PuzzleNet: Boundary-Aware Feature Matching for Non-overlapping 3D Point Clouds Assembly

Haoyu Liu[2,1], Jianwei Guo[1,2]*, Haiyong Jiang[2], Yanchao Liu[2,1], Xiaopeng Zhang[1,2], Dong-Ming Yan[1,2]
[1]MAIS, Institute of Automation, Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences

Abstract

We address the 3D shape assembly of multiple geometric pieces without overlaps, a scenario often encountered in 3D shape design, field archeology, and robotics. Existing methods depend on strong assumptions on the number of shape pieces and coherent geometry or semantics of shape pieces. Despite raising attention on 3D registration with complex or low overlapping patterns, few methods consider shape assembly with rare overlaps. To address this problem, we present a novel framework inspired by solving puzzles called PuzzleNet, which conducts multi-task learning by leveraging 3D alignment and boundary information. Specifically, we design an end-to-end neural network based on a point cloud transformer with a two-way branch for estimating rigid transformation and predicting boundaries simultaneously. The framework is then naturally extended to reassemble multiple pieces into a full shape by using an iterative greedy approach based on the distance between each pair of candidate matched pieces. We train and evaluate PuzzleNet on both real-world urban scan data (DublinCity) and synthetic CAD models (ModelNet40). Experiments demonstrate our effectiveness in solving 3D shape assembly for multiple pieces with arbitrary geometry and inconsistent semantics. We significantly outperform state-of-the-art 3D registration approaches as well as the closely related method for shape mating. Our code is available at *https://github.com/Gibbs-liu/PuzzleNet.*

Keywords: Shape assembly, 3D registration, Geometric learning, Boundary feature.

1. Introduction

Shape assembly to create 3D complex scenes is one of the most central problems in computer graphics. The problem of assembling non-overlapping point clouds is fundamental to many practical applications, such as 3D design [10], field archaeology [20], visual art [38], and robotics. For example, when an object like porcelain is broken into multiple pieces, we need to scan the pieces and automatically assemble them to form a complete one. This problem is challenging in several aspects. First, the multiple pieces are usually highly or low overlapped, making it difficult to determine their alignment. Second, shape pieces may have arbitrary geometry and no consistent semantics, thus reconstruction has to be based purely on the geometric cues. Third, the target shape is usually composed of multiple pieces, where the organization between different pieces is unknown.

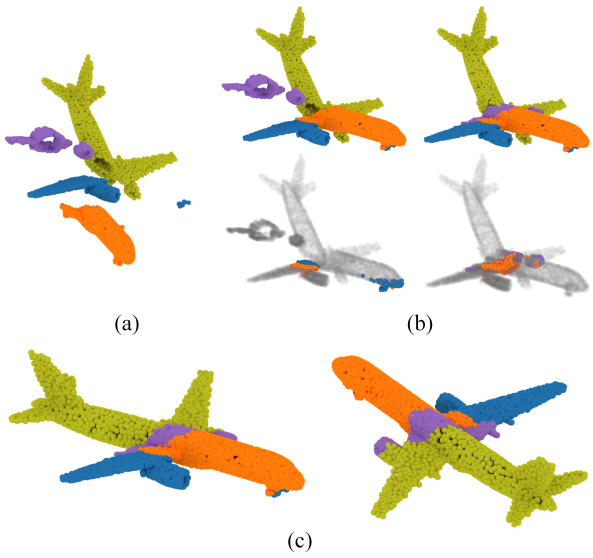

Figure 1: PuzzleNet is designed to automatically assemble complete shapes from multiple pieces. Given 4 geometric pieces (a), PuzzleNet learns pairwise shape alignment by predicting the transformation and the boundary information (b). Then we achieve a full 3D shape (c) by iteratively computing a global multi-piece assembly based on the distance between the boundaries of each pair.

Assembling multiple shape pieces consists of two key steps, *i.e.*, pairwise part alignment and multiple parts assembly. Pairwise part alignment registers two pieces so that

*Corresponding Author: jianwei.guo@nlpr.ia.ac.cn

they can be fused together seamlessly. Multi-piece assembly finds the correct connectivity between individual pieces to make a plausible configuration. These two steps are interwined and make the problem more challenging.

Previous works have made many attempts at shape assembly. A typical line of studies [7, 18] composes two disassembled shape pieces by exploring implicit shape reconstruction or field guidance. However, these methods mainly focus on the assembly of two pieces and cannot handle multiple ones with complex connectivity. Recent works [19, 28, 27] deal with shape assembly of multiple shape parts with consistent semantic labels. Therefore it suffers from the generalization problem when encountering shape pieces with arbitrary geometry and without coherent semantics.

A key step for shape assembly is partial shape alignment. Despite extensive studies, most previous works focus on 3D registration [23], which usually has a heavy reliance on high overlaps. Recently, there is also an increasing number of literature on low overlapping scenarios for more robust 3D registration [21, 32]. However, it is still an open problem how to extend the scheme to 3D shape assembly, which usually assumes rarer and even no overlaps.

To address the above challenges, we present a novel deep learning-based framework, named PuzzleNet. Our basic idea is inspired by puzzle solving, where puzzles with similar boundary features tend to be assembled together and the alignment quality between the boundaries of two pieces hints if the two pieces are well aligned. In shape assembly, disassembled point clouds are usually aligned by matching the boundaries and the corresponding pieces' extensions. Therefore boundary features are informative about the final alignment. As for the assembly of multiple pieces, we can check whether two pieces are aligned by comparing the corresponding boundary distances.

Our PuzzleNet comprises three steps, *i.e.*, 3D transformation estimation, boundary point classification, and multi-piece assembly. The first two steps are learned with an end-to-end multi-task neural network, while the last step takes a greedy strategy to find the best-aligned pieces one by one until attaining the final assembled shape. For transformation estimation and boundary prediction, PuzzleNet combines an attention mechanism and point-based learning modules to focus on connecting boundaries implicitly. In particular, we employ a point-transformer-like network to predict the proper transformation matrix to align two point clouds.

Our contributions can be summarized as follows:

- We propose a novel framework for learning to assemble non-overlapping point clouds with arbitrary geometry and without prior semantic knowledge, as shown in Fig. 1.
- We present a multi-task neural network, titled PuzzleNet, for inferring rigid transformation and boundary features simultaneously. We show that the boundary-aware design can be used to iteratively reassemble multiple pieces into a full shape.
- We construct two datasets based on the ModelNet40 and DublinCity for studying the problem of multi-piece assembly. Extensive experiments demonstrate the performance and superiority of our framework through comprehensive evaluation and comparisons.

2. Related Work

3D shape assembly. Over the past decades, shape assembly has been widely studied in computer graphics and computer vision communities. Many algorithms have been proposed to address part assembly that aims at composing component parts for model creation. Huang et al. [18] introduce a field-guided registration algorithm by aligning two input parts in a feature-conforming manner. However, their method is only applicable to registering two parts and the open boundaries should be well-defined. The majority of subsequent methods apply deep neural networks to facilitate assembly-based modeling by leveraging part segmentation [28, 44, 37, 25, 26] or formulating part assembly as a graph learning problem [15, 19]. Therefore, the above methods usually rely on a part database (*e.g.*, PartNet [30]) or assume prior part semantics.

Without the assistance of known semantic labels, researchers propose to reassemble the original shape from a set of fragments. The typical pipeline of geometric guided methods [20, 16, 46] is to construct different handcraft features of contacting regions and then increasingly reassemble shapes from many candidate matches. For example, Huang et al. [20] present a system for fragmented solid reassembly, but they require an explicit segmentation step to construct patch-based fracture surface features. Zhang et al. [46] reassemble broken pieces by using template guidance, *i.e.*, composing fragmented pieces based on their best match to a complete model. Neural shape mating (NSM) [7] is the most relative to ours, which focuses on learning the alignment of two non-overlapping point clouds from geometric information rather than semantics. They develop a Transfomer-based rigid pose regressor to learn to approximate the spatial transformation, then adopt an adversarial learning scheme that learns shape priors for evaluating the plausibility of the predicted shape mating configurations. However, this approach is designed for only aligning two object parts as they can not conduct contact-region matching. Instead, our PuzzleNet explicitly learns boundary features and assembles multiple pieces by exploiting the similarities in the local boundary geometry of adjacent pieces.

