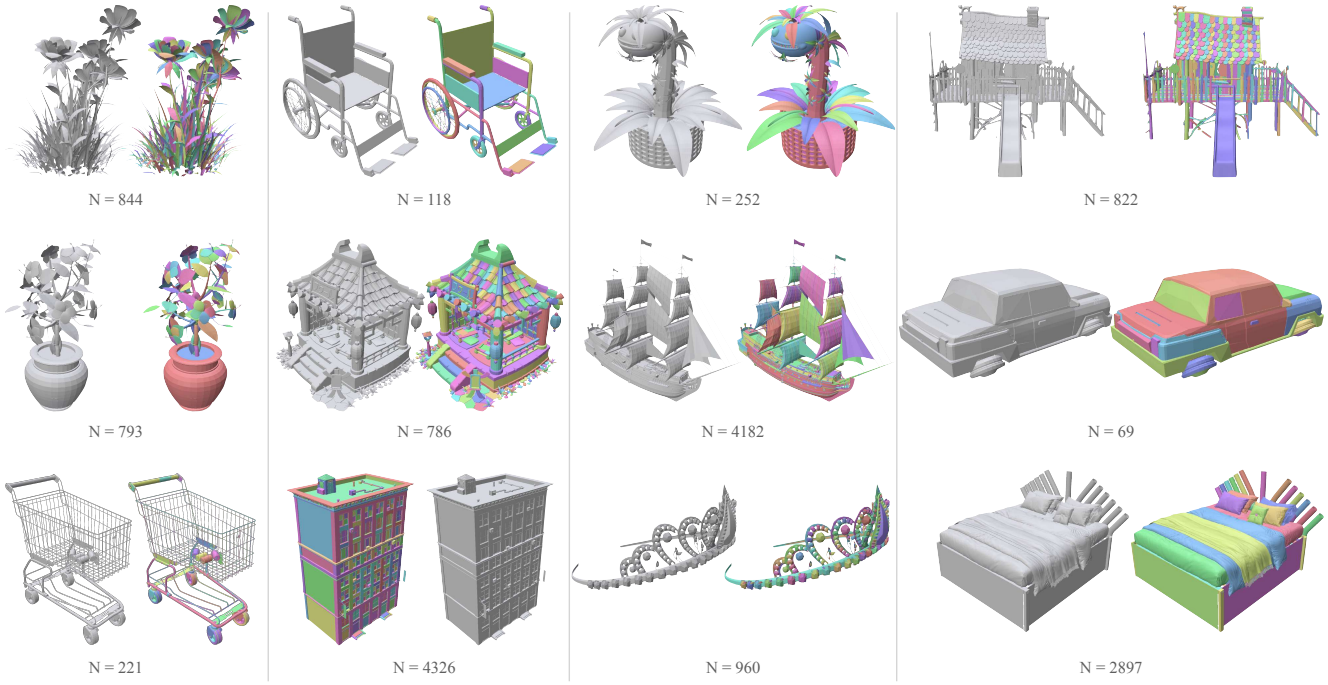

Figure 2. **Connected components as segments.** Examples of part segmentation produced using connected components. Each color corresponds to a different part. N denotes the number of parts in the mesh, illustrating the fine granularity of the decomposition.

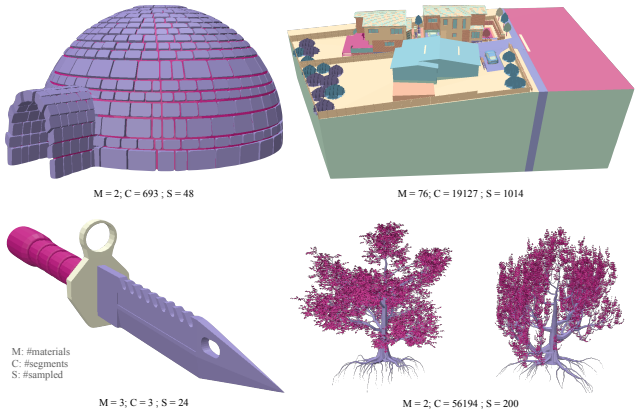

Figure 3. **Data sampling.** (M: #materials, C: #segments, S: #sampled after balancing). **Top-left (a):** Within-mesh imbalance where one material (purple bricks) dominates. **Top-right (b):** Long-tail distribution with 76 materials but most appearing in few parts. **Bottom-left (c):** Insufficient samples requiring multi-view augmentation. **Bottom-right (d):** Mesh with 56K several near-duplicate parts requiring capping.

