# Combinative Matching for Geometric Shape Assembly

Supplementary Material

In this supplementary material, we present additional information and analyses not included in the main paper. In Section A, we provide further analysis of Combinative Matching using a toy dataset as well as extended experimental results and analyses, including further analysis on learned descriptors and orientations, as well as additional experimental results. In Section B, we detail the network architecture, including equivariant feature extractor, orientation hypothesizer, invariant feature computation, matching modules, and training objectives. In Section C, we describe additional details on the training and evaluation recipes, including hyperparameter settings and the evaluation details for multi-part assembly.

## A. Additional Experimental Results

### A.1. Results on Vanilla Breaking Bad Dataset

Since all our experiments were conducted on the volume-constrained version of the Breaking Bad dataset [13], we additionally provide a quantitative comparison on the vanilla version of the Breaking Bad dataset for a fair comparison with prior methods.

| Method | RMSE(R) ↓ (°) | RMSE(T) ↓ ($10^{-2}$) | $PA_{CD}$ ↑ (%) | CD ↓ ($10^{-3}$) |
|---|---|---|---|---|
| | `everyday` | | | |
| Global [6, 12] | 80.7 | 15.1 | 24.6 | 14.6 |
| LSTM [17] | 84.2 | 16.2 | 22.7 | 15.8 |
| DGL [4] | 79.4 | 15.0 | 31.0 | 14.3 |
| Wu et al. [18] | 79.3 | 16.9 | 8.41 | 28.5 |
| DiffAssemble [11] | 73.3 | 14.8 | 27.5 | - |
| Jigsaw [7] | 42.3 | 10.7 | 57.3 | 13.3 |
| PuzzleFusion++ [16] | <u>38.1</u> | **8.0** | <u>71.0</u> | <u>6.0</u> |
| **CMNet (Ours)** | **32.0** | <u>9.6</u> | **77.3** | **3.5** |
| | `everyday → artifact` | | | |
| Jigsaw [7] | 52.4 | 22.2 | 45.6 | 14.3 |
| PuzzleFusion++ [16] | 52.1 | **13.9** | 49.6 | 14.5 |
| **CMNet (Ours)** | **46.0** | <u>14.3</u> | **52.6** | **9.8** |

Table A1. Multi-part assembly results on vanilla Breaking Bad dataset [13]. Numbers in **bold** indicate the best performance and <u>underlined</u> ones are the second best.

Table A1 shows that ours consistently achieves accurate assembly, outperforming previous state-of-the-art methods. The robustness of our approach is particularly evident in cross-subset evaluation (`everyday` → `artifact`), where the performance remains stable despite substantial variations in object categories and fragment characteristics.

### A.2. Further Analysis on Combinative Matching

In this section, we provide deep, but intuitive analyses of our combinative matching, highlighting the necessity of explicit occupancy learning for robust shape assembly. To illustrate its significance, we build a synthetic dataset that highlights the critical role of occupancy learning, particularly in scenarios with ambiguous geometric patterns, *e.g.*, visual resemblance shown in Fig. A1.

#### A.2.1. Toy dataset with local ambiguity

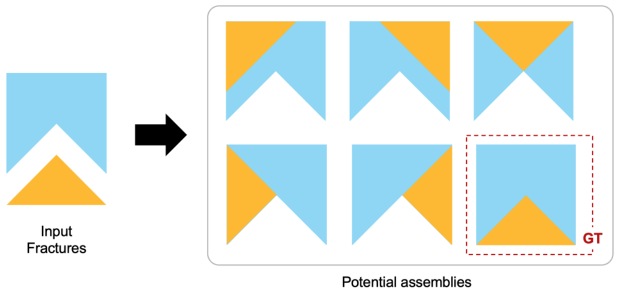


Figure A1. Example of potential failure assemblies caused by ***visual ambiguity*** within matching.

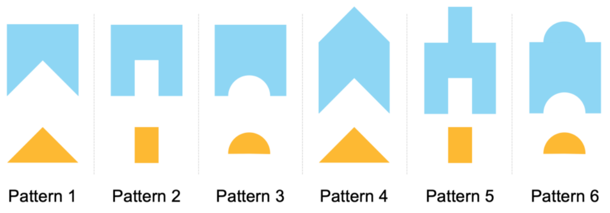


Figure A2. Six types of ambiguity pattern for toy dataset. 2D polygons are further extruded to 3D meshes.

**Ambiguity patterns.** The synthetic dataset consists of six carefully designed patterns, as shown in Fig. A2. These patterns were deliberately crafted to ensure that correct assembly depends primarily on recognizing occupancy and directionality rather than visual resemblance. We use pattern 1~3 for training, enabling the model to learn occupancy relationships and effectively mitigate visual ambiguities explicitly. Then patterns 4~6 are used to validate our assumption: if the model successfully learns occupancy-based complementary relationships from the train set, it should inherently generalize well to the visually analogous but structurally different test patterns.

We generate 200 random objects of each pattern {1, 2, 3} for training, 50 objects of each for validation, and 50 objects of each pattern {4, 5, 6} for testing. Although each pattern retains its pattern structure, small shape variations are introduced randomly. For further data generation details, we refer the reader to Algs. 1 and 2.

### A.2.2. Experiments

| Method | CRD ↓ ($10^{-2}$) | CD ↓ ($10^{-3}$) | RMSE(R) ↓ (°) | RMSE(T) ↓ ($10^{-2}$) |
|---|---|---|---|---|
| Jigsaw [7] | 17.64 | 6.95 | 84.98 | 24.81 |
| PMTR [5] | <u>16.01</u> | <u>6.01</u> | <u>70.13</u> | <u>15.70</u> |
| **CMNet (Ours)** | **9.24** | **2.07** | **55.29** | **13.40** |

Table A2. Pairwise shape assembly results on toy dataset.

| Shape Matching | Occupancy Matching | CRD ↓ ($10^{-2}$) | CD ↓ ($10^{-3}$) | RMSE(R) ↓ (°) | RMSE(T) ↓ ($10^{-2}$) |
|---|---|---|---|---|---|
| ✓ | | 11.95 | 3.92 | 66.90 | 17.06 |
| ✓ | ✓ | **9.24** | **2.07** | **55.29** | **13.40** |

Table A3. Ablation study on matching strategy.

Table A2 compares our method against recent state-of-the-art methods, *e.g.*, Jigsaw [7] and PMTR [5], on the synthetic dataset. In particular, both Jigsaw [7] and PMTR [5] show limited performance when tackling parts that exhibit visually similar but occupancy-opposed surfaces. By contrast, our method enforces a direct contrast in volume occupancy alongside shape similarity, thereby achieving robust, interlocking matches and significantly improving alignment accuracy across all metrics.