Point cloud registration. The task of shape assembly is

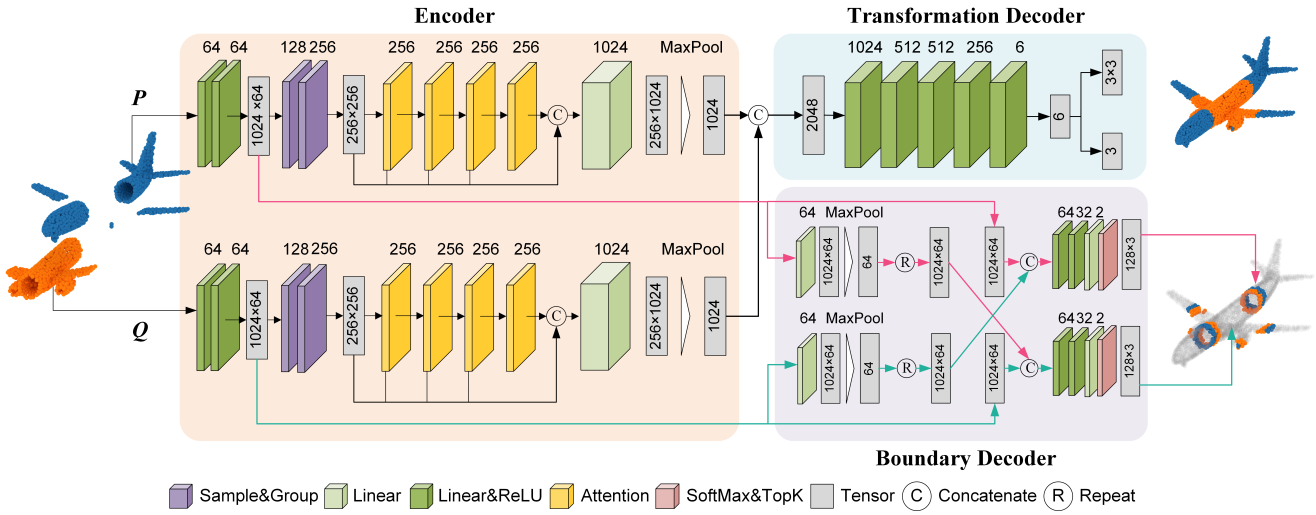

Figure 2: Network architecture of our proposed PuzzleNet. We apply a transformer-based encoder to extract both global and local geometric features, which are then passed to two decoders to estimate a rigid transformation and predict the boundary respectively. The number of output channels is shown above the corresponding modules.

also intimately related to 3D shape registration. Classical 3D registration methods can be divided into global registration and local registration. The former computes a coarse alignment of two shapes, while the latter aims at further improving the accuracy of the initial estimate. Global registration methods work through the mechanism of first searching correspondence based on geometric features (*e.g.*, point-based [1, 48, 29, 13, 40] or primitive-based [6, 47] descriptors), then estimating the transformation by using a RANSAC-like algorithm [9]. For local registration, ICP [3] and its variants [11, 33, 4] are the representatives that iteratively search closet points and updates transformation by solving a least square problem.

Thanks to the advances in point-based 3D deep learning [31, 36, 14, 17], deep point cloud registration has also drawn a lot of attention recently. Such methods either estimate an accurate correspondence search by robustly feature learning [45, 42, 8, 12], or directly learn the final transformation matrix by proposing end-to-end neural networks [2, 35, 24]. Although achieving encouraging results for dense point clouds, they have problems when the input has a small overlapping area. Thus, registration with low overlap has become the focus of research. Predator [21] first introduces an overlap attention block that concentrates on the points in the overlapping region and predicts the saliency of overlap points. GeoTransformer [32] proposes a geometric transformer for accurate super-point mapping, making it robust in low-overlap cases. NgeNet [49] utilizes a geometric guided encoding module and a multi-scale architecture to learn point-wise features, then apply a learning-free voting strategy for filtering wrong feature mappings. PCAM [5] learns the matching regions by a point-wise product of cross-attention matrices mixing both low-level and high-level features. OMNet [41] learns masks to reject non-overlapping geometric features. However, the above methods still heavily rely on overlap regions, thus they can not handle non-overlap examples as in our work.

3. Overview

3.1. Problem Formulation

Given a set of point clouds without overlaps, we aim to assemble them piece by piece to form an integrated shape. The problem can be decomposed into two sub-problems: (i) assembling two pieces with an optimal rigid transformations; (ii) finding pair-wise pieces close to each other able to be assembled. As boundaries between two pieces usually give a clue about assemblable pieces, we convert the second problem as boundary prediction and matching. To make it brief, we take two point clouds $P \in \mathbb{R}^{N\times 3}$ and $Q \in \mathbb{R}^{N\times 3}$ as inputs to explain the problem.

For the first goal, we aim to predict a rigid transformation $\mathbf{T} = [\mathbf{R}, \mathbf{t}; \mathbf{0}, 1]$ with 6DoF, where $\mathbf{R} \in SO(3)$ is a rotation matrix and $\mathbf{t} \in \mathbb{R}^3$ is a translation vector. The rigid transformation transforms the point cloud Q to $\bar{Q}$ that has the closest distance to the ground-truth $\hat{Q}$. In this way, the following objective function is minimized:

$$\underset{\mathbf{R}\in SO(3), t\in\mathbb{R}^3}{\arg\min} \; d_{pc}\left(\hat{Q}, \mathbf{R}Q + \mathbf{t}\right), \tag{1}$$

where $d_{pc}(\cdot,\cdot)$ is a distance magnanimity of two point clouds.

The second goal is to predict the boundary between the two point clouds, *i.e.*, classifying each point in P (or $\hat{Q}$) as interior points or boundary points. In this way, the boundary prediction task can be formulated as:

$$\underset{B_P \subset P, B_Q \subset Q}{\arg\min} \; d_{pc}\left(B_P, \bar{Q}\right) + d_{pc}\left(P, \mathbf{R}B_Q + \mathbf{t}\right), \qquad (2)$$

where the $B_P \in \mathbb{R}^{N^B \times 3}$ and $B_Q \in \mathbb{R}^{N^B \times 3}$ are the boundary points extracted from the point cloud P and Q, respectively. Last but not least, the possibility that two pieces are matched can be easily determined by distances between transformed boundary points.

3.2. Our Approach

To tackle the above problem, we present a novel multi-task neural network that is composed of three main modules: one transformer-based point cloud encoder, and two decoder branches that infer transformation and boundary points, respectively.

Our pipeline is visualized in Fig. 2. Given the input point clouds, we first extract latent geometry features from each point cloud by applying a transformer-based encoder (see Sec. 4.1). Specifically, the encoder outputs a global feature for pose estimation and a local feature for boundary prediction. Next, different decoders are used in two separate branches to decode the features for particular sub-tasks, which will be described in Sec. 4.2 and Sec. 4.3. For the pose estimation, the global features of the input point clouds are concatenated and passed to a transformation decoding module, which consists of a couple of fully connected layers. This module directly learns a transformation matrix supervised by the ground-truth pose. For the boundary prediction, we consider using the local feature to recognize the explicit boundary points.

At last, we utilize an iterative greedy approach to assemble more than two pieces into a complete shape (Sec. 4.4). Given multiple pieces, we conduct pairwise matching by using our PuzzleNet to predict the transformation and boundary points between each pair. Then we select the best matching according to the distance between the boundary points, and the corresponding two point clouds are registered together as a new piece. This pairwise matching operation is repeated until all of the pieces are reassembled.

4. Method

4.1. Point Cloud Encoder

Since our method focuses on learning alignment from geometric information rather than semantic information, we compute geometric features for the input points by applying a point cloud encoder. Inspired by PCT [14], our encoder is built upon a transformer mechanism, which mainly consists of four components: feature mapping, sampling and grouping, attention-based feature augmentation, and feature aggregation. The detailed network architecture of this module is illustrated in Fig. 2.

As mentioned above, the goal of the point cloud encoder is to extract the local contextual feature F^l and the global contextual feature F^g which is used in the following two-branch decoders. To this end, we take both point clouds P and Q as input for two encoders respectively in each epoch. The input points are uniformly downsampled with $N = 1024$ by employing a farthest point sampling (FPS) strategy. Then two fully-connected layers with batch-norm and ReLU activation functions are utilized as feature mapping, and the points' coordinates are mapped to a high-dimensional space as embedding features $\mathcal{F}^l \in \mathbb{R}^{N \times 64}$. Different from the local feature (named as point feature in PCT) extracted from the vanilla PCT, our local features are extracted before the sampling modules with the purpose of avoiding messing up the back-propagation in our later multi-decoder network.

Second, two sampling and grouping (SG) modules are employed. In the first SG module, we downsample the input further by half to obtain P' (or Q') containing $N' = 512$ points. Then for each sampled point $\mathbf{p}'_i$, we consider its $K = 32$ nearest points $\mathbf{p}'_j$, and concatenate the point coordinates and the local features $\mathcal{F}^l_j$ to computer a new per-point feature $\mathcal{F}^{SG}_i \in \mathbb{R}^{K \times (64+3)}$. Therefore, the point cloud P' (or Q') can be represented as a tensor $\mathcal{F}^{SG}$ with the shape of $\mathbb{R}^{N' \times K \times (64+3)}$. Next, the concatenated feature $\mathcal{F}^{SG}$ is passed to two fully-connected layers with batch-norm and ReLU layers to remap the aggregated feature's dimension into $\mathbb{R}^{N' \times K \times 128}$. The newly created axis is then squeezed by max-pooling to gain the grouped feature with a shape of $\mathbb{R}^{N' \times 128}$. In the second SG module, the above feature grouping process is performed again and we obtain a new feature $\mathcal{F}^{SG} \in \mathbb{R}^{\frac{N'}{2} \times 256}$.

After SG modules, the above grouped feature $\mathcal{F}^{SG}$ is forward propagated by four offset-attention blocks to obtain four attention features $\{\mathcal{F}^{Att.}_i\}^4_{i=1}$. Then $\mathcal{F}^{SG}$ and the four attention features are concatenated in the feature channel and followed by a linear layer to align the length equal to input length $N = 1024$. Finally, the concatenated feature is fed into a max-pooling layer on the point-wise channel to obtain the global feature $F^g \in \mathbb{R}^{1024}$.