100 samples across the training set to avoid overrepresentation of highly frequent materials.

- For each mesh, we constrain the ratio between the most frequent and least frequent material (in terms of part count) to be at most 5, preventing single dominant materials from overwhelming supervision.

We choose these hyperparameters heuristically to balance training scale with data diversity. After balancing, the final training set contains 1.9M part samples drawn from 22k meshes, and data generation takes approximately two days on 4 NVIDIA L40 GPUs.

C. Histogram matching

We use a geometry based approach [1, 2] to identify the near-duplicate parts within each mesh. For each part, we compute a histogram of vertex distances measured from the part's centroid, forming a fixed-length descriptor of its overall vertex distribution. Two parts are considered duplicates only if (i) their vertex counts differ by less than 5%, (ii) their histograms fall within a small ℓ_1 difference threshold, and (iii) their overall scales (maximum radial extent) agree within a relative tolerance. We use conservative thresholds (64 histogram bins, ℓ_1 threshold 10^{-2}, scale tolerance 10^{-2}) to minimize false positives. From each duplicate group, we retain a single representative part, except in cases where additional instances are intentionally reused for data augmentation as described in Section B. We show the qualitative results from Histogram Matching baseline in Figure 1. It successfully identifies near-identical parts but misses those with moderate geometric variation.

This deduplication step and learned embeddings serve different roles: histogram matching only collapses near-identical rigid-transformed duplicates to shrink the search

Table 1. Effect of deduplication on inference runtime.

Inference	#Parts†	Time(s)†	AUC ↑	mAP ↑
No Dedup	265; 1144.2	48.91; 208.05	88.13	89.68
Dedup	60; 181.0	11.14; 33.01	89.74	91.70

†Reported numbers are (Median; IQR) over full test set.

space, while embeddings retrieve geometrically varied parts with shared material. We apply deduplication during training to reduce redundancy and improve batch diversity, and at inference because rendering and encoding dominates the runtime, as discussed in section F. Table 1 shows that deduplication at inference yields a median $4.4\times$ and up to $520\times$ runtime speedup. Histogram matching itself accounts for only 0.16% of total *Dedup* time. The small accuracy drop without deduplication stems from increased chance of poor viewpoints; deduplication enables more stable canonical renderings. We use conservative thresholds, merging only essentially identical parts; a manual audit shows a 1.8% false-merge rate over 2349 groups across 141 meshes (Fig. 4), typically when identical structures are used across different materials. Importantly, the histogram-matching baseline remains far below our method in AUC-PR, confirming deduplication does not solve grouping task.

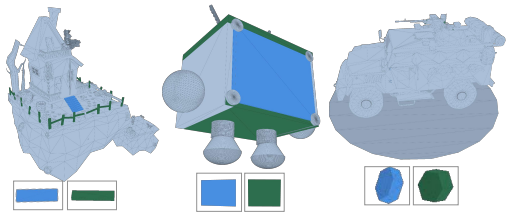

Figure 4. Deduplication error when using similar pieces across categories (e.g., cuboid for stairs and fence).

D. Additional results

D.1. Qualitative results on Objaverse

In Figure 5, we present additional results from the Objaverse test benchmark showing diverse structures that can be grouped based on material (ropes across the boat, ladder body, heterogeneous locomotive wheels, wheelchair frame), demonstrating robustness under structural differences.

D.2. Qualitative results on TexVerse

To evaluate generalization beyond Objaverse, we qualitatively test our method on meshes from TexVerse [3]. We select meshes that are artist-generated (not scanned), decomposable into connected components, and contain non-identical parts that share the same material, which narrows the selection pool.

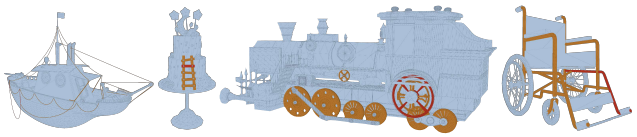

Figure 5. Additional qualitative results on Objaverse test set.