**Ablation Study.** In addition to comparing against existing methods, we validate the effect of explicit occupancy matching by ablating our framework. Table A3 reports the performance when removing the occupancy branch and relying solely on shape-based similarity. This confirms that identifying visually similar surfaces alone is not sufficient: *opposite* volume occupancy must also be enforced to prevent misleading matches arising from superficially alike shapes.

| Surface Segmentation | Desc. Type | CRD ↓ ($10^{-2}$) | CD ↓ ($10^{-3}$) | RMSE(R) ↓ (°) | RMSE(T) ↓ ($10^{-2}$) |
|---|---|---|---|---|---|
| | single | **17.24** | 7.48 | 87.84 | **21.84** |
| | primal-dual | 17.80 | **6.74** | **87.47** | 23.17 |
| ✓ | single | **17.19** | 7.29 | 89.06 | **21.68** |
| | primal-dual | 17.64 | **6.95** | **84.98** | 24.81 |
| GT | single | **12.88** | **4.63** | **71.10** | **10.89** |
| | primal-dual | 13.04 | 5.63 | 72.04 | 11.98 |

Table A4. Ablation study on surface segmentation and primal-dual matching modules in Jigsaw [7].

**Discussion on Jigsaw [7].** Jigsaw [7] applies a primal-dual descriptor specifically to the predicted mating surfaces, aiming to distinguish convex regions from concave ones.

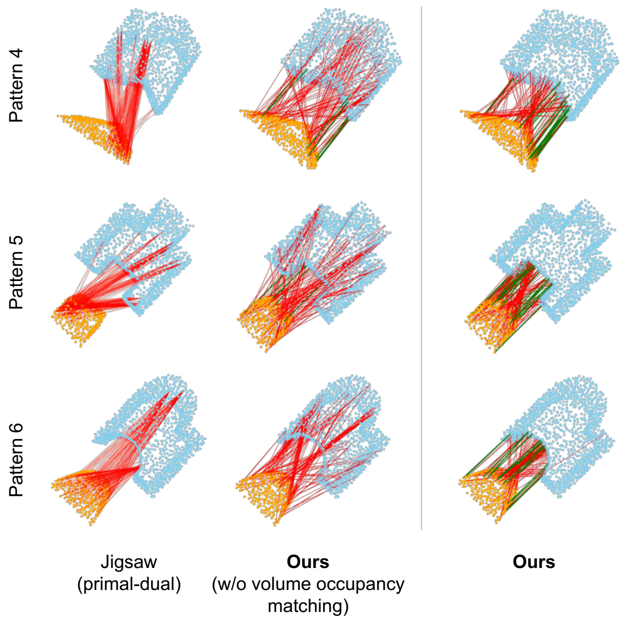


Figure A3. Visualization of top-$k$ matches ($k = 128$). Positive matches are colored in green, while negative matches are colored in red.

However, Table A4 shows that whether Jigsaw uses single-descriptor or primal-dual streams, and regardless of whether it segments mating surfaces or not, the improvements over a naive approach remain marginal. In several cases, primal-dual matching even underperforms single matching, indicating that the method struggles to capture a genuine complementary relationship (*i.e.*, opposing occupancy). We additionally provide ground-truth surface masks (GT) as a further upper bound, yet still see little gain from dual learning. Visualization result of top-$k$ matches in Fig. A3 corroborate this: the two-stream descriptor alone does not robustly reject visually deceptive matches, confirming that an explicit "identical shape + opposite occupancy" objectives are key to resolving ambiguous interfaces.

| Coarse Matcher | Fine Matcher | CRD ↓ ($10^{-2}$) | CD ↓ ($10^{-3}$) | RMSE(R) ↓ (°) | RMSE(T) ↓ ($10^{-2}$) |
|---|---|---|---|---|---|
| PMT | - | **15.80** | **5.55** | 71.91 | **14.31** |
| PMT | PMT | 16.01 | 6.01 | **70.13** | 15.70 |

Table A5. Ablation study on the feature matcher in PMTR [5].

**Discussion on PMTR [5].** As shown in Tab. A5, whether PMTR employs only a coarse matcher (PMT) or includes an additional fine matcher, both settings still struggle to address patterns demanding explicit occupancy opposition. While fine matching often helps localize small-scale interfaces in standard registration tasks, here it does not substantially alleviate the core limitation of PMTR's design:

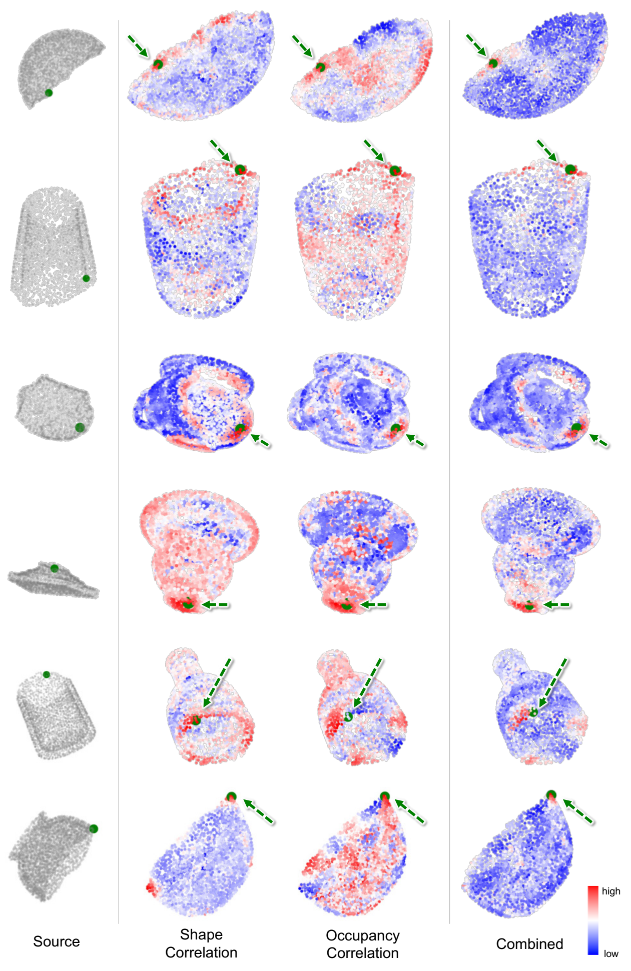


Figure A4. **Additional visualization of correlation distribution.** A green dot (●) on the left point cloud marks the source's $i$-th point, with corresponding true match points marked with green dots and arrows. Point colors represent correlation score magnitudes for the $i$-th point's similarity to each target point, with red and blue indicating high and low correlation scores, respectively.

maximizing only *identical surface shape* similarity. Without negatively enforcing "occupancy," the method remains prone to errors on visually alike yet geometrically mismatched parts.

## A.3. Additional orientation analysis

As discussed in our main paper (Fig. 4 in Sec. 4.3), the learned orientations, $\mathbf{F}_{\mathrm{d}}^{\mathrm{P}}$ and $\mathbf{F}_{\mathrm{d}}^{\mathrm{Q}}$, exhibit several notable patterns that enable the model to effectively align mating surface along with high interpretability. To further analyze these learned patterns across various examples, we provide additional visualizations of learned orientation under the same experimental setups described in Sec. 4.3.

Figure A5 presents the results for six distinct objects, again showcasing consistent patterns: (1) parallel alignment of source and target orientations ($\mathbf{x}_i$ and $\mathbf{y}_i$), (2) $\mathbf{x}_i$ directed toward the center of mating surfaces, (3) parallel alignment of $\mathbf{x}_i$ with 2D plane of mating surfaces, (4) outward/inward directed $\mathbf{y}_i$ based on convexity/concavity, and (5) correlation between magnitudes of $\mathbf{y}_i$ and surface curvature. These patterns highlight the robustness and adaptability of our method in learning *valid orientations* that respect the geometry and complementarity of mating surfaces *without any explicit supervision*, verifying both the effectiveness and interpretability of the proposed combinative matching.

## A.4. Additional correlation heatmap analysis

As in Fig. 5 of Sec. 4.3, we compare the correlation matrices for shape $\mathbf{C}_{\mathrm{s}}$, occupancy $\mathbf{C}_{\mathrm{o}}$, and combined correlation $\mathbf{C}$ of additional examples to further validate the efficacy of the proposed combinative matching approach over the conventional matching. Following the same experimental setup described Sec. 4.3 of the main paper, Fig. A4 illustrates the comparison, where we observe similar phenomena to those presented in the main paper.

When relying solely on the shape distribution $(\mathbf{C}_{\mathrm{s}})_i$, the best target match for the $i$-th source point is distributed over multiple regions due to local ambiguity of visual resemblance. The occupancy distribution $(\mathbf{C}_{\mathrm{o}})_i$ reveals relatively uniform scores across the surface, with a slightly higher concentration near the true match, offering complementary information but lacking precise localization. Integrating shape and occupancy information effectively resolves both local ambiguity and match confidence uncertainty, highlighting the importance of task-oriented multiple representation learning in combinative matching.

## A.5. Additional qualitative results

We provide additional qualitative comparisons against recent state-of-the-art methods [5, 7, 9, 18] on the Breaking Bad dataset. Figures A9 and A10 illustrate the comparison results in pairwise and multi-part settings, respectively. From the baselines, several notable patterns emerge: (1) **Failure in localizing mating surfaces**: The baselines lack an understanding of local orientations and occupancy on the mating surfaces. This results in incorrect placement of parts, which are often located in the air rather than at the interfaces of their corresponding parts. Examples of this failure include (a,c,d,e,f,h,i,j,k,n)-Wu *et al.* [18], (c,d,f,g,h,i,k)-Jigsaw [7], and (d,g)-PMTR [5] as shown in Fig. A10.
(2) **Failure in establishing correct correspondence**: While some methods perform decent localization of mating surfaces in pairwise assembly, they often fail to establish accurate correspondences due to local ambiguities, thus leading to incorrect assembly configurations such as reversed or overlapping parts. Specific examples of this issue are (d,g,h,k)-GeoTr [9], (b,i,m)-Jigsaw [7], and (a,c,e,f,h,j,k)-PMTR [5], as observed in Fig. A9. These observations highlight critical challenges faced by existing methods in

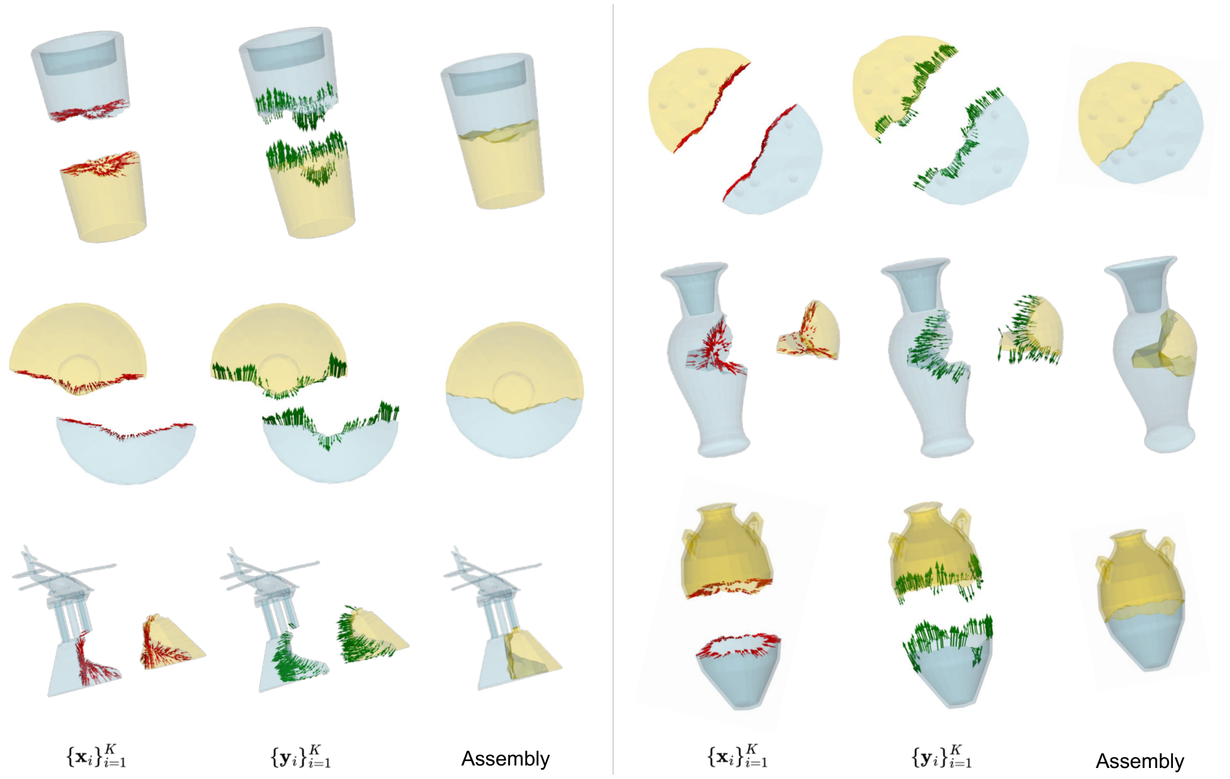

$\{\mathbf{x}_i\}_{i=1}^{K}$
$\{\mathbf{y}_i\}_{i=1}^{K}$
Assembly
$\{\mathbf{x}_i\}_{i=1}^{K}$
$\{\mathbf{y}_i\}_{i=1}^{K}$
Assembly

Figure A5. **Additional visualization of learned orientations.** We visualize learned orientations (in $\mathbb{R}^3$) of $\{\mathbf{x}_i\}_{i\in\mathcal{I}}$ (left, red arrows) and $\{\mathbf{y}_i\}_{i\in\mathcal{I}}$ (middle, green arrows). The assembly results are shown on the right.

both accurate interface localization and resolving ambiguities during assembly. By addressing these issues, we show that the proposed combinative matching demonstrates superior quantitative and qualitative results in both pairwise and multi-part scenarios.