4.2. Transformation Estimation

The transformation estimation module is a core component of the proposed framework, which aims to infer the rigid pose $\mathbf{T}$ (*i.e.*, $\mathbf{R}$ and $\mathbf{t}$) that aligns the point cloud Q as close as ground-truth $\hat{Q}$. Different from traditional registration methods that rely on overlap points, our input point clouds are exactly disjoint without overlap regions, making it hard to align two point clouds by constructing valid

point-wise feature correspondences from only the contacting boundaries. Instinctively, we take full advantage of the extracted global features of the pieces and formulate the pose estimation as a regression problem:

$$\mathbf{S} = \mathcal{D}_{\mathbf{T}}(\mathcal{F}_P^g, \mathcal{F}_Q^g) \tag{3}$$

where $\mathcal{D}_{\mathbf{T}}(\cdot)$ is a transformation decoder, $\mathbf{S} \in \mathfrak{se}(3)$ represents the transformation tensor predicted by our network, and $\mathcal{F}_P^g$ and $\mathcal{F}_Q^g$ are the global feature of input point cloud P and Q, respectively.

Specifically, we concatenate the two global features as $\mathcal{F}^{cross}$ to capture cross-piece information. By feeding $\mathcal{F}^{cross}$ into the decoder, two fully-connected layers with batch-norm and ReLU layers are employed to predict a tensor $\mathbf{S} \in \mathfrak{se}(3)$ with a length of 6 representing the transformation $\mathbf{T}$. The tensor $\mathbf{S}$ are then converted to the rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$. The detailed parameters of this module can be referred to in Fig. 2.

Training loss. The pose estimation network is trained in a supervised fashion by an efficient joint loss function containing three components: mean squared error (MSE) loss, Chamfer distance (CD) loss and Earth mover's distance (EMD) loss:

$$L_{pose} = L_{MSE} + L_{CD} + L_{EMD}\,. \tag{4}$$

The MSE loss is used to evaluate the precision of the predicted transformation:

$$L_{MSE} = MSE\left(\mathbf{T} \cdot \bar{\mathbf{T}}^{-1}, \mathbf{I}\right), \tag{5}$$

where $\bar{\mathbf{T}} = [\bar{\mathbf{R}}, \bar{\mathbf{t}}; \mathbf{0}, 1]$ is the ground-truth transformation. If the pose is learnt correctly, the matrix production of predicted transformation $\mathbf{T}$ and ground-truth $\bar{\mathbf{T}}$ should be equal to the identity matrix $\mathbf{I}$.

The second term L_{CD} is the Chamfer distance loss, which is a regular metric to evaluate distance between two point clouds:

$$d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2, \tag{6}$$

where S_1, S_2 are two point sets. We use the Chamfer distance as a constraint between the ground-truth $\hat{Q}$ and the prediction $\bar{Q}$ as

$$L_{CD} = d_{CD}(\hat{Q}, \bar{Q}) \tag{7}$$

for the purpose of finding the transformation that converts the input Q to the ground-truth one.

The Earth mover's distance loss L_{EMD} is another loss function that is commonly used for point cloud registration. The EMD distance is based on finding the 1-1 correspondence between the two point sets, so that the sum of distances between corresponding points is minimal:

$$d_{EMD} = \min_{\phi: S_1 \to S_2} \sum_{x \in S_1} \|x - \phi(x)\|_2 \tag{8}$$

where ϕ is a bijection. Our L_{EMD} is then calculated by

$$L_{EMD} = d_{EMD}(\hat{Q}, \bar{Q}). \tag{9}$$

We train the pose estimation network with the loss function L_{pose} to optimize the Equation 1, aiming to minimize the distance between point clouds $\hat{Q}$ and $\bar{Q}$, as well as the distance between transformation matrix $\mathbf{T}$ and $\bar{\mathbf{T}}$.

4.3. Boundary Prediction

The boundary prediction module aims to extract the boundary points from each input point cloud. Note that we focus on the geometric alignment where we do not have particular semantic labels as in previous semantic segmentation methods, there is no clear criterion for deciding which points should be the boundary points. As the supervision on our neural network, we regard the closest N^B points to the input point clouds as ground-truth boundary parts. We set $N^B = 128$ by default in all of our experiments. Moreover, the boundary prediction should not have an influence on the back-propagation of transformation estimation encoding. To avoid messing up these two totally different functions in the encoding stage, the decoder for boundary predicting takes only two local features $\mathcal{F}_P^l$ and $\mathcal{F}_Q^l$ as inputs rather than the global context. We define the boundary prediction module by the following formulation:

$$\begin{aligned} B_P &= \mathcal{D}_{\mathbf{B}}^P(\mathcal{F}_P^l, \mathcal{F}_Q^l), \\ B_Q &= \mathcal{D}_{\mathbf{B}}^Q(\mathcal{F}_Q^l, \mathcal{F}_P^l), \end{aligned} \tag{10}$$

where $\mathcal{D}_{\mathbf{B}}^P$ and $\mathcal{D}_{\mathbf{B}}^Q$ are the decoders for predicting the boundary points.

To efficiently learn the local boundary information, it is crucial that both decoders should have the receptive field covering each other. Without loss of generality, we take the boundary prediction of point cloud P as an example to illustrate the prediction process, as indicated in Fig. 2. Our networks take the low-level local features $\mathcal{F}_P^l$ and $\mathcal{F}_Q^l$ as input. Feature $\mathcal{F}_Q^l$ with a shape of $\mathbb{R}^{N \times 64}$ is first remapped into a feature with the same shape by a fully-connected layer. The point channel of the new feature is zeroed by a max-pooling operation, and then repeated to form a high-level feature $\mathcal{F}_Q^h \in \mathbb{R}^{N \times 64}$ that depicts the shape of Q. By concatenating the low-level feature $\mathcal{F}_P^l$ of P and high-level feature $\mathcal{F}_Q^h$ of Q in the feature channel, we obtain a cross-geometry feature with a shape of $\mathbb{R}^{N \times 128}$. Next, linear layers with ReLU are employed to transform the cross-geometry feature into point-wise classification results $\mathcal{F}_P^B \in \mathbb{R}^{N \times 2}$, which infer the probabilities of a point belonging to the boundary or interior regions, respectively. Finally, we use a softmax function to obtain a 1-dimensional probability score vector, and then we select the first N^B points with the highest probabilities as boundary points B_P. The same operations are conducted on point cloud Q to obtain B_Q.

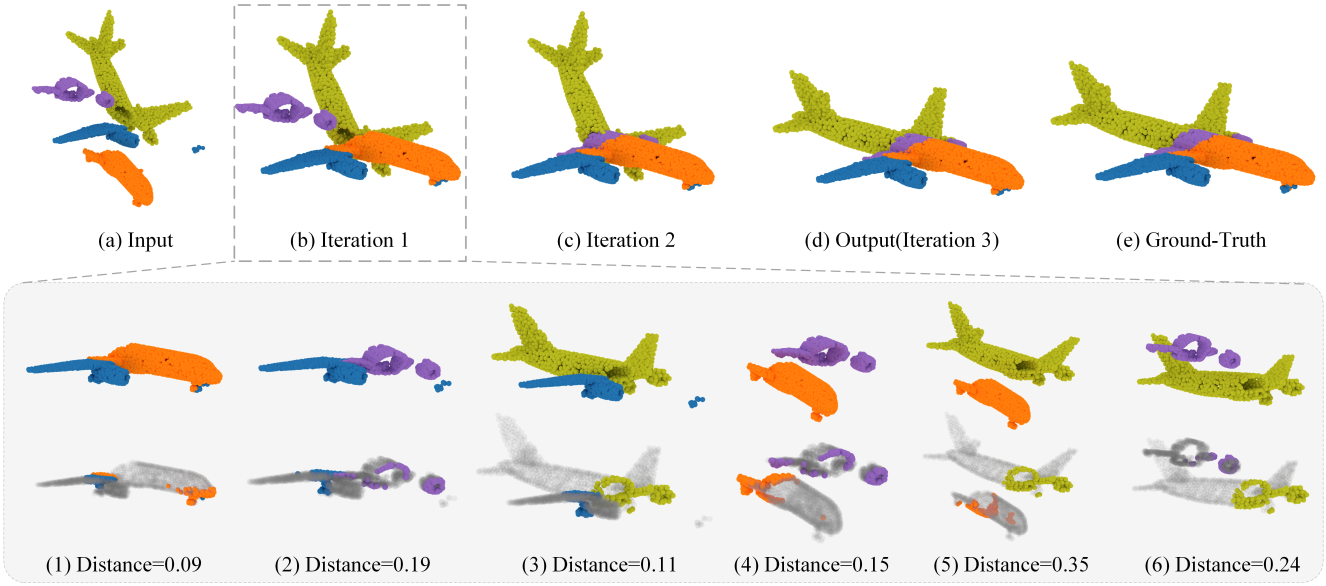

Figure 3: Detailed illustration of our multi-piece assembly process. Top row shows that four input pieces can be assembled into a complete shape after three iterations. In each iteration (bottom row), PuzzleNet is first applied to predict pairwise rigid pose and boundary points. Then we select the best alignment with the minimum distance between the boundary points. The corresponding two pieces are merged together and involved in later iterations.

To supervise the training of the boundary prediction network, we leverage the cross-entropy loss and Chamfer distance loss. We first use a cross-entropy to measure the loss between the predicted point-wise classification and the ground truth label:

$$L_{CE} = CE(B_P, \hat{B_P}) + CE(\hat{B_Q}, B_Q \cdot \mathbf{R} + \mathbf{t}), \quad (11)$$

where $\hat{B_P}$ and $\hat{B_Q}$ are the ground-truth boundary points. The Chamfer loss is used to restrict the distance of predicted boundary points from the ground truth:

$$L'_{CD} = d_{CD}(\hat{B_Q}, B_Q \cdot \mathbf{R} + \mathbf{t}). \quad (12)$$

The summation of L_{CE}, L'_{CD} are minimized to optimize the Equation 2.

4.4. Multi-piece Assembly

Thanks to the proposed multi-task neural network, we are able to not only conduct reasonable pairwise alignment but also predict the open boundaries. This allows us to easily generalize the pairwise matching to multi-piece assembly, which can not be achieved by previous learning-based assembly approaches [7].

Given n unaligned and roughly cut pieces of point clouds from a 3D shape, we first perform the pairwise matching among all the pieces. Then, we construct an undirected complete graph $\mathcal{G} = \{V, E\}$, whose nodes V are the collection of input pieces and edges E represent the matching relationships between the nodes. Initially, between each pair of nodes we build an edge as a candidate match by pairwise matching, and let $Dis(E_i)$ denote the weight of such an edge and $\mathbf{T}(E_i)$ its estimated transformation. The weight $Dis(E_i)$ is computed as the Chamfer distance of the predicted boundary points. Based on the assumption that incorrect matches lead to large transformation errors, we propose a greedy approach to iteratively compute a multi-piece matching. At each iteration, we first select the best pairwise matching that has the least weighted edge E_i. We merge the two nodes into a new one by registering the corresponding point clouds together based on the predicted $\mathbf{T}(E_i)$. Then the point cloud of the new node is re-sampled into $N = 1024$ points, and the graph $\mathcal{G}$ is updated by pairwise matching between the new node and other nodes. The above procedure is repeated until only one node in the graph, *i.e.*, all of the pieces have been reassembled into a complete shape. The numerical algorithm for multi-piece assembly is given in Algorithm 1, and Fig. 3 visually illustrates the reassembling process with a 4-pieces example.

5. Dataset

To train our PuzzleNet and demonstrate its efficacy, we utilize two kinds of datasets, including a real-world scanned dataset and a synthetic CAD dataset.

DublinCity dataset. The DublinCity constructed by Iman et al. [50] is an airborne LiDAR dataset with hierarchical labels, which is often used as a benchmark for evaluating point cloud segmentation or classification algorithms. The

Algorithm 1 Mulit-pieces Assembly

Input: n pieces of point clouds $S_P = \{P_1, P_2 \dots P_n\}$
Output: Point cloud S'_P representing the reassembled complete shape

$n_S \leftarrow 0$
for $P_i \in S_P$ **do**
  $S_i \leftarrow$ MAKE-SET(P_i)
  $n_S \leftarrow n_S + 1$
end for
$S_P \leftarrow \{S_1, S_2, \dots, S_i, \dots S_n\}$
while $n_S > 1$ **do**
  $D_i^j \leftarrow \{\}$
  for $S_i \leftarrow \{S_1, \dots, S_{n-1}\}$ **do**
    for $S_j \leftarrow \{S_{i+1}, \dots, S_n\}$ **do**
      $\mathbf{T}_i^j, d_i^j \leftarrow$ PuzzleNET(FPS(S_i), FPS(S_j))
      D_i^j.add(d_i^j)
    end for
  end for
  $S_i, S_j \leftarrow min(\{D_i^j\})$
  S_P.pop(S_i); S_P.pop(S_j)
  S_P.add(UNION(S_i, $S_j \cdot \mathbf{T}_i^j$))
  $n_S \leftarrow n_S - 1$
end while
$S'_P \leftarrow S_P$.pop(S_1)
return S'_P

objects of the city are classified into 4 categories: *building, ground, vegetation, and undefined*. Among them, the building is the most complicated category with diverse types of historic and modern urban elements, such as offices, shops, libraries, and residential houses. Specifically, the facade and roof are separated and attached with different labels manually as two disjoint parts of the buildings. This characteristic makes the dataset suitable for testing our PuzzleNet.

To prepare the dataset, we manually extract all the buildings and then segment 3D building instances based on their semantic labels. Each building is assembled by two parts which represent the facade and roof, respectively. After filtering the outliers, there remain about 600 buildings, which are randomly split into training (400), validation (100) and test (100) sets. All of the point clouds are downsampled by FPS with $N = 1024$ points and normalized into a zero-centered unit sphere $[0, 1]^3$.

Without the loss of generality, point clouds with label *Facade* are regarded as P, while point clouds with label *Roof* are treated as Q. Then during training, each point cloud Q is rotated and translated on the fly with a random transformation matrix $\mathfrak{se}(3)$. The rotation and translation are initialized to range $[0, 60°]$ and $[0, 1.0]$ following [22]. Note that although we have the semantic labels of each part, in our neural network we do not require semantic information and only focus on geometric alignment.

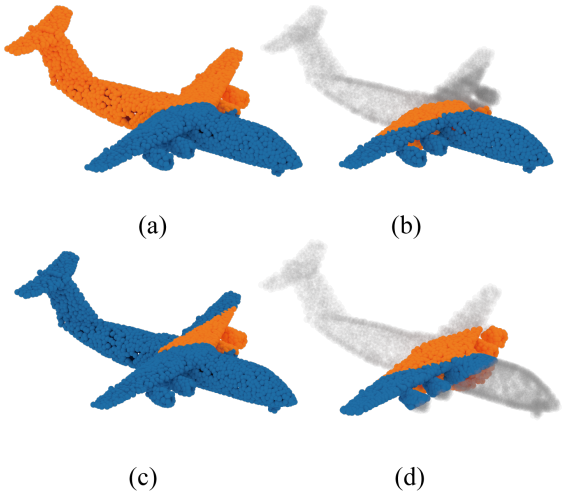

Figure 4: Four basic cutting cases that can represent the most pairwise matching relationships among multi-piece data. The detailed illustration is described in Sec. 5.

ModelNet40 dataset. We also evaluate our method on a CAD dataset, *i.e.*, ModelNet40 [39], from which we choose two categories of objects: 726 airplanes, and 608 beds. For the pairwise assembling task, we first build a dataset in a *two-piece mode*. We sample a dense point cloud from the original CAD models and normalize the sampled points into a unit sphere. To cut each point cloud, we use different 3D primitives that are randomly placed in a unit sphere to cut the point cloud into two pieces. We define four different cutting types, each of which is represented by a specific primitive, including *plane, sphere, cylinder, and cone*. After cutting, we uniformly sample each piece and conduct a random transformation as described in the DublinCity dataset. Finally, we select 100 models as the test set for each category, and the remaining models are used as training and validation sets with a rate of 9:1.

In the *multi-piece mode*, as a 3D shape may be cut into much smaller pieces revealing less global geometric features, we should train our network by extending the two-piece dataset to the multi-piece one. We adopt a coarse-to-fine cutting strategy. As shown in Fig. 4, we consider four basic cutting cases that can represent the most pairwise matching relationships among multi-piece data: (a) The most common case is that the whole shape is split into two pieces. (b) The whole shape is first cut to obtain two pieces, then we randomly discard one piece (indicated by the gray color in the figure) and cut the other piece further into two smaller parts. (c) The whole shape is first cut twice randomly to obtain four pieces. After that, we merge three random neighboring pieces into a larger one. Thus, we finally also have two pieces. (d) The whole shape is also cut

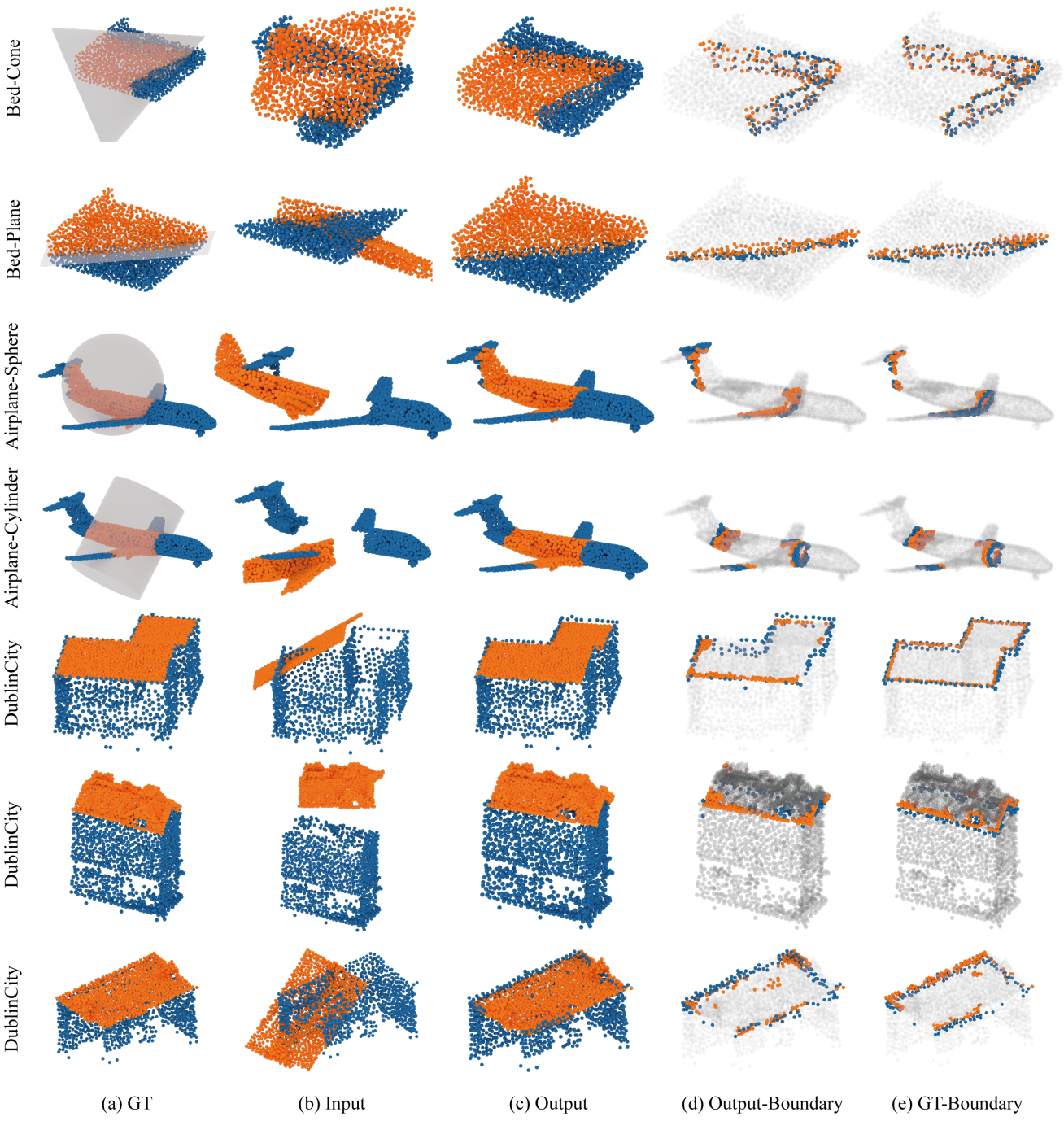

Figure 5: Visual results of pairwise shape alignment using the bed, airplane and Dublin buildings. For bed and airplane models, we show the cutting types using different 3D primitives (cone, Plane, sphere and cylinder) which are drawn in transparent gray color. Besides, in (d) and (e) we show the predicted boundaries and ground-truth boundary points.

twice to obtain four pieces. Then we select the two middle neighboring pieces while discarding the other two. Initially, these four cases are selected with the same probability. However, cases (c) and (d) are more complicated considering the 3D primitives are randomly posed, so we first try five times in these cases. If the number of sampled points is still less than 1024, we go back to the case (a) to speed up the sampling steps in every epoch. Finally, the ratios of four cases are about 35%, 25%, 20%, 20% respectively. Overall, after simulating the above pairwise matching cases, our

PuzzleNet is still only trained by taking two pieces as inputs in each epoch.

6. Experimental Results

6.1. Experimental Setup

Implementation details. Our PuzzleNet is implemented in Pytorch on a single Nvidia RTX 2080Ti (11GB memory) graphics card. We train the neural network on a two-piece dataset for 5000 epochs, and on a multi-piece dataset for 10000 epochs with an Adam optimizer. The initial learning rate is set to 10^{-3} and reduced with an attenuation coefficient of 0.99 every 30 steps.

Evaluation metrics. To evaluate the accuracy of our estimated transformation, we use several commonly used metrics [35, 43] to measure the difference between ground-truth $\bar{\mathbf{R}}$ and $\bar{\mathbf{t}}$ and our predicted $\mathbf{R}$ and $\mathbf{t}$. We first compute the mean isotropic rotation and translation errors by:

$$R_{iso} = \arccos\left((Tr(\bar{\mathbf{R}}^{-1}\mathbf{R}) - 1)/2\right), \quad t_{iso} = \|\bar{\mathbf{R}}^{-1}\mathbf{t} - \bar{\mathbf{t}}\|_1, \tag{13}$$

where $Tr(\mathbf{M})$ returns the trace of a matrix $\mathbf{M}$, R_{iso} measures the minimum rotation angle required to align the rotations ($\bar{\mathbf{R}}$ and $\mathbf{R}$) of two poses. In addition, anisotropic metrics are calculated in the forms of *mean squared error* (mse) and *mean absolute error* (mae) to evaluate the absolute errors over Euler angles and translation vectors:

$$\begin{aligned} R_{mse} &= MSE(Euler(\bar{\mathbf{R}}), Euler(\mathbf{R})), \\ R_{mae} &= MAE(Euler(\bar{\mathbf{R}}), Euler(\mathbf{R})), \\ t_{mse} &= MSE(\bar{\mathbf{t}}, \mathbf{t}), \quad t_{mae} = MAE(\bar{\mathbf{t}}, \mathbf{t}), \end{aligned} \tag{14}$$

where $MAE(\cdot)$ and $MSE(\cdot)$ are two functions to calculate mean absolute errors and mean squared errors, respectively. $Euler(\cdot)$ calculates the corresponding Euler angles from a rotation matrix.

To determine the accuracy of boundary prediction, we adopt the metric of point-wise *intersection over union* (IoU) which is widely used in 3D part-segmentation works. Besides, we also use the Chamfer distance to evaluate the distance between the predicted boundaries B_P, B_Q, and ground-truth boundary points $\hat{B_P}, \hat{B_Q}$:

$$CD_P = d_{CD}(B_P, \hat{B_P}), \quad CD_Q = d_{CD}(B_Q, \hat{B_Q}) \tag{15}$$

6.2. Self Evaluation

Pairwise shape alignment. We first evaluate the performance of our PuzzleNet on pairwise geometric shape alignment. Fig. 5 shows the visual shape alignment results using several CAD models (airplane and bed) and Dublin buildings, where we cut the CAD models using different primitives. For each example, we also draw the predicted boundaries to show whether our approach is able to learn effective boundary features. For quantitative evaluation, Fig. 6 reports the recall as the ratio of successful alignment, where a transformation is accepted as positive if the rotation and translation errors are within the thresholds ξ_r and ξ_t. Table 1 numerically evaluates the accuracy of boundary prediction. As shown in the figure and table, our PuzzleNet learns reasonable boundary predictions, and successfully aligns the shape pieces that have no overlaps or semantic information.

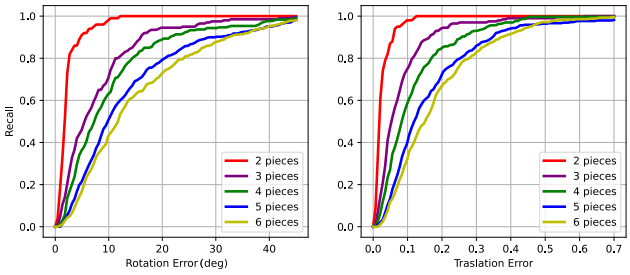

Figure 6: Transformation estimation performance of our PuzzleNet for multi-piece assembly. We show the recall as a function of rotation and translation errors, where the X-axis represents the thresholds ξ_r and ξ_t, respectively.

Table 1: Numerical statistics of the accuracy of boundary prediction tested on the DublinCity dataset and the airplane dataset under different cut types.

Dataset	$IoU_P \uparrow$	$IoU_Q \uparrow$	$CD_P \downarrow$	$CD_Q \downarrow$
DublinCity	0.3258	0.1784	0.0233	0.4016
Plane cut	0.5522	0.6640	0.0023	0.4601
Cylinder cut	0.7388	0.7513	0.0027	0.3788
Cone cut	0.6429	0.7359	0.0214	0.4407
Sphere cut	0.4949	0.7212	0.0260	0.3899

Table 2: The recall performance (as a function of rotation and translation error) on several specific thresholds.

pieces	$R_{err} = 15$	$t_{err} = 0.15$
2	1.0	0.9800
3	0.8700	0.7600
4	0.8100	0.5833
5	0.6907	0.4046
6	0.6283	0.3238

Multi-piece assembly. The ability of assembling multiple point clouds is gained by predicting the transformation and the boundary information simultaneously, and then choosing the most reasonable one from many candidates by comparing the distance between the boundaries of each pair. We train PuzzleNet only on the 4-pieces dataset and test it on other multi-piece cases. Figures 8 and 6 display the visual assembling and quantitative evaluation results, respectively.

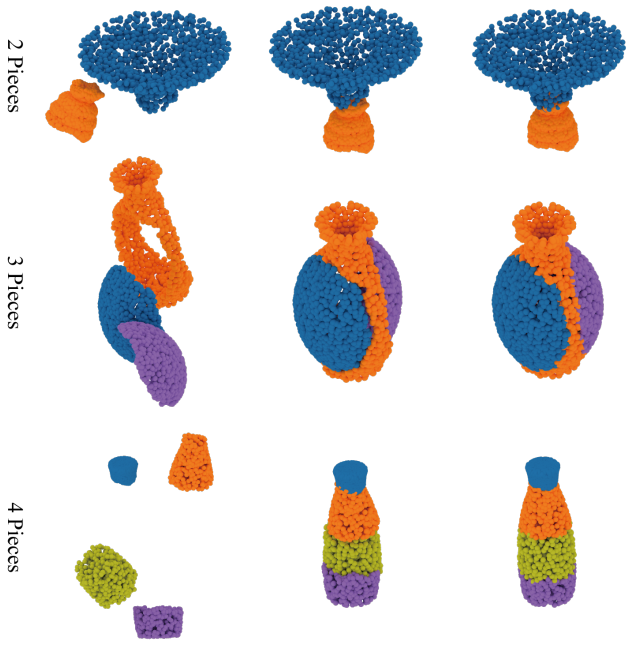

Figure 7: PuzzleNet can be applied to assemble 3D shapes that are broken by different cutting approaches, such as using the realistic breaking simulation method [34].

Table 2 reports the recall performance value on several specific thresholds. Both of them demonstrate the robustness of our PuzzleNet against a different number of input pieces. Furthermore, to test the robustness of PuzzleNet against different cutting types, we select the category of the vase from ModelNet40 and train PuzzleNet on the dataset constructed by using our cutting approach (Sec. 5). To prepare the testing data, we simulate the realistic breaking results of the vase models by using the method proposed by [34] to simulate the object's most natural ways of breaking. Thus the broken way of such models is different from our training data. Fig. 7 shows the assembling results of our PuzzleNet, indicating that we still achieve promising outcomes.

6.3. Comparisons

We compare our method against both registration and shape mating methods. We select a representative global registration methods, *Fast Global Registration* (FGR) [48], and a learning-based low-overlap registration method, Predator [21]. Then the *Neural Shape Mating* (NSM) [7] is used as the competitor that is most relevant to ours. For a fair comparison, we re-train NSM on our both airplane and DublinCity datasets. Note that as Predator still relies on overlap information, we failed to train a valid network on our dataset. Thus, we use the officially released Predator model with weights trained on ModelNet40, and we compare with it on airplane shapes. Furthermore, we only compare with those methods on pairwise registration/mating, because they can not handle multi-piece assembly.

Table 3: Quantitative comparison of different methods for pairwise shape alignment. We test the methods on the DublinCity dataset and the airplane dataset which is cut using different 3D cutting primitives.

Dataset	Method	$R_{iso}\downarrow$	$t_{iso}\downarrow$	$R_{mse}\downarrow$	$R_{mae}\downarrow$	$t_{mse}\downarrow$	$t_{mae}\downarrow$
DublinCity	FGR	39.8084	0.8332	770.4874	20.1878	0.2660	0.4169
	NSM	29.5259	0.3260	324.4744	14.6914	0.0398	0.1626
	PuzzleNet	**7.2873**	**0.0764**	**30.402**	**3.6528**	**0.0028**	**0.0379**
Plane cut	FGR	79.0867	0.9084	4452.1583	41.8482	0.3144	0.4592
	predator	71.1062	0.5575	2750.7584	37.3222	0.1331	0.2510
	NSM	25.6709	0.4004	252.2779	12.7266	0.0609	0.1958
	PuzzleNet	**2.2049**	**0.0251**	**2.9568**	**1.0631**	**0.0004**	**0.0121**
Cylinder cut	FGR	65.2692	0.8421	3590.8886	36.4658	0.2667	0.4215
	predator	60.5815	0.5477	1722.9497	30.8506	0.1201	0.2486
	NSM	31.4569	0.2999	366.0138	16.0299	0.0326	0.1505
	PuzzleNet	**1.6524**	**0.0172**	**2.3867**	**0.8269**	**0.0003**	**0.0085**
Cone cut	FGR	49.1665	0.8197	2081.8663	27.3513	0.2613	0.4062
	predator	60.6906	0.6908	1792.4524	31.8404	0.1866	0.3113
	NSM	27.2779	0.3786	278.4967	13.7801	0.0560	0.1906
	PuzzleNet	**3.1074**	**0.0312**	**5.2512**	**1.5822**	**0.0006**	**0.0155**
Sphere cut	FGR	87.4699	0.7973	4783.8691	43.8576	0.2446	0.3984
	predator	65.0936	0.5627	2079.1263	31.7647	0.1359	0.2549
	NSM	31.7505	0.3219	376.8902	15.8825	0.0396	0.1617
	PuzzleNet	**2.2254**	**0.0199**	**3.6674**	**1.0985**	**0.0003**	**0.0101**

Table 4: Quantitative evaluation of different loss functions on the effect of transformation estimation. This ablation study is conducted on the category of airplane cut by using the plane primitive.

L_{MSE}	L_{CD}	L_{EMD}	$R_{mse}\downarrow$	$t_{mse}\downarrow$	$R_{iso}\downarrow$	$t_{iso}\downarrow$
✓			4.0198	0.0007	1.9487	0.0295
	✓		9.3231	0.0013	2.8730	0.0422
		✓	4.6257	0.0005	2.4934	0.0258
✓	✓		3.1079	0.0005	**1.7379**	0.0281
✓	✓	✓	**2.9568**	**0.0004**	2.2049	**0.0251**

Figures 9 and 10 show the qualitative comparison results on airplane and DublinCity, respectively. Previous registration methods fail when matching geometric features without overlapping regions. Instead, they attempt to overlay pieces by trying to find overlapping point-pairs for further optimization. Compared to NSM, our PuzzleNet achieves more accurate shape mating results. The quantitative comparison is provided in Table 3. We outperform all the handcrafted and learned methods by a large margin in both rotation and translation prediction.

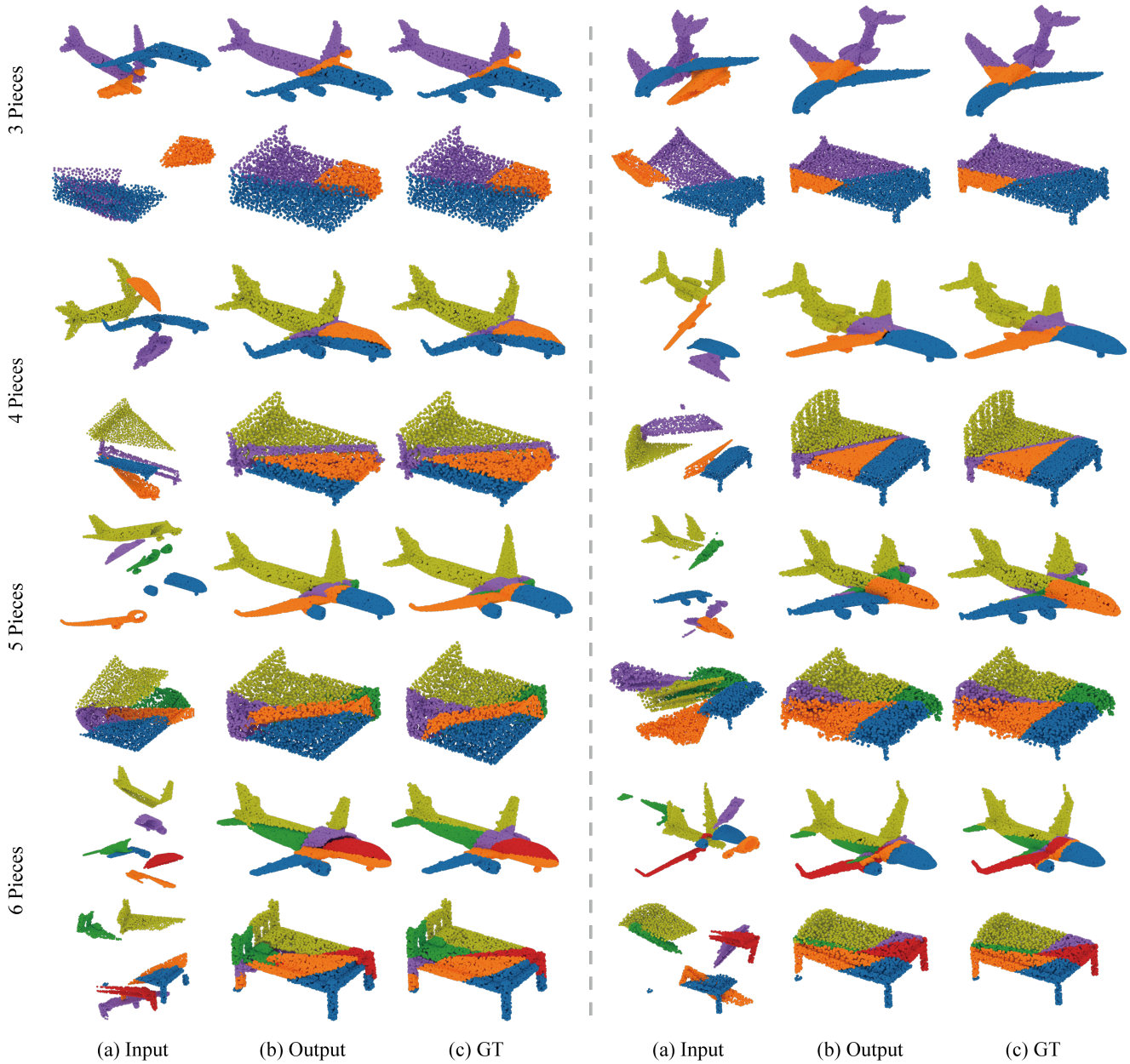

Figure 8: Visual results of our multi-pieces assembly. We test our approach on CAD models where the number of input pieces ranges from 3 to 6. For each example, we select two airplanes and two beds, and show the input point clouds, our assembling results and the ground-truth, respectively.

Table 5: Ablation study of our multi-task neural network on airplane dataset.

Trans.Branch	Bound.Branch	$R_{iso} \downarrow$	$t_{iso} \downarrow$	$IoU_P \uparrow$	$IoU_Q \uparrow$
✓		2.9178	0.0324		
	✓			0.5653	0.7169
✓	✓	2.2049	0.0251	0.5522	0.6640

6.4. Ablation Study

We present ablation studies to verify the effectiveness of the designed framework.

Loss functions. We first evaluate the effectiveness of different losses used in our network. Without loss of generality, we use the category of airplane cut by the plane primitive for conducting the ablation study.

Table 4 shows the accuracy of transformation estimation

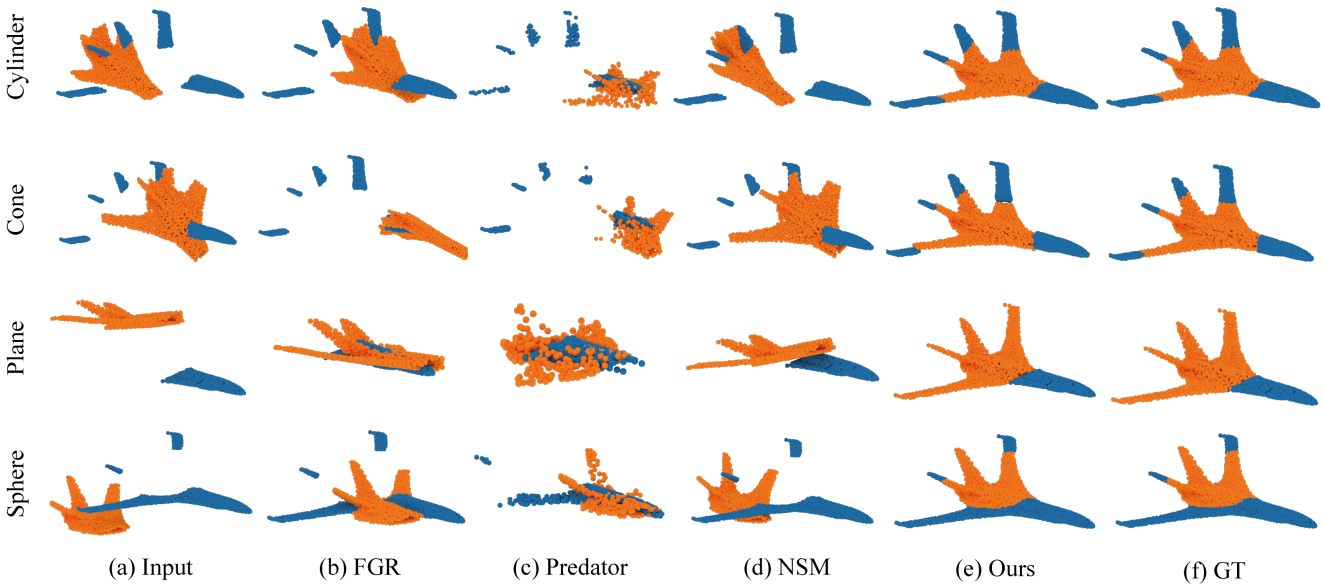

Figure 9: Visual comparison of our approach with FGR [48], Predator [21] and NSM [7] for pairwise shape mating. We test the methods on the category of airplane which is cut using different 3D cutting primitives.

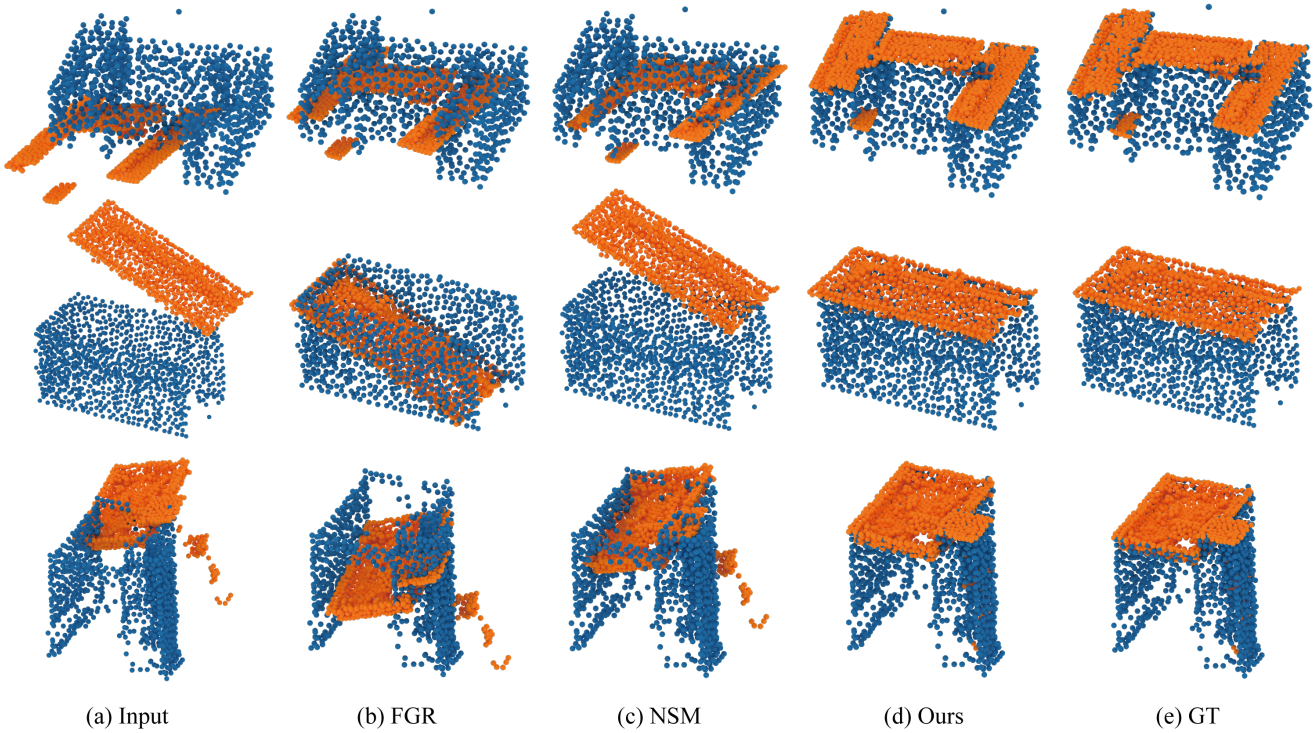

Figure 10: Visual comparison of our approach with FGR [48] and NSM [7] using DublinCity dataset.

of our PuzzlNet which is trained with different combinations of losses. As we can see, the L_{MSE} loss plays an important role in reducing rotation errors while L_{EMD} achieves a better translation accuracy. Although using L_{CD} alone can not obtain satisfactory results, it enhances the capability of improving both rotation and translation accuracy. Overall, the combination of three losses achieves the best performance.

(a) Input (b) Output (c) GT

Figure 11: An example of a 7-piece assembly, where the registration error will become larger as the number of input pieces increases.

Multi-task network. To prove that the two decoders in Fig. 2 do not affect each other, we train two independent networks for transformation and boundary prediction, respectively. All networks are trained using the same point cloud encoder, parameters, and epochs. The performance of transformation and boundary prediction is summarized in Table 5. In terms of rotation and translation errors, the transformation estimation results of our proposed joint network are better than only training a transformation network. At the same time, we do not reduce the accuracy of boundary prediction too much (see IoU_P and IoU_Q in Table 5).

Since the two decoders in our joint neural network use the same point cloud encoder, they can simultaneously have an influence on the weights of the encoder. Therefore, by balancing the two decoders, the results of our network are more stable compared with the two independent networks. In addition, as we propose an end-to-end solution, our proposed method is also more efficient than training two independent networks.

7. Conclusion and Future Work

We have advanced 3D shape assembly by introducing PuzzleNet, a deep neural network for learning to assemble non-overlapping point clouds by using purely geometric information. Inspired by puzzle solving, PuzzleNet learns accurate pairwise alignment by inferring the rigid transformation and boundary features simultaneously, where two decoders are carefully designed in a unified network to remain unaffected by each other. Benefiting from the boundary feature learning and matching, the multi-piece assembly can be easily achieved by an iterative greedy algorithm. We have demonstrated the effectiveness and advantages of our approach by comparing it with state-of-the-art methods on synthetic and real-scanned datasets.

Limitations and future work. While producing convincing results, our framework still has several limitations. First, our multi-piece reassembling method relies on minimizing the errors between each pair of pieces locally. Thus, the deviation will be accumulated in the process of reassembling point clouds piece-by-piece, taking Fig. 11 as an example. To tackle this problem, a global optimization approach is required, which is still not trivial considering the non-overlap scenarios. Another possible solution for future study would be to extend PuzzleNet to learn end-to-end multi-pieces assembly. Another limitation is that we may not properly handle pieces with similar cutting boundaries. This case can cause ambiguous piece assembly and requires global information to determine which one is optimal. Also, like most learning-based methods for shape analysis, PuzzleNet is still category-specific and relies on the pre-collected and labeled dataset. That is to say, the neural model trained on one category cannot be used for assembling the shapes of another category. In the future, we would like to study real-world multi-piece assembling by expanding our dataset with more realistic breaking shapes of complex and arbitrary cutting boundaries.

Acknowledgments

This work is partially funded by the National Natural Science Foundation of China (U22B2034, 62172416, U21A20515, 62172415, 62271467), and the Youth Innovation Promotion Association of the Chinese Academy of Sciences (2022131).

References

[1] D. Aiger, N. J. Mitra, and D. Cohen-Or. 4-points congruent sets for robust pairwise surface registration. In *ACM Trans. Graph. (Proc. SIGGRAPH)*, pages 1–10. 2008. 3

[2] Y. Aoki, H. Goforth, R. A. Srivatsan, and S. Lucey. Pointnetlk: Robust & efficient point cloud registration using pointnet. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 7163–7172, 2019. 3

[3] P. J. Besl and N. D. McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606, 1992. 3

[4] S. D. Billings, E. M. Boctor, and R. H. Taylor. Iterative most-likely point registration (imlp): A robust algorithm for computing optimal shape alignment. *PloS one*, 10(3):e0117688, 2015. 3

[5] A.-Q. Cao, G. Puy, A. Boulch, and R. Marlet. PCAM: Product of Cross-Attention Matrices for Rigid Registration of Point Clouds. In *IEEE International Conference on Computer Vision (ICCV)*, 2021. 3

[6] S. Chen, L. Nan, R. Xia, J. Zhao, and P. Wonka. Plade: A plane-based descriptor for point cloud registration with small overlap. *IEEE Trans. Geosci. Remote. Sens.*, 2019. 3

[7] Y.-C. Chen, H. Li, D. Turpin, A. Jacobson, and A. Garg. Neural shape mating: Self-supervised object assembly with adversarial shape priors. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 6, 10, 12

[8] H. Deng, T. Birdal, and S. Ilic. Ppfnet: Global context aware local features for robust 3d point matching. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 195–205, 2018. 3

[9] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image

analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 3

[10] T. A. Funkhouser, M. M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, and D. P. Dobkin. Modeling by example. *ACM Trans. Graph.*, 23(3):652–663, 2004. 1

[11] N. Gelfand, L. Ikemoto, S. Rusinkiewicz, and M. Levoy. Geometrically stable sampling for the icp algorithm. In *International Conference on 3-D Digital Imaging and Modeling*, pages 260–267, 2003. 3

[12] Z. Gojcic, C. Zhou, J. D. Wegner, and A. Wieser. The perfect match: 3d point cloud matching with smoothed densities. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 5545–5554, 2019. 3

[13] J. Guo, X. Xing, W. Quan, D.-M. Yan, Q. Gu, Y. Liu, and X. Zhang. Efficient center voting for object detection and 6d pose estimation in 3d point cloud. *IEEE Trans. Image Process.*, 30:5072–5084, 2021. 3

[14] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7(2):187–199, 2021. 3, 4

[15] A. N. Harish, R. Nagar, and S. Raman. RGL-NET: A recurrent graph learning framework for progressive part assembly. In *IEEE/CVF Winter Conference on Applications of Computer Vision, WACV*, pages 647–656, 2022. 2

[16] J. H. Hong, S. J. Yoo, M. A. Zeeshan, Y. M. Kim, and J. Kim. Structure-from-sherds: Incremental 3d reassembly of axially symmetric pots from unordered and mixed fragment collections. In *IEEE International Conference on Computer Vision (ICCV)*, pages 5423–5431, 2021. 2

[17] S.-M. Hu, D. Liang, G.-Y. Yang, G.-W. Yang, and W.-Y. Zhou. Jittor: a novel deep learning framework with meta-operators and unified graph execution. *Science China Information Sciences*, 63(12):1–21, 2020. 3

[18] H. Huang, M. Gong, D. Cohen-Or, Y. Ouyang, F. Tan, and H. Zhang. Field-guided registration for feature-conforming shape composition. *ACM Trans. Graph.*, 31:171:1–171:11, 2012. 2

[19] J. Huang, G. Zhan, Q. Fan, K. Mo, L. Shao, B. Chen, L. Guibas, and H. Dong. Generative 3d part assembly via dynamic graph learning. In *The IEEE Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 2

[20] Q. Huang, S. Flöry, N. Gelfand, M. Hofer, and H. Pottmann. Reassembling fractured objects by geometric matching. *ACM Trans. Graph.*, 25(3):569–578, 2006. 1, 2

[21] S. Huang, Z. Gojcic, M. Usvyatsov, and K. S. Andreas Wieser. Predator: Registration of 3d point clouds with low overlap. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2021. 2, 3, 10, 12

[22] X. Huang, G. Mei, and J. Zhang. Feature-metric registration: A fast semi-supervised approach for robust point cloud registration without correspondences. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, June 2020. 7

[23] X. Huang, G. Mei, J. Zhang, and R. Abbas. A comprehensive survey on point cloud registration. *arXiv preprint arXiv:2103.02690*, 2021. 2

[24] J. P. Iglesias, C. Olsson, and F. Kahl. Global optimality for point set registration using semidefinite programming. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 8287–8295, 2020. 3

[25] B. Jones, D. Hildreth, D. Chen, I. Baran, V. G. Kim, and A. Schulz. Automate: A dataset and learning approach for automatic mating of cad assemblies. *ACM Trans. Graph.*, 40(6), 2021. 2

[26] R. K. Jones, T. Barton, X. Xu, K. Wang, E. Jiang, P. Guerrero, N. Mitra, and D. Ritchie. Shapeassembly: Learning to generate programs for 3d shape structure synthesis. *ACM Trans. Graph.*, 39(6):Article 234, 2020. 2

[27] Y. Lee, E. S. Hu, and J. J. Lim. IKEA furniture assembly environment for long-horizon complex manipulation tasks. In *IEEE International Conference on Robotics and Automation, ICRA*, pages 6343–6349. IEEE, 2021. 2

[28] Y. Li, K. Mo, L. Shao, M. Sung, and L. Guibas. Learning 3d part assembly from a single image. *European Conference on Computer Vision (ECCV)*, 2020. 2

[29] N. Mellado, D. Aiger, and N. J. Mitra. Super 4pcs fast global pointcloud registration via smart indexing. In *Comput. Graph. Forum*, volume 33, pages 205–215, 2014. 3

[30] K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and H. Su. PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[31] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 3

[32] Z. Qin, H. Yu, C. Wang, Y. Guo, Y. Peng, and K. Xu. Geometric transformer for fast and robust point cloud registration. *arXiv preprint arXiv:2202.06688*, 2022. 2, 3

[33] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*, pages 145–152, 2001. 3

[34] S. Sellán, J. Luong, L. M. Da Silva, A. Ramakrishnan, Y. Yang, and A. Jacobson. Breaking good: Fracture modes for realtime destruction. *ACM Trans. Graph.*, 2022. 10

[35] Y. Wang and J. Solomon. Deep closest point: Learning representations for point cloud registration. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3522–3531, 2019. 3, 9

[36] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Trans. Graph.*, 38(5):1–12, 2019. 3

[37] K. D. Willis, P. K. Jayaraman, H. Chu, Y. Tian, Y. Li, D. Grandi, A. Sanghi, L. Tran, J. G. Lambourne, A. Solar-Lezama, and W. Matusik. Joinable: Learning bottom-up assembly of parametric cad joints. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[38] K. Wu, X.-M. Fu, R. Chen, and L. Liu. Survey on computational 3d visual optical art design. *Visual Computing for Industry, Biomedicine, and Art*, 5(31):1–31, 2022. 1

[39] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric

shapes. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015. 7

[40] X. Xing, J. Guo, L. Nan, Q. Gu, X. Zhang, and D.-M. Yan. Efficient mspso sampling for object detection and 6-d pose estimation in 3-d scenes. *IEEE Transactions on Industrial Electronics*, 69(10):10281–10291, 2022. 3

[41] H. Xu, S. Liu, G. Wang, G. Liu, and B. Zeng. Omnet: Learning overlapping mask for partial-to-partial point cloud registration. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3132–3141, 2021. 3

[42] Z. J. Yew and G. H. Lee. 3dfeat-net: Weakly supervised local 3d features for point cloud registration. In *European Conference on Computer Vision (ECCV)*, pages 607–623, 2018. 3

[43] Z. J. Yew and G. H. Lee. Rpm-net: Robust point matching using learned features. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2020. 9

[44] K. Yin, Z. Chen, S. Chaudhuri, M. Fisher, V. Kim, and H. Zhang. Coalesce: Component assembly by learning to synthesize connections. In *Proc. of 3DV*, 2020. 2

[45] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 1802–1811, 2017. 3

[46] K. Zhang, W. Yu, M. Manhein, W. Waggenspack, and X. Li. 3d fragment reassembly using integrated template guidance and fracture-region matching. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2138–2146, 2015. 2

[47] L. Zhang, J. Guo, Z. Cheng, J. Xiao, and X. Zhang. Efficient pairwise 3-d registration of urban scenes via hybrid structural descriptors. *IEEE Trans. Geosci. Remote. Sens.*, 60:1–17, 2022. 3

[48] Q.-Y. Zhou, J. Park, and V. Koltun. Fast global registration. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *European Conference on Computer Vision (ECCV)*, pages 766–782, 2016. 3, 10, 12

[49] L. Zhu, H. Guan, C. Lin, and R. Han. Neighborhood-aware geometric encoding network for point cloud registration. *arXiv preprint arXiv:2201.12094*, 2022. 3

[50] I. Zolanvari, S. Ruano, A. Rana, A. Cummins, A. Smolic, R. Da Silva, and M. Rahbar. Dublincity: Annotated lidar point cloud and its applications. 09 2019. 6