Figure 6 shows the qualitative results. We observe similar trends as in our test bench from Objaverse. Baselines often miss valid instances, such as non-identical window frames on the house, vertical masts on the boat, lower-floor stair steps, or roof logs, or retrieve unrelated structures such as bricks on the house, ropes on the boat, stair stringers, or rooftops in the huts. In contrast, our method retrieves these material-consistent parts across geometric and structural variations more reliably.

D.3. DINO variants

We finetune our supervised model from DINO-v3 *small* backbone. To test the effect of backbone scale, we also evaluate multiple DINO-v2 and DINO-v3 variants, as summarized in Table 2. While larger models offer marginal improvements in some metrics (e.g., mAP), we do not observe consistent gains across all evaluation criteria. Given its competitive performance and fewer parameters, we adopt DINO-v3 *small* in all our experiments.

E. Limitation and failure cases

Despite strong performance, our methods can face limitations:

Large geometric variation within the same material. Our model relies partly on geometric cues to infer material consistency. When parts share a material but differ significantly in shape, retrieval may be incomplete (Figure 7). For example, machinery cables that are folded and knotted at their endpoints appear geometrically distinct from their linear midsections. While our method successfully retrieves similar mid cable segments, it misses the endpoints.

Geometric similarity across different materials. In some cases, when parts from different materials share highly similar geometry and local context, our method can incorrectly group them (Figure 8). In a chess set, pieces from opposing teams have identical forms and differ only by material or color. Without sufficient training examples demonstrating such material distinctions in similar geometric contexts, the model groups them together.

Inherent ambiguity arising from artistic intent. Certain material assignments reflect artistic or stylistic decisions that cannot be inferred from geometry or spatial context (Figure 9). Visually similar cake toppings may be intentionally assigned different materials for aesthetic variation. Our

Table 2. Performance comparison of different DINO backbone variants on our material-aware part retrieval benchmark. Results show that scaling to larger models provides only marginal or inconsistent gains across metrics. All metrics in (%).

Method	**AUC PR**	**R-Prec**	**mAP**	**Recall@K**			
				R@5	**R@10**	**R@20**	**R@100**
Dino v2 (small)	80.13	77.33	82.23	33.39	45.14	57.03	81.60
Dino v2 (base)	78.99	76.24	81.98	33.37	45.10	56.73	81.00
Dino v2 (large)	78.28	74.41	80.59	33.94	44.70	55.92	81.00
Dino v2 (giant)	78.30	76.51	81.78	33.57	45.15	56.29	80.47
Dino v3 (small)	**81.14**	78.32	83.49	34.57	45.49	56.63	82.18
Dino v3 (base)	80.26	77.07	82.08	33.52	44.65	55.60	81.22
Dino v3 (large)	<u>80.86</u>	78.92	84.01	35.18	47.09	57.70	81.83
Dino v3 (huge)	79.50	78.03	83.11	33.76	45.11	56.89	80.98
Dino v3 (7B)	79.27	77.89	83.13	34.26	45.91	56.98	81.76

method retrieves similar parts based on learned patterns, but such artistic choices represent inherent ambiguity that cannot be resolved without additional user input.

We focus on designer-created meshes with reliable part structure; evaluation on emerging AI-generated assets is future work.

F. Runtime

For each mesh, rendering and encoding is a one-time cost. The representations can be reused for retrievals on the mesh.

Rendering time. Per-part rendering of $I^{part}, I^{ctx}, I^{full}$ takes 0.138 ± 0.016s.

Encoding time. For meshes with < 50, 50–200, and > 200 parts, the median encoding times with DINO-v3 *small* are 1.11s, 3.67s, and 22.78s respectively.

Retrieval time. For queries on meshes with < 50, 50–200, and > 200 parts, the retrieval times are 0.000638s, 0.001633s, and 0.010847s respectively.

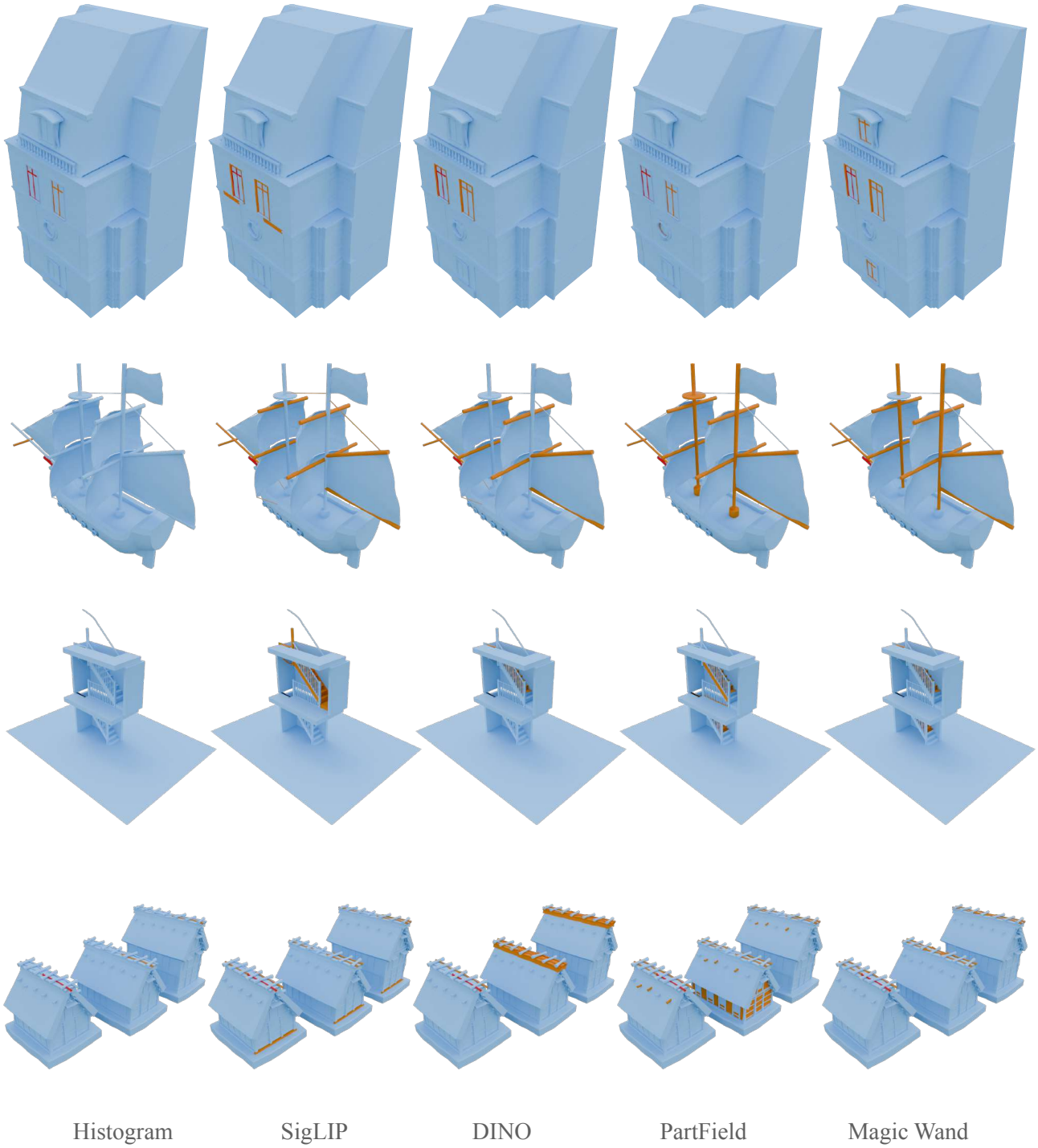

Figure 6. **Results from Texverse.** For each mesh in a row, the red part denotes the query and the orange parts indicate the retrieved matches. While baselines often miss non-identical instances or include unrelated parts, our method can retrieve material-consistent components (window frames, ship masts, stairs, and roof logs).

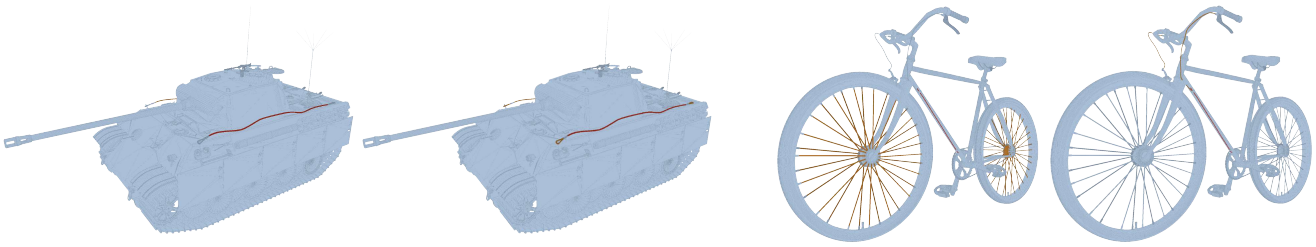

Figure 7. **Geometric variation within a material.** Given a query part (red), we show retrieved parts (left in each pair) versus ground truth (right in each pair). *Left:* Cable endpoints in the machinery exhibit different geometry from the cable body due to folding and knotting. *Right:* Bike cables near the down tube are tightly coiled, appearing geometrically distinct from the loosely suspended cable segments near the handlebars. Such cases can result in incomplete retrieval.

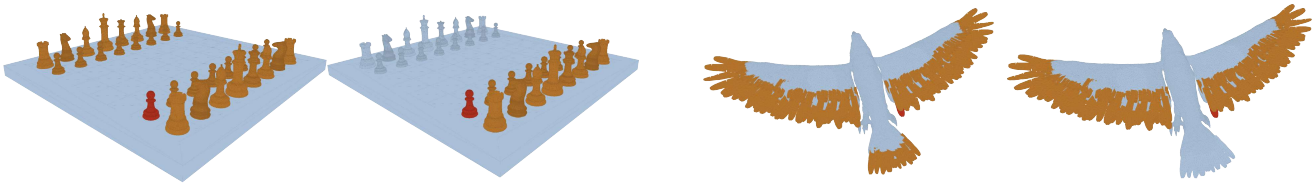

Figure 8. **Geometric similarity across different materials.** Given a query part (red), we show retrieved parts (left in each pair) versus ground truth (right in each pair). *Left:* Chess pieces have identical geometry across opposing teams, differing only in material. *Right:* Tail and wing feathers share similar geometric structure, leading to incorrect grouping despite different material assignments. These cases could be mitigated by training on more examples with similar contextual patterns.

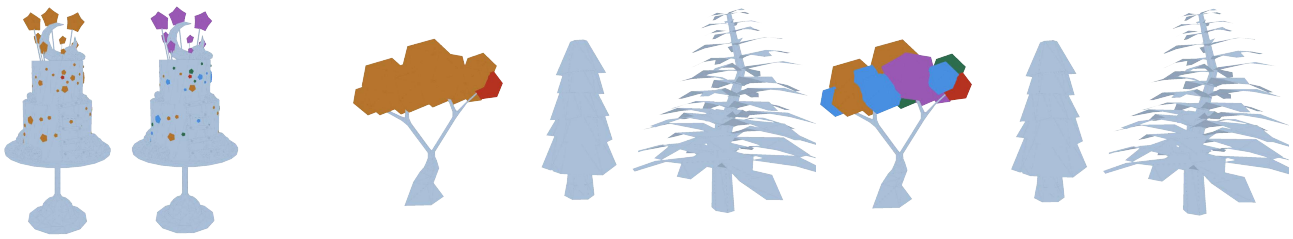

Figure 9. **Ambiguity arising from artistic intent.** Given a query part (red), we show retrieved parts (left in each pair) versus ground truth (right in each pair). *Left:* Our method retrieves all pentagon-shaped cake toppings based on geometric and contextual similarity, but the actual assignments had different materials (shown in varying colors). *Right:* We retrieve all foliage when querying one leaf cluster, but the mesh contains intentional material variation across the tree. Such artistic choices represent inherent ambiguity that cannot be resolved from geometry and context.

References

[1] Mihael Ankerst, Gabi Kastenmüller, Hans-Peter Kriegel, and Thomas Seidl. 3d shape histograms for similarity search and classification in spatial databases. In *International symposium on spatial databases*, pages 207–226. Springer, 1999. 2

[2] Siddhartha Chaudhuri. Thea. https://github.com/sidch/Thea. accessed Nov 2025. 2

[3] Yibo Zhang, Li Zhang, Rui Ma, and Nan Cao. Texverse: A universe of 3d objects with high-resolution textures. *arXiv preprint arXiv:2508.10868*, 2025. 3