## B. Additional Network Details

In this section, we provide the details of the network components that were omitted in the main paper for brevity.

### B.1. Equivariant feature extractor

For our backbone network $f_{\text{VNN}}$, we adopt Vector Neuron Network (VNN) [2], which represents neurons as 3D vectors, *i.e.*, $(\mathbf{F}_{\text{eqv}})_{i,j} \in \mathbb{R}^3$ for all $i, j$. This representation enables the network to handle SO(3) transformations directly, preserving consistent local feature orientations. Specifically, for any given input feature $\mathbf{F}$, each $i$-th layer $f^i_{\text{VNN}}$ of the network satisfies the following property: $f^i_{\text{VNN}}(\mathbf{FR}) = f^i_{\text{VNN}}(\mathbf{F})\mathbf{R}$ where $\mathbf{R} \in$ SO(3) [2]. To enhance the general context learning capabilities of the network, we modify the original VN-DGCNN [2, 15] architecture to broaden the receptive field of the features $\mathbf{F}_{\text{eqv}}$ by redesigning the network into a U-shaped architecture, as illustrated in Fig. A6. For downsampling and upsampling of the features in the process, we utilize the TransitionDown and TransitionUp modules, similar to the approach in [19]. This modification allows for an efficient contextual feature extraction while preserving the rotational equivariance property.

### B.2. Orientation hypothesizer

The backbone output $\mathbf{F}_{\text{eqv}} \in \mathbb{R}^{K\times D\times 3}$ is processed through an orientation hypothesizer $f_{\text{hyp}}$ to provide orientations such that $f_{\text{hyp}}(\mathbf{F}_{\text{eqv}}) = \mathbf{F}_{\text{d}} \in \mathbb{R}^{K\times 3\times 3}$. The hypothesizer contains a VN-Linear [2] layer that reduces the channel dimension of the input features from $D$ to 2, producing two vectors of size $\mathbb{R}^{K\times 2\times 3}$, where these two vectors, for each point, represent candidate orientations for the $x$-axis and $y$-axis, respectively. To ensure these vectors form a valid orientation basis, we apply the Gram-Schmidt process: First, we compute the 2D plane spanned by the two vectors and adjust the $y$-axis vector to ensure it is orthogonal to the other $x$-axis vector within this plane. Next, we calculate a third vector orthogonal to the 2D plane, resulting in a complete orthonormal basis. Finally, we apply L2 normalization to these vectors, ensuring that each $(\mathbf{F}_{\text{d}})_i \in$ SO(3) for all $i$, representing a valid 3D rotation matrix for each point. Note that this Gram-Schmidt process is rotation-equivariant, *i.e.*, $f_{\text{hyp}}(\mathbf{F}_{\text{eqv}}\mathbf{R}) = f_{\text{hyp}}(\mathbf{F}_{\text{eqv}})\mathbf{R}$ for any $\mathbf{R} \in$ SO(3), as discussed by Luo *et al*. [8].

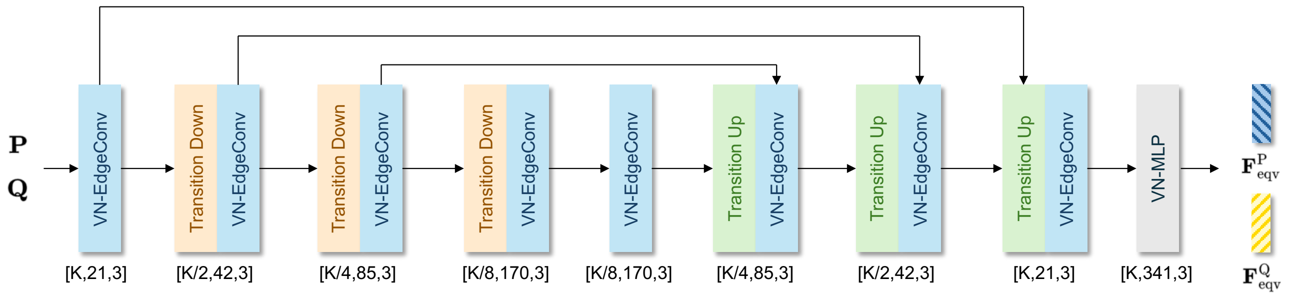


Figure A6. Overall framework of our U-shaped equivariant feature extractor.

## B.3. Invariant feature computation

Given the equivariant network $f_{\text{VNN}}$ which satisfies $f_{\text{VNN}}(\mathbf{XR}) = f_{\text{VNN}}(\mathbf{X})\mathbf{R}$ for any rotations $\mathbf{R} \in \text{SO}(3)$ and input points $\mathbf{X} \in \mathbb{R}^{K\times 3}$, along with the hypothesizer $f_{\text{hyp}}$, we aim to define a function $f_{\text{inv}}$ that provides invariant features $\mathbf{F}_{\text{inv}} \in \mathbb{R}^{K\times 3D}$, satisfying the following property:

$$f_{\text{inv}}(\mathbf{X}) = f_{\text{inv}}(\mathbf{XR}) = \mathbf{F}_{\text{inv}}, \tag{1}$$

for any rotation matrix $\mathbf{R}$. To achieve this, we define $f_{\text{inv}}$ as the dot product between the equivariant features and the hypothesized orientations:

$$f_{\text{inv}}(\mathbf{X}_i) = (\mathbf{F}_{\text{eqv}})_i \cdot (\mathbf{F}_{\text{d}})_i^\top, \tag{2}$$

for all $i \in \{1, \dots, K\}$, where $(\mathbf{F}_{\text{eqv}})_i$ represents the equivariant features output by $f_{\text{VNN}}$ and $(\mathbf{F}_{\text{d}})_i$ represents the hypothesized orientations output by $f_{\text{hyp}}$. The invariance property of $f_{\text{inv}}$ can be verified through the following proof:

$$\begin{aligned} f_{\text{inv}}(\mathbf{X}_i\mathbf{R}) &= f_{\text{VNN}}(\mathbf{X}_i\mathbf{R}) \cdot (f_{\text{hyp}}(f_{\text{VNN}}(\mathbf{X}_i\mathbf{R})))^\top \\ &= (f_{\text{VNN}}(\mathbf{X}_i)\mathbf{R}) \cdot (f_{\text{hyp}}(f_{\text{VNN}}(\mathbf{X}_i))\mathbf{R})^\top \\ &= f_{\text{VNN}}(\mathbf{X}_i)\mathbf{R}\mathbf{R}^\top(f_{\text{hyp}}(f_{\text{VNN}}(\mathbf{X}_i)))^\top \\ &= f_{\text{VNN}}(\mathbf{X}_i)(f_{\text{hyp}}(f_{\text{VNN}}(\mathbf{X}_i)))^\top \\ &= (\mathbf{F}_{\text{eqv}})_i \cdot (\mathbf{F}_{\text{d}})_i^\top \\ &= f_{\text{inv}}(\mathbf{X}_i). \end{aligned} \tag{3}$$

Thus, $f_{\text{inv}}$ is provably invariant to rotations, making $\mathbf{F}_{\text{inv}}$ suitable for the subsequent tasks requiring rotational invariance, such as shape and occupancy matching.

## B.4. Shape and occupancy matcher

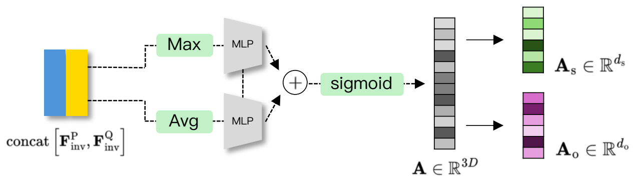
In Tab. A6, we tabularize the components of each layer for shape and occupancy matchers. Each matcher consists of a three-layer MLP, where each layer includes a linear transformation followed by normalization and activation. The key difference between shape and occupancy matchers lies in the activation function used in the final layer: LeakyReLU is used for the shape matcher to allow the shape correlation matrix $\mathbf{C}_{\text{s}}$ to have a wider range of values, capturing large variations in shape similarity. In contrast, Tanh

| Layer | Shape Matcher | Occupancy Matcher |
|---|---|---|
| 1 | Conv1D(1023 $\rightarrow$ 512)<br>InstanceNorm(512)<br>LeakyReLU | Conv1D(1023 $\rightarrow$ 512)<br>InstanceNorm(512)<br>LeakyReLU |
| 2 | Conv1D(512 $\rightarrow$ 512)<br>InstanceNorm(512)<br>LeakyReLU | Conv1D(512 $\rightarrow$ 512)<br>InstanceNorm(512)<br>LeakyReLU |
| 3 | Conv1D(512 $\rightarrow$ 512)<br>InstanceNorm(512)<br>LeakyReLU | Conv1D(512 $\rightarrow$ 512)<br>InstanceNorm(512)<br>Tanh |

Table A6. Components of the shape and occupancy matchers.

is employed for the occupancy descriptors to constrain extreme activations, ensuring that the occupancy correlation matrix $\mathbf{C}_{\text{o}}$ avoids overemphasizing noisy or outlier regions, particularly those unrelated to occupancy learning, such as non-mating surfaces.

Empirical observations, as shown in Fig. A4, indicate that outliers occur more frequently in occupancy correlations compared to shape correlations, with large occupancy scores being more uniformly distributed across surfaces, whereas shape scores are more localized and structured. Allowing large variations in $\mathbf{C}_{\text{s}}$, therefore, ensures that dominant shape features are captured (introducing local ambiguity), while controlling $\mathbf{C}_{\text{o}}$ prevents outliers or irrelevant regions from skewing the overall correlation (resolving the ambiguity), resulting in more reliable correlations $\mathbf{C}$.



concat $[\mathbf{F}^{\mathrm{P}}_{\mathrm{inv}}, \mathbf{F}^{\mathrm{Q}}_{\mathrm{inv}}]$
Max
Avg
MLP
MLP
sigmoid
$\mathbf{A} \in \mathbb{R}^{3D}$
$\mathbf{A}_{\text{s}} \in \mathbb{R}^{d_{\text{s}}}$
$\mathbf{A}_{\text{o}} \in \mathbb{R}^{d_{\text{o}}}$

Figure A7. Pipeline for soft-attention generation.

When generating shape and occupancy descriptors, we dynamically adjust the importance of feature channels using soft-channel attention, inspired by SENet [3]. As illustrated in Fig. A7, we first concatenate the pair of invariant features along the spatial dimension and apply average-pooling and max-pooling. The pooled outputs are passed to two shared MLPs followed by a sigmoid activation to compute channel-wise statistics, which serve as the soft-attention values for the shape and occupancy descriptors. The output $\mathbf{A}$ is divided by two along the channel dimension to produce $\mathbf{A}_\text{s}$ and $\mathbf{A}_\text{o}$, the soft attention weights for the shape and occupancy descriptors, respectively, each of which weights the feature channels to enhance relevant information for each descriptor as in [3].

### B.5. Details on training objectives

**Circle loss [14].** In the definition of circle loss provided in the main paper, we omit hyperparameter $\gamma$ used in the original circle loss formulation, for brevity in our demonstration. The definition of $\mathcal{L}_\text{circle}$ is formulated as

$$\mathop{\mathbb{E}}_{i\sim\mathcal{I}}\left[\log\left(\sum_{j\in\mathcal{E}_\text{p}(i)} e^{\gamma(\phi^{i,j}-\Delta_\text{p})^2}\cdot\sum_{k\in\mathcal{E}_\text{n}(i)} e^{\gamma(\Delta_\text{n}-\phi^{i,k})^2}\right)\right], \tag{4}$$

where $\gamma$ scales the sharpness of the exponential terms, amplifying or reducing the emphasis on outliers and enabling more stable training and $\phi^{i,j}$ computes similarity or dissimilarity between a pair of given features.

**Point matching loss [10].** We now detail the point matching loss $\mathcal{L}_\text{p}$ which is jointly used alongside the combinative matching objectives. The point matching loss is a negative log-likelihood loss on predicted match probabilities $\mathbf{Z}\in\mathbb{R}^{(N+1)\times(M+1)}$, the dual-normalized assignment matrix with dustbin, the output from the Optimal Transport layer [10]. Given a set of ground-truth correspondences $\mathcal{C}$, and the sets of unmatched points, $(\mathcal{I}^\text{P})' = \{x : x \in [N] \wedge x \notin \mathcal{I}^\text{P}\}$ and $(\mathcal{I}^\text{Q})' = \{x : x \in [M] \wedge x \notin \mathcal{I}^\text{Q}\}$ where $[K] = \{1,\dots,K\}$, the point matching loss is defined as

$$\begin{aligned}\mathcal{L}_\text{p} = &-\sum_{(i,j)\in\mathcal{C}} \log \mathbf{Z}_{i,j} \\ &-\sum_{i\in(\mathcal{I}^\text{P})'} \log \mathbf{Z}_{i,M+1} - \sum_{j\in(\mathcal{I}^\text{Q})'} \log \mathbf{Z}_{N+1,j},\end{aligned} \tag{5}$$

which enforces the predicted match probabilities to align closely with the ground-truth matches, ensuring accurate point-to-point correspondence.

## C. Details on Training and Evaluation Setup

### C.1. Hyperparameter setup

To determine positive matches between mating surfaces, the distance threshold was set to $\tau = 0.018$. For the circle loss [14], we used margin hyperparameters $\Delta_p = 0.1$ and $\Delta_n = 1.4$, along with a scale factor $\gamma = 24$. The channel sizes for the equivariant feature embedding, shape descriptor, and occupancy descriptor are set to $D = 341$, $d_\text{s} = 512$, and $d_\text{o} = 512$, respectively. We use the normalization constant $Z = \sqrt{512}$ to construct the cost matrix $\mathbf{C}$.

### C.2. Multi-part assembly details

Following the approach of Lee *et al.* [5], we extend our pairwise matching framework CMNet to handle multi-part assembly in a consistent manner. Specifically, our method first learns ***local pairwise compatibilities*** between part pairs through pairwise matching, and then estimates globally consistent poses via ***pose graph optimization***.

**Learning Pairwise Compatibility.** We construct 2-part training pairs from all objects in the Breaking Bad dataset [13], where each object consists of 2 to 20 parts. Specifically, for each training sample, we randomly select a *source* part and choose as its *target* part the one that shares the largest ground-truth mating surface area with it. We train the network for 350 epochs on the `everyday` subset and 300 epochs on the `artifact` subset, using the same model configurations (*e.g.*, hyperparameters) as in the pairwise setup. This training is intended to allow the model to learn local pairwise compatibilities, which are subsequently utilized during global optimization in the multi-part setting.

**Inference & Pose-Graph Optimization.** Given an $N$-part object at the inference time, we first predict relative poses for all $\binom{N}{2}$ part pairs to construct a fully connected pose graph, as illustrated in Fig. A8.

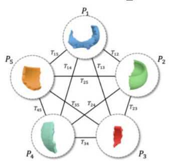


Figure A8. Pose graph example.

In the *pose graph*, each part $P_i$ (with $i = 1,\dots,N$) becomes a node, and each weighted edge carries a *pairwise relative pose* $T_{ij} = \{\mathbf{R}_{ij}, \mathbf{t}_{ij}\}$ along with its information matrix $\mathbf{I}_{ij} = \left(\sum_{p=1}^{|P_i|}\sum_{q=1}^{|P_j|}\exp(\mathbf{Z}_{p,q}^{(ij)})\right)^{-1}\cdot I_6$, where $\mathbf{Z}^{(ij)} \in \mathbb{R}^{|P_i|\times|P_j|}$ denotes the soft assignment matrix between parts $P_i$ and $P_j$, obtained via Optimal Transport [10].

To reduce noise and eliminate unreliable matches, we prune all outgoing edges except the one with the highest matchability score for each node. We then recover global poses $(\tilde{\mathbf{R}}_i, \tilde{\mathbf{t}}_i)$ via Shonan Averaging [1], with the largest part set as an anchor, similar to [5, 7].

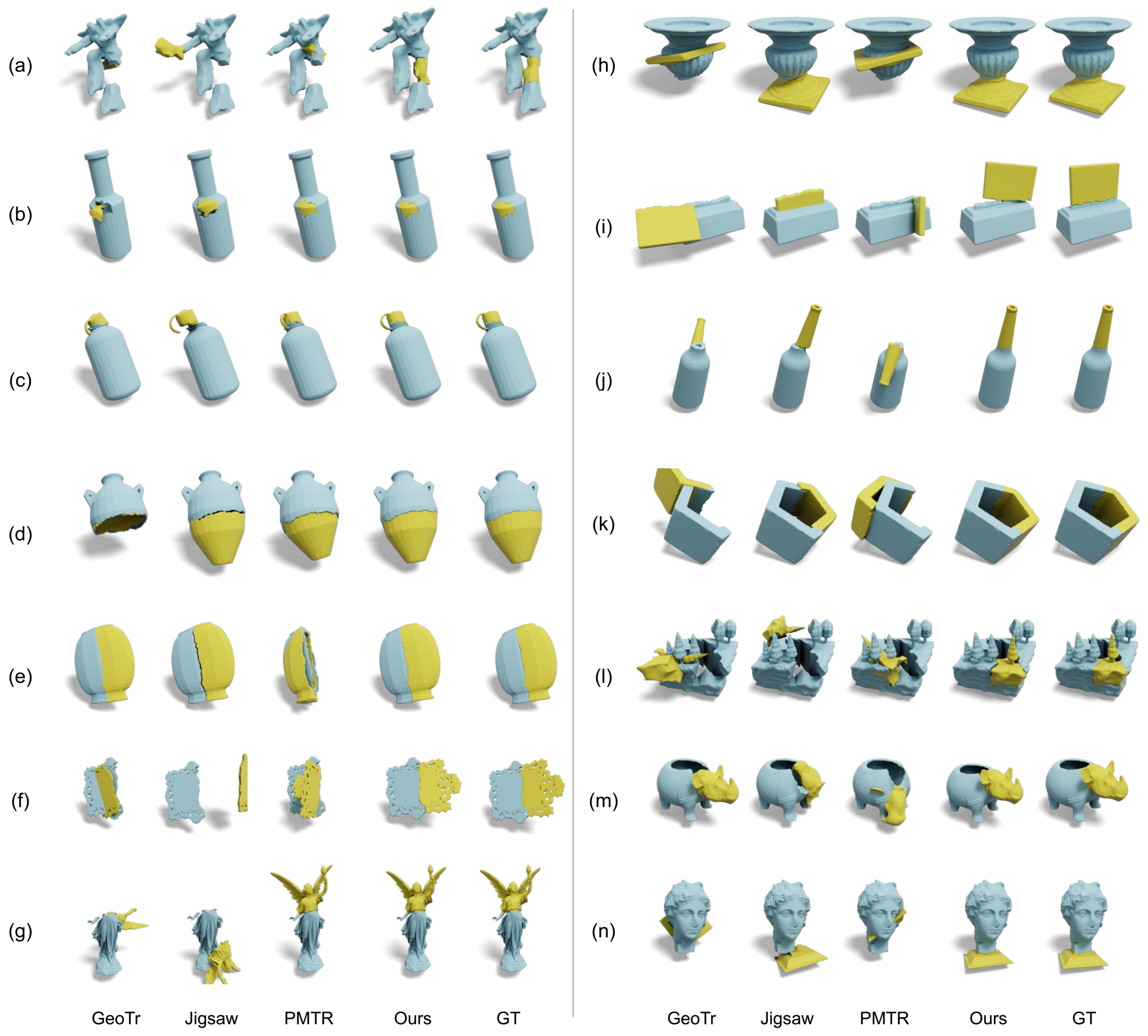


Figure A9. Additional qualitative comparison for pairwise shape assembly.

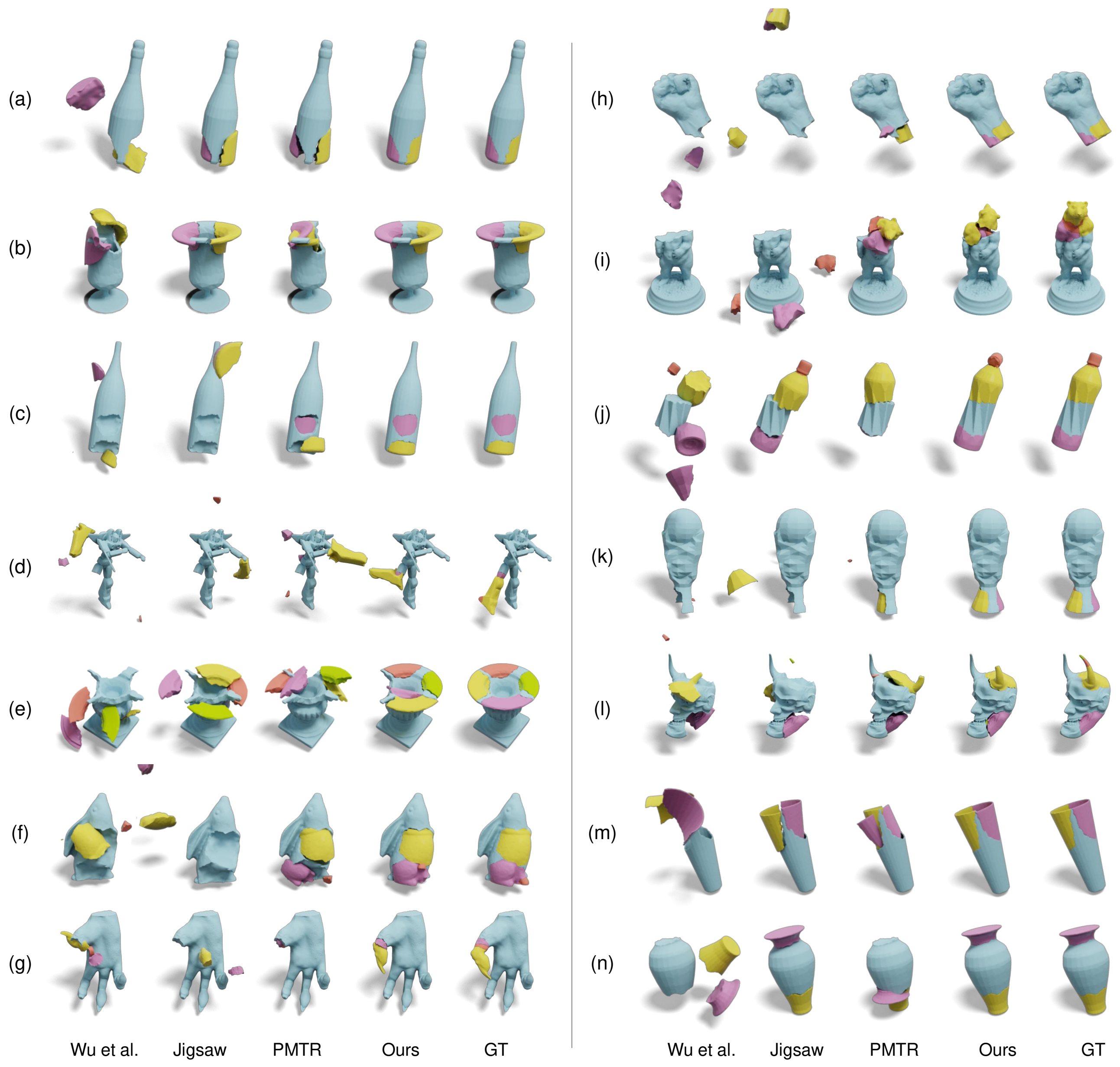


Figure A10. Additional qualitative comparison for multi-part assembly.

**Algorithm 1** Synthetic Data Generation Process (`train`/`val`)

```
1: **Input:** Pattern $p$
2: **Output:** 3D mesh pair $(\mathcal{M}_{\text{src}}, \mathcal{M}_{\text{trg}})$
3: **if** $p =$ pattern1 **then**
4:     $h \leftarrow \text{Uniform}\Big(0, \frac{1}{3}\Big)$
5:     center $\leftarrow \big[0.5,\ 0.5 + h\big]$
6:     src_points $\leftarrow \big[[0, h],\ [0, 1],\ [1, 1],\ [1, h],\ \texttt{center}\big]$
7:     trg_points $\leftarrow \big[[0, h],\ [1, h],\ \texttt{center}\big]$
8: **else if** $p =$ pattern2 **then**
9:     $h \leftarrow \text{Uniform}\Big(0, \frac{1}{3}\Big), \quad d \leftarrow \text{Uniform}\Big(\frac{1}{6}, \frac{1}{2}\Big)$
10:    left_center $\leftarrow \big[\frac{1}{3},\ h + d\big]$,    right_center $\leftarrow \big[\frac{2}{3},\ h + d\big]$
11:    src_points $\leftarrow \big[[0, h],\ [\frac{1}{3}, h],\ \texttt{left\_center},\ \texttt{right\_center},\ [\frac{2}{3}, h],\ [1, h],\ [1, 1],\ [0, 1]\big]$
12:    trg_points $\leftarrow \big[[\frac{1}{3}, h],\ [\frac{2}{3}, h],\ \texttt{right\_center},\ \texttt{left\_center}\big]$
13: **else if** $p =$ pattern3 **then**
14:    $h \leftarrow \text{Uniform}\Big(0, \frac{1}{3}\Big), \quad r \leftarrow \text{Uniform}\Big(\frac{1}{6}, \frac{1}{2}\Big)$
15:    $\theta \leftarrow \text{linspace}(0, \pi, n)$
16:    circle_points $\leftarrow \Big(0.5 + r\cos\theta,\ h + r\sin\theta\Big)$
17:    src_points $\leftarrow \big[[0, h],\ [0.5 - r, h],\ \texttt{circle\_points},\ [0.5 + r, h],\ [1, h],\ [1, 1],\ [0, 1]\big]$
18:    trg_points $\leftarrow \big[[0.5 - r, h],\ \texttt{circle\_points},\ [0.5 + r, h]\big]$
19:    $\mathcal{P}_{\text{src}} \leftarrow$ Polygon(src_points),    $\mathcal{P}_{\text{trg}} \leftarrow$ Polygon(trg_points)
20:    $\mathcal{M}_{\text{src}} \leftarrow$ Extrude($P_{\text{src}}, z$),    $\mathcal{M}_{\text{trg}} \leftarrow$ Extrude($P_{\text{trg}}, z$)
```

**Algorithm 2** Synthetic Data Generation Process (`test`)

```
1: **Input:** Pattern $p$
2: **Output:** 3D mesh pair $(\mathcal{M}_{\text{src}}, \mathcal{M}_{\text{trg}})$
3: **if** $p =$ pattern4 **then**
4:     $h \leftarrow \text{Uniform}\Big(0, \frac{1}{3}\Big)$
5:     center $\leftarrow \big[\text{Uniform}\Big(\frac{1}{6}, \frac{5}{6}\Big),\ \text{Uniform}\Big(\frac{1}{6}, \frac{1}{2}\Big) + h\big]$
6:     src_points $\leftarrow \big[[0,h],\ \texttt{center},\ [1,h],\ [1,1],\ [\texttt{center}_x,\ \texttt{center}_y + (1-h)],\ [0,1]\big]$
7:     trg_points $\leftarrow \big[[0,h],\ \texttt{center},\ [1,h]\big]$
8: **else if** $p =$ pattern5 **then**
9:     $h \leftarrow \text{Uniform}\Big(0, \frac{1}{3}\Big), \quad d \leftarrow \text{Uniform}\Big(\frac{1}{6}, \frac{1}{2}\Big)$
10:    $x_1 \leftarrow \text{Uniform}\Big(\frac{0.5}{6}, \frac{2.5}{6}\Big), \quad x_2 \leftarrow \text{Uniform}\Big(\frac{3.5}{6}, \frac{5.5}{6}\Big)$
11:    left_center $\leftarrow \big[x_1,\ h+d\big]$,    right_center $\leftarrow \big[x_2,\ h+d\big]$
12:    src_points $\leftarrow \big[[0,h],\ [x_1,h],\ \texttt{left\_center},\ \texttt{right\_center},\ [x_2,h],\ [1,h],$
13:              $[1,1],\ [x_2,1],\ [x_2,d+1],\ [x_1,d+1],\ [x_1,1],\ [0,1]\big]$
14:    trg_points $\leftarrow \big[[x_1,h],\ \texttt{left\_center},\ \texttt{right\_center},\ [x_2,h]\big]$
15: **else if** $p =$ pattern6 **then**
16:    $h \leftarrow \text{Uniform}\Big(0, \frac{1}{3}\Big), \quad r \leftarrow \text{Uniform}\Big(\frac{1}{6}, \frac{1}{3}\Big)$
17:    $c \leftarrow \text{Uniform}(r, 1-r)$
18:    $\theta \leftarrow \text{linspace}(0, \pi, n)$
19:    circle_points_down $\leftarrow \Big(0.5 + r\cos\theta,\ h + r\sin\theta\Big)$
20:       circle_points_up $\leftarrow$ update each point's $y$ as $y \leftarrow y + (1-h)$.
21:    src_points $\leftarrow$ vstack$\Big(\ [[0,h],\ [c-r,h],\ \texttt{circle\_points\_down},\ [c+r,h],\ [1,h],\ [1,1],\ [c+r,1],$
22:              $\texttt{circle\_points\_up},\ [c-r,1],\ [0,1]]\Big)$
23:    trg_points $\leftarrow$ vstack$\Big(\ [[c-r,h],\ \texttt{circle\_points\_down},\ [c+r,h]]\Big)$
24: **end if**
25: $\mathcal{P}_{\text{src}} \leftarrow$ Polygon(src_points),    $\mathcal{P}_{\text{trg}} \leftarrow$ Polygon(trg_points)
26: $\mathcal{M}_{\text{src}} \leftarrow$ Extrude$(P_{\text{src}}, z)$,    $\mathcal{M}_{\text{trg}} \leftarrow$ Extrude$(P_{\text{trg}}, z)$ =0
```

## References

[1] Frank Dellaert, David M Rosen, Jing Wu, Robert Mahony, and Luca Carlone. Shonan rotation averaging: Global optimality by surfing so (p)ˆ n so (p) n. In *ECCV*, 2020. 6

[2] Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas J Guibas. Vector neurons: A general framework for so (3)-equivariant networks. In *ICCV*, 2021. 4

[3] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018. 5, 6

[4] Jialei Huang, Guanqi Zhan, Qingnan Fan, Kaichun Mo, Lin Shao, Baoquan Chen, Leonidas Guibas, and Hao Dong. Generative 3d part assembly via dynamic graph learning. In *NeurIPS*, 2020. 1

[5] Nahyuk Lee, Juhong Min, Junha Lee, Seungwook Kim, Kanghee Lee, Jaesik Park, and Minsu Cho. 3d geometric shape assembly via efficient point cloud matching. In *ICML*, 2024. 2, 3, 6

[6] Jun Li, Chengjie Niu, and Kai Xu. Learning part generation and assembly for structure-aware shape synthesis. In *AAAI*, 2020. 1

[7] Jiaxin Lu, Yifan Sun, and Qixing Huang. Jigsaw: Learning to assemble multiple fractured objects. In *NeurIPS*, 2023. 1, 2, 3, 6

[8] Shitong Luo, Jiahan Li, Jiaqi Guan, Yufeng Su, Chaoran Cheng, Jian Peng, and Jianzhu Ma. Equivariant point cloud analysis via learning orientations for message passing. In *CVPR*, 2022. 4

[9] Zheng Qin, Hao Yu, Changjian Wang, Yulan Guo, Yuxing Peng, and Kai Xu. Geometric transformer for fast and robust point cloud registration. In *CVPR*, 2022. 3

[10] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *CVPR*, 2020. 6

[11] Gianluca Scarpellini, Stefano Fiorini, Francesco Giuliari, Pietro Moreiro, and Alessio Del Bue. Diffassemble: A unified graph-diffusion model for 2d and 3d reassembly. In *CVPR*, 2024. 1

[12] Nadav Schor, Oren Katzir, Hao Zhang, and Daniel Cohen-Or. Componet: Learning to generate the unseen by part synthesis and composition. In *ICCV*, 2019. 1

[13] Silvia Sellán, Yun-Chun Chen, Ziyi Wu, Animesh Garg, and Alec Jacobson. Breaking bad: A dataset for geometric fracture and reassembly. In *NeurIPS Datasets and Benchmarks Track*, 2022. 1, 6

[14] Yifan Sun, Changmao Cheng, Yuhan Zhang, Chi Zhang, Liang Zheng, Zhongdao Wang, and Yichen Wei. Circle loss: A unified perspective of pair similarity optimization. In *CVPR*, 2020. 6

[15] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM TOG*, 38(5): 1–12, 2019. 4

[16] Zhengqing Wang, Jiacheng Chen, and Yasutaka Furukawa. Puzzlefusion++: Auto-agglomerative 3d fracture assembly by denoise and verify. In *ICLR*, 2025. 1

[17] Rundi Wu, Yixin Zhuang, Kai Xu, Hao Zhang, and Baoquan Chen. Pq-net: A generative part seq2seq network for 3d shapes. In *CVPR*, 2020. 1

[18] Ruihai Wu, Chenrui Tie, Yushi Du, Yan Zhao, and Hao Dong. Leveraging se (3) equivariance for learning 3d geometric shape assembly. In *ICCV*, 2023. 1, 3

[19] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *ICCV*, 2021. 4