NEURAL SHAPE MATING: SELF-SUPERVISED OBJECT ASSEMBLY WITH ADVERSARIAL SHAPE PRIORS

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning to autonomously assemble shapes is a crucial skill for many robotic applications. Whereas the majority of existing part assembly methods focus on correctly posing semantic parts to recreate a whole object, we interpret assembly more literally: as mating geometric parts together to achieve a snug fit. By focusing on shape alignment, rather than semantic cues, we can achieve across category generalization and scaling. In this paper, we introduce a novel task, pairwise 3D *geometric* shape assembly, and propose Neural Shape Mating (NSM) to tackle this problem. Given point clouds of two object parts, NSM learns to reason about their geometric structure and fit in order to predict a pair of 3D poses that tightly mate them together. In addition, we couple the training of NSM with an implicit shape reconstruction task, making NSM more robust to imperfect point cloud observations. To train NSM, we present a self-supervised data collection pipeline that generates pairwise shape assembly data with ground truth by randomly cutting an object mesh into two parts, resulting in a dataset that consists of 19,226 shape assembly pairs with numerous object meshes and diverse cut types. We train NSM on the collected dataset and compare it with several point cloud registration methods and one part assembly baseline approach. Extensive experimental results and ablation studies under various settings demonstrate the effectiveness of the proposed algorithm. Additional material available at: neural-shape-mating.github.io.

1 INTRODUCTION

The human-built world is filled with objects that are shaped to fit, snap, connect, or mate together. Consider joining a broken figurine, or putting a pen in a cap, or inserting a plug into a socket are all instances of geometric mating. To achieve object mating, humans rely on the ability to perceive and reason about the fit between shapes. However, it is worth noting that while semantics of the output may improve the solution, semantic understanding is not a necessacity to find geometric solutions. This kind of geometric reasoning for pairwise object mating is foundational for 3D reasoning and appears in many domains ranging from computer graphics (Li et al., 2012), animation, 3D design (Chen et al., 2015; Jacobson, 2017), and embodied interaction in robotics.

Many attempts to learn an algorithmic variant of this geometric reasoning have been attempted in application specific domains: assembling a car or furniture (Lee et al., 2019), object assembly (Agrawala et al., 2003), object packing (Wang & Hauser, 2019) and kitting (Zakka et al., 2020). Most of these assembly algorithms rely on the semantic of each component to bring different parts together (Lee et al., 2019). While promising results have been demonstrated, heavily relying on semantic information (e.g., part segmentation), target shapes (Li et al., 2020) as guidance, and ground-truth part pose annotation (Li et al., 2020; Huang et al., 2020) makes these methods application specific, hard to scale, and difficult to generalize.

In this paper, we study the problem of 3D-assembly of unknown objects, where we focus on pairwise shape mating without having semantic information or target shapes as guidance. As shown in Figure 1, given a pair of shapes in the form of point clouds with random intial poses, we aim to find a suitable mating configuration for them using geometric cues. The proposed task is more challenging yet practical with potential applications in CAD and graphics for fracture reassembly (Zhang et al., 2015) (e.g., reassembling fragments of a broken vase, where each fragment does not have a well defined semantic meaning), robotic reassembly and object packing (Wang & Hauser, 2019; Goyal & Deng, 2020) (e.g., packing objects into a box, where the object arrangement requires reasoning about the fit between adjacent objects).

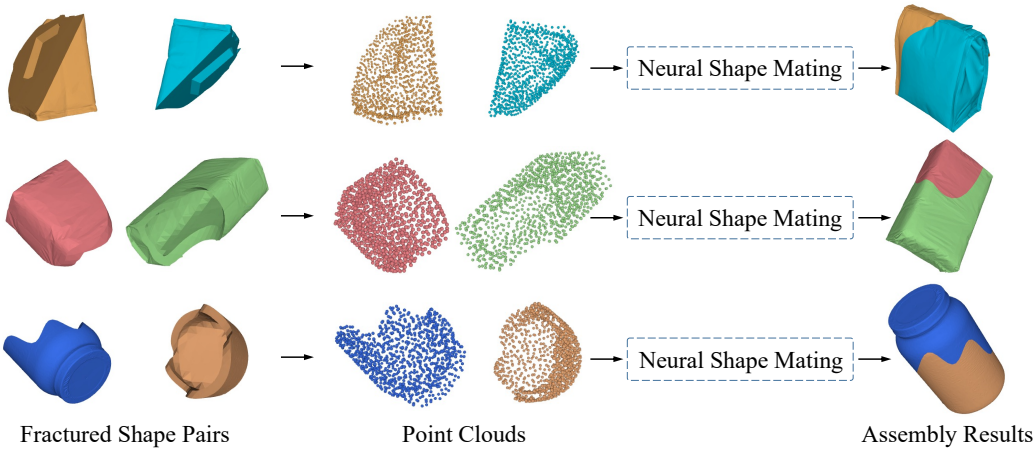

Figure 1: **Pairwise 3D geometric shape assembly.** Neural Shape Mating (NSM) takes a pair of point clouds of the fractured shapes as input and predicts the assembled configuration as output. NSM learns to assemble shapes with self-supervision and does not require semantic information or target shape as guidance during test time.

We formulate this task as a pose prediction problem and propose Neural Shape Mating (NSM), which takes as input point cloud observations of the two shapes, reasons about their structures and the fit, and predicts respective poses that bring them together using a transformer based architecture to attend to assymetric correlations between local geometric cues. To account for imperfect point cloud observations (e.g., noisy point clouds), we couple the training of NSM with an implicit shape reconstruction task (Park et al., 2019b; Sitzmann et al., 2020). Furthermore, we also use an adversarial shape prior evaluate the plausability of the generated shape. This joint learning scheme allows the model to learn more robust features for reliable mating poses.

Furthermore, to train NSM, we present a self-supervised data collection pipeline that generates pairwise shape assembly data with ground truth by randomly cutting an object mesh with different cut types into two parts. We collect object meshes from the Thingi10K (Zhou & Jacobson, 2016) and Google Scanned Objects (GoogleResearch, 2021) datasets and apply our data generation algorithm to each object mesh. The resulting geometric shape assembly dataset covers a diverse set of cut types applied to numerous object instances of three categories, combining a total of 19,226 shape pairs suitable for evaluating the proposed task. We train NSM on the collected dataset in a self-supervised fashion and compare our method with several point cloud registration algorithms and one part assembly baseline approach. Extensive experimental results and ablation studies under various settings demonstrate the effectiveness of the proposed algorithm.

Summary of contributions:

1. We introduce a novel task of pairwise 3D-assembly of unknown objects and propose a self-supervised learning algorithm Neural Shape Mating which predicts mating configurations using primarily geometric cues.
2. We collect a large-scale geometric shape assembly dataset for evaluating the proposed task. The dataset and the data generation tools will be released to facilitate future research.
3. We benchmark the comparisons with several point cloud registration methods and one part assembly baseline approach and provide a testbed that ensures reproducibility of the results.
4. Experimental results and analysis support our design choice and demonstrate the effectiveness and robustness of our approach when presented with realistically noisy observations.

2 Related Work

3D shape assembly. A distinct, but related, line of work investigates generative models that learn to represent objects as concatenations of simple 3D shapes. Tulsiani et al. (2017) train per-class models that generate objects by assembling volumetric primitives (cuboids). Khan et al. (2019) train a single model that can generate cuboid primitives across all classes. Jones et al. (2020) model objects with ShapeAssembly programs, learned by a VAE, which can be executed to generate 3D shapes as concatenations of cuboids. These methods provide powerful abstractions – diverse objects can be represented as different arrangements of a small set of shape primitives – and reveal correspondences between objects by abstracting away details of local geometry. In contrast, we consider the problem

of discovering plausible fits between shapes with complex geometry that do not correspond to any semantic part or natural object decomposition. The validity of a fit relies on the alignment of detailed local geometric features, which provide cues for assembly. The task that comes closest to our own is part assembly - make a complete object from a set of parts. Li et al. (2020) learn to predict translations and rotations for point cloud parts to assemble a target object specified by an image. Huang et al. (2020) frame the part assembly problem as graph learning, inducing a relational prior, and learn to assemble parts into complete objects by iterative message passing. Both methods use the PartNet dataset (Mo et al., 2019), and thus the parts to assemble are always a reasonable semantic decomposition of the target object. Shape is important in part assembly, but cues can also be taken from part semantics directly, shortcutting the geometric cues. In contrast, we consider the problem of learning to fit together pieces with no particular semantics and without a provided target.

Pose estimation. Existing pose estimation methods predict poses for known objects by aligning a provided model with an observation (Besl & McKay, 1992; Zeng et al., 2017). Other learning based approaches predict poses for novel objects as bounding box corners (Law et al., 2019) or semantic keypoints (You et al., 2020; Wang et al., 2020) or mappings to a normalized coordinate space (Wang et al., 2019a). Rather than estimating the current pose of an observed object, our problem requires predicting a new pose that assembles the observed shapes into a whole object.

Learning shape priors. Our model includes an adversarial prior implemented by a discriminator that learns to distinguish between ground-truth assembled shape pairs and the predicted assembly results. Conditional generative adversarial networks (Goodfellow et al., 2014; Mirza & Osindero, 2014) have achieved impressive results on image generation tasks even when paired ground truth is unavailable, as in unpaired image translation (Zhu et al., 2017), or when ground truth is available but multiple plausible outputs exist, as in MUNIT (Huang et al., 2018). Even when the ground truth is available and a unimodal correct output exists, adversarial losses lead to enhanced detail and more realistic outputs, e.g., for super-resolution (Lucas et al., 2019). In our problem, we learn shape priors with an adversarial loss that encourages our model to generate plausible assembly configurations.

Implicit shape reconstruction. A core problem in computer vision is reconstructing 3D shapes from 2D observations. Rather than representing the reconstructed shapes as finite sets of points, voxels, or triangles, a recent line of work aims to represent them as implicit functions parameterized by neural networks. These encode shapes by their signed distance function (SDF) (Park et al., 2019b; Sitzmann et al., 2020) or the indicator function (Mescheder et al., 2019), which are continuous, differentiable and can be queried at arbitrary resolution. DeepSDF (Park et al., 2019b) learns a generative model of SDFs for many shape classes with a feedforward neural network. Further work (Genova et al., 2019; 2020) adds additional structure to further improve reconstruction accuracy and memory efficiency. Importantly, for our purposes, the learned SDFs are able to capture fine-grained geometric details that are necessary for accurately predicting shape fits. We follow a similar approach to Karunratanakul et al. (2020) and Jiang et al. (2021) which take inspiration from implicit reconstruction to improve performance on a pose prediction task (in their case grasping). Specifically, as in Jiang et al. (2021), we include implicit reconstruction as an auxiliary task and show, through ablation, that this improves performance on our main assembly task, suggesting significant synergies between assembly and shape representations.

Point cloud registration. If we had access to additional information, our problem would reduce to point cloud registration. Specifically, if we had a segmentation of the interface of each piece (the subset of its surface that contacts the other piece in the assembled pose), computing the assembled pose would reduce to aligning the paired interfaces. If, in addition, we were given correspondences between these interfaces, alignment would become a well-characterized optimization problem solvable with the Procrustes method. Without correspondences, we are left with a registration problem. Feature-free methods such as Iterative Closest Point (ICP) (Besl & McKay, 1992) approximate correspondences simply as pairs of closest points. SparseICP (Bouaziz et al., 2013) improves robustness to noise by distinguishing between inliers and outliers. Learning-based methods seek to approximate correspondences in a learned feature space (Deng et al., 2018; Gojcic et al., 2019; Wang & Solomon, 2019). Different from the objectives of these registration approaches, our method is designed to predict paired poses that bring two shapes together to form a whole object.

3 Problem Statement: Pairwise 3D Geometric Shape Assembly

We formulate the pairwise 3D geometric shape assembly task as a pose prediction problem. In this task, we assume we are given the point cloud observations P_A and P_B of the two shapes S_A and S_B, where $P_A = \{p_i^A\}_{i=1}^N$, $p_i^A \in \mathbb{R}^3$, $P_B = \{p_j^B\}_{j=1}^N$, $p_j^B \in \mathbb{R}^3$, and N denotes the number of

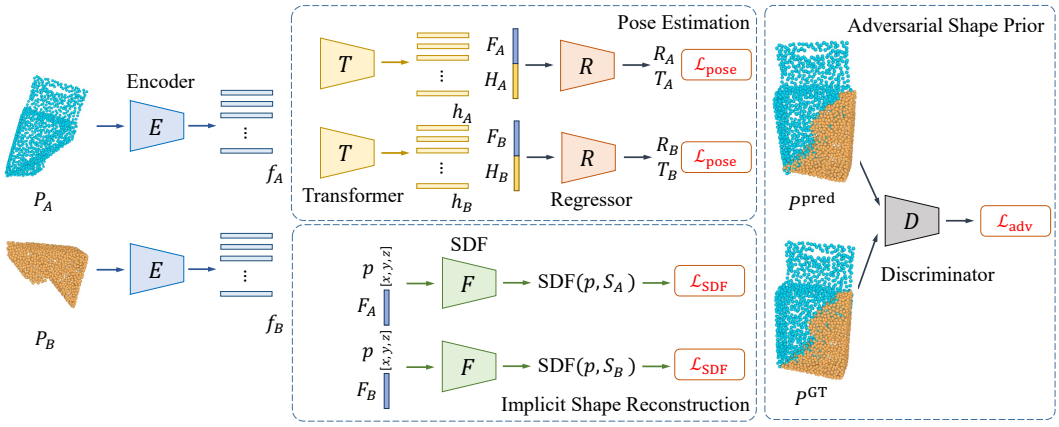

Figure 2: **Overview of Neural Shape Mating.** Neural Shape Mating is composed of four main components: a point cloud encoder E, a pose estimation module that consists of a Transformer network T and a regressor network R, an implicit shape reconstruction module that learns signed distance functions (SDFs), and a discriminator D for learning shape priors.

points in a point cloud. Shape S_A and shape S_B are the two parts of a whole object S. We aim to learn a model that takes as input the two point clouds P_A and P_B and predicts an SE(3) pose $\{(R_k, T_k) \mid R_k \in \mathbb{R}^{3\times3}, T_k \in \mathbb{R}^3\}$ for each point cloud P_k, where $k \in \{A, B\}$. The predicted SE(3) poses will then be applied to transform each respective input point cloud. The union of the two transformed point clouds $\bigcup_{i\in\{A,B\}} R_i P_i + T_i$ forms the shape assembly result.

4 Method: Neural Shape Mating

We describe Neural Shape Mating and the loss functions used to train our model in this section.

4.1 Algorithmic Overview

Given two point clouds P_A and P_B, our goal is to learn a model that predicts SE(3) poses for the input point clouds. We propose Neural Shape Mating, which comprises four components: 1) a point cloud encoder, 2) a pose estimation network, 3) an implicit shape reconstruction network, and 4) an adversarial shape prior module.

As shown in Figure 2, we first apply the point cloud encoder E to each input point cloud P_i to extract the point feature f_i for $i \in \{A, B\}$. Next, the point features are passed to the pose estimation network for reasoning about cross-shape information and predicting SE(3) poses $\{R_i, T_i\}$ for $i \in \{A, B\}$, and to the SDF network F for learning implicit shape reconstruction. The predicted SE(3) poses are then applied to the respective input point cloud. The union of the two transformed point clouds forms the assembly result. To learn plausible shape assembly results, we have a discriminator that takes as input the predicted assembly result and the ground truth and distinguishes whether the input assembly result looks visually realistic or not.

Point cloud encoder. There are several point cloud encoders such as PointNet (Qi et al., 2017a), PointNet++ (Qi et al., 2017b), and DGCNN (Wang et al., 2019b) that are applicable for learning point feature embeddings. In this work, we adopt DGCNN as our point cloud encoder E, since DGCNN jointly considers local and global information during feature extraction.

Rotation representation. We follow prior work (Li et al., 2020) and use quaternion to represent rotations.

The details of the pose estimation network, the adversarial shape prior module, and the implicit shape reconstruction network are described in the following subsections.

4.2 Pose Estimation for Shape Assembly

To predict an SE(3) pose for each point cloud, we have a feature correlation module T that captures cross-shape information and a pose regressor R for pose prediction.

Given the encoded point features f_A and f_B as input, the feature correlation module T takes as input the two point features f_A and f_B and computes features $h_A = \{h_i^A\}_{i=1}^N$ and $h_B = \{h_j^B\}_{j=1}^N$ as output, where $h_i^A \in \mathbb{R}^d$ and $h_j^B \in \mathbb{R}^d$. To realize the feature correlation module, we select the

Transformer network (Vaswani et al., 2017), as it allows the model to learn asymmetric cross-shape information.

To predict SE(3) poses, we aggregate the feature h_i to form the feature $H_i \in \mathbb{R}^d$ for $i \in \{A, B\}$ and aggregate the feature f_i to form the feature $F_i \in \mathbb{R}^d$ for $i \in \{A, B\}$. The features F_i and H_i are then concatenated and passed to the pose regressor R to predict as SE(3) pose $\{R_i, T_i\}$ for $i \in \{A, B\}$. In this work, we use quaternion to represent rotation.

To guide the learning of the pose estimation network, we have a pose loss $\mathcal{L}_{\text{pose}}$, which is defined as

$$\mathcal{L}_{\text{pose}} = \sum_{i \in \{A,B\}} \|R_i^\top R_i^{\text{GT}} - I\| + \|T_i - T_i^{\text{GT}}\|, \tag{1}$$

where R_i^{GT} and T_i^{GT} denote the ground-truth rotation and translation, respectively.

4.3 Adversarial learning of shape priors

Since the task we consider is multimodal in nature (i.e., two shapes can be assembled in many ways), we propose to learn the global shape prior to further constrain the prediction space. We exploit the idea that when the two point clouds are assembled using the predicted poses, the resulting assembly result should look visually realistic like an object. Specifically, we cast this as an adversarial learning problem and introduce a shape assembly discriminator D that takes as input the assembled point cloud $P^{\text{pred}} = \bigcup_{i \in \{A,B\}} R_i P_i + T_i$ and the ground-truth assembled one $P^{\text{GT}} = \bigcup_{i \in \{A,B\}} R_i^{\text{GT}} P_i + T_i^{\text{GT}}$ and distinguishes whether the input assembly results are visually realistic or not.

To achieve this, we have an adversarial loss $\mathcal{L}_{\text{adv}}$, which is defined as

$$\mathcal{L}_{\text{adv}} = \mathbb{E}[\|D(P^{\text{pred}})\|] + \mathbb{E}[\|D(P^{\text{GT}}) - 1\|]. \tag{2}$$

Having the adversarial loss allows our model to predict SE(3) poses that result in visually realistic assembly results.

4.4 Implicit shape reconstruction

To account for the noise in point cloud sampling (i.e., same objects can be described by different point clouds, and the point cloud capturing can be imperfect), we couple the learning of our model with a shape reconstruction network. This is motivated by recent advances in implicit shape modeling (Park et al., 2019a; Mescheder et al., 2019), where learning signed distance functions (SDFs) allows the model to learn more robust shape representations. Specifically, we have an SDF network F that takes as input the point features f_A and f_B, respectively, and a point $p \in \mathbb{R}^3$ in the 3D space, and predicts the signed distance between point p and each respective shape S_A and S_B.

To train the SDF decoder, we have an SDF regression loss, which is defined as

$$\mathcal{L}_{\text{SDF}} = \sum_{i \in \{A,B\}} \|\text{SDF}(p, S_i) - \text{SDF}^{\text{GT}}(p, S_i)\|, \tag{3}$$

where $\text{SDF}(p, S_i)$ and $\text{SDF}^{\text{GT}}(p, S_i)$ denote the predicted and ground-truth SDF values between the point p and the shape S_i, respectively.

5 The Geometric Shape Assembly Dataset for Self-Supervision

To train our model, we present a self-supervised learning pipeline that generates pairwise 3D geometric shape assembly data with ground truth by randomly cutting an object mesh into two parts.

Mesh cutting. We normalize each object mesh by the longest bounding box length such that the object mesh has a maximum bounding box length of 1 and the aspect ratio remains the same. To cut the object, we use the mesh boolean functions provided by the `libigl` C++ library (Jacobson et al., 2018). We construct a height field that will be used to intersect the object mesh for mesh cutting. The height field can be parameterized by different functions. In our work, we define five different types of functions, including a planar function, a sine function, a parabolic function, a square function, and a pulse function. Each of these functions will result in a type of cut. We generate two types of shapes when performing cutting: the solid shape and the shell shape. Figure 3 presents example data generated by applying different types of cuts. More visual examples of our datasets are presented in Figure 6, Figure 7, and Figure 8.

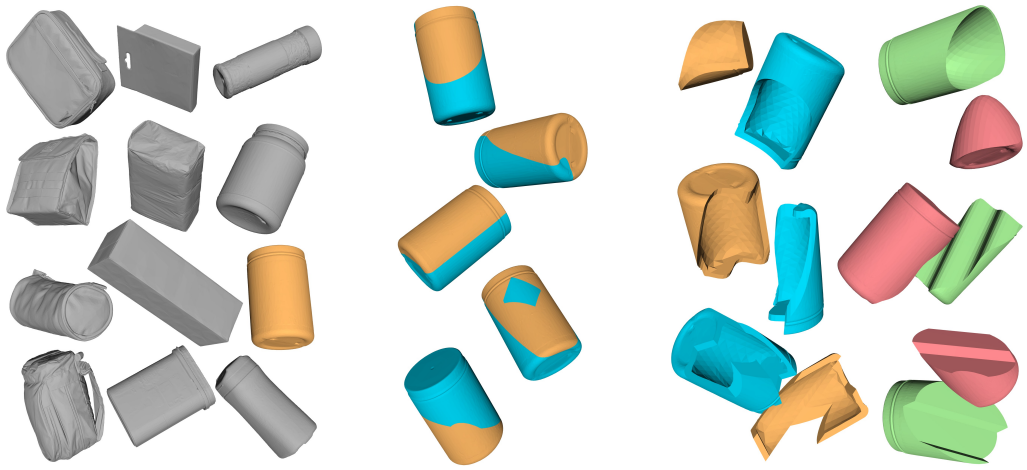

Figure 3: **Dataset overview.** (*Left*) Our dataset is composed of object meshes from three categories. (*Middle*) We define five different types of cut functions. Each object mesh can then be cut with many different ways using varying parametric cut functions. (*Right*) Finally, each pair of parts can be randomized with an initial SE(3) pose. In our dataset, we also generate solid and hollow/shell variations of each shape, when cutting a mesh to create different mating interfaces for the same problem instance.

Table 1: **Dataset statistics.** We summarize the number of shape pairs of the Geometric Shape Assembly dataset. Our dataset contains a large number shape pairs, covering a diverse combination of different shape types, object categories, and cut types.

Category	Number of objects	Solid shape pairs						Shell shape pairs					
		Plane	Parabola	Sine	Square	Pulse	All	Plane	Parabola	Sine	Square	Pulse	All
Bag	28	50	50	50	42	40	232	190	190	190	190	190	950
Box	191	1,390	1,390	1,410	1,380	1,380	6,950	1,360	1,360	1,380	1,350	1,344	6,794
Jar	106	440	440	460	430	430	2,200	420	420	440	410	410	2,100
All	325	1,880	1,880	1,920	1,852	1,850	9,382	1,970	1,970	2,010	1,950	1,944	9,844

SDF ground truths. We uniformly sample 1,024 points on each of the resulting object part meshes. We use the Fast Winding Numbers method (Barill et al., 2018) for computing ground-truth SDF values. For each object part mesh, we sample 40,000 points that are close to the object surface for learning the SDF network.

Pose transformation. Each point cloud is mean centered (i.e., the centroid of the point cloud is the origin). During training, we randomly sample two rotations on the fly and apply them to transform the poses of the two input point clouds, respectively.

Statistics. We use 3 shape categories: `bag`, `box`, and `jar` in initial dataset version used in this paper due to computational reasons, however noting that the data generation procedure extends naively to other shape categories, and NSM is category-agnostic both in training and testing. We collect object meshes from the Thingi10K (Zhou & Jacobson, 2016) and Google Scanned Objects (GoogleResearch, 2021) datasets. The dataset statistics are summarized in Table 1.

6 Experimental Results

We perform evaluations and analysis of NSM to answer the following questions:

1. How well does NSM perform when compared to point cloud registration methods and the graph-network based assembly baseline approaches?
2. Can NSM generalize to unseen object categories and unseen cut types, without fine-tuning?
3. Does the performance deteriorate on more realistic, noisy point clouds?
4. How much do the adversarial, reconstruction and pose loss contribute to final performance?

Table 2: **Experimental results of geometric shape assembly.** R and T denote Rotation and Translation, respectively. Lower is better on all metrics. It is worth noting that many methods can get reasonably close in position, but be completely off in orientation as demonstrated by the RMSE error in rotation. NSM outperforms the best baseline in predicting the correct orientation by upto $5\times$.

Method	MSE (R)	RMSE (R)	MAE (R)	MSE (T)	RMSE (T)	MAE (T)
Solid Objects					($\times 10^{-3}$)	
ICP (point-to-point) (Besl & McKay, 1992)	9565.40	97.80	84.90	238.97	488.85	379.97
ICP (point-to-plane) (Besl & McKay, 1992)	9095.82	95.37	79.06	84.32	290.38	130.98
Sparse ICP (point-to-point) (Bouaziz et al., 2013)	6234.84	78.96	71.04	197.42	444.32	247.06
Sparse ICP (point-to-plane) (Bouaziz et al., 2013)	4164.25	64.53	59.13	186.40	431.74	227.86
DCP (Wang & Solomon, 2019)	8593.54	92.70	73.77	540.23	735.01	574.70
GNN Assembly (Li et al., 2020)	947.84	30.79	25.20	10.24	101.19	76.70
Neural Shape Mating (NSM)	49.45	7.03	5.63	9.17	98.42	73.20
Shell/Hollow Objects						
ICP (point-to-point) (Besl & McKay, 1992)	11 186.87	105.77	102.90	823.32	907.37	840.77
ICP (point-to-plane) (Besl & McKay, 1992)	9424.92	97.08	89.03	722.61	850.07	813.37
Sparse ICP (point-to-point) (Bouaziz et al., 2013)	7533.89	86.80	83.14	516.67	718.80	660.59
Sparse ICP (point-to-plane) (Bouaziz et al., 2013)	7108.51	84.31	80.44	607.79	779.69	699.81
DCP (Wang & Solomon, 2019)	9400.10	96.95	87.07	635.37	778.06	605.33
GNN Assembly (Li et al., 2020)	787.53	28.06	23.02	14.81	121.69	94.91
Neural Shape Mating (NSM)	96.39	9.82	8.77	15.62	124.98	97.63

Figure 4: **Visual results on pairwise 3D geometric shape assembly.**

6.1 Experimental Setup

Evaluation metrics. We follow the evaluation scheme from DCP (Wang & Solomon, 2019). We compute the mean squared error (MSE), root mean squared error (RMSE), and mean absolute error (MAE) between the predicted rotation and translation values and the ground truth values.

Baselines. We compare our model with several point cloud registration methods: ICP (Besl & McKay, 1992), Sparse ICP (Bouaziz et al., 2013) and DCP (Wang & Solomon, 2019) as well as GNN Assembly, a graph-based part assembly approach adapted from Li et al. (2020). The three registration methods are all correspondence-based, that is, they approximate correspondences between point clouds then find poses that minimize an energy based on those correspondences. ICP estimates correspondences as closest points and proceeds to iterate between updating poses (from the latest correspondences) and updating correspondences (from the latest poses). Since ICP weighs all correspondences equally, it can be thrown off by a few bad points. Sparse ICP improves robustness to noise by downweighting outliers. We include two variants of the ICP methods, one computing distances point-to-point and the other point-to-plane. DCP is a deep learning-based method which learns to compute correspondences from which a final pose is generated with SVD. GNN Assembly is another deep-learning-based method, but predicts rotations and translations with a message passing algorithm without correspondences (more details on both in Section 2). In each experiment, DCP, GNN Assembly and Neural Shape Mating (Ours) are trained on the same ground truth pose data.

Implementation details. We implement our model using PyTorch (Paszke et al., 2019). We use the ADAM (Kingma & Ba, 2014) optimizer for training. The learning rate is set to 1×10^{-3}. The batch size is set to 4. We train our model on two NVIDIA P100 GPUs with 12GB memory each for 100 epochs. We use the Open3D implementation for the ICP method. The implementations of Sparse ICP and DCP are from their official GitHub repository. We use the codebase from Li et al. (2020) for GNN Assembly and remove the part segmentation network branch. We evaluate all methods on both the solid shape and shell shape assembly settings.

6.2 Performance evaluation and comparisons on 3D geometric shape assembly

E1. Comparison to existing approaches

We compare the performance of our NSM method with existing approaches on pairwise 3D geometric shape assembly. In this evaluation, we use 80% of the shape pairs for training, 10% for validation

Table 3: **Generalization: Unseen categories geometric shape assembly.** The training set contains shape pairs from the bag and box categories. The test set contains shape pairs from the jar category. Results reported are on solid objects. Results on shell/hollow objects are available in the appendix (Table 6).

Method	MSE (R)	RMSE (R)	MAE (R)	MSE (T)	RMSE (T)	MAE (T)
Solid Objects					($\times 10^{-3}$)	
ICP (point-to-point) (Besl & McKay, 1992)	16 378.88	127.98	113.78	598.88	773.87	724.10
ICP (point-to-plane) (Besl & McKay, 1992)	13 140.04	114.63	109.71	702.56	838.19	774.47
Sparse ICP (point-to-point) (Bouaziz et al., 2013)	9145.48	95.63	90.45	697.70	853.29	741.78
Sparse ICP (point-to-plane) (Bouaziz et al., 2013)	5326.96	72.99	68.14	578.83	760.80	714.43
DCP (Wang & Solomon, 2019)	19 692.49	140.33	106.10	612.43	782.58	615.04
GNN Assembly (Li et al., 2020)	1060.15	32.56	29.98	8.59	92.67	72.18
Neural Shape Mating (NSM)	140.82	11.87	10.33	9.41	97.02	75.70

Table 4: **Generalization: Unseen cut types geometric shape assembly.** The training set contains the planar, sine, square and pulse cut types. The test set contains the parabolic cut type. Results reported are on solid objects. Results on shell/hollow objects are available in the appendix (Table 7).

Method	MSE (R)	RMSE (R)	MAE (R)	MSE (T)	RMSE (T)	MAE (T)
Solid Objects					($\times 10^{-3}$)	
ICP (point-to-point) (Besl & McKay, 1992)	7436.99	86.24	80.09	441.31	664.31	570.90
ICP (point-to-plane) (Besl & McKay, 1992)	6171.52	78.56	72.30	401.14	633.43	583.41
Sparse ICP (point-to-point) (Bouaziz et al., 2013)	7053.14	83.98	77.04	704.81	839.53	679.91
Sparse ICP (point-to-plane) (Bouaziz et al., 2013)	3231.01	56.84	49.08	231.71	481.36	418.59
DCP (Wang & Solomon, 2019)	8538.16	92.40	73.51	544.28	737.75	576.42
GNN Assembly (Li et al., 2020)	1059.18	32.55	27.08	13.63	116.75	79.55
Neural Shape Mating (NSM)	135.82	11.65	10.21	13.41	115.82	79.63

and 10% for testing (metrics are reported on this holdout set). Table 2 present results on both solid and shell shapes. Figure 4 presents a visual comparison between methods.

Quantiatively, results in both settings follow a similar pattern. NSM achieves the best rotation MSE by an order of magnitude. For translation prediction, NSM and GNN Assembly both achieve strong results, several orders of magnitude better the other methods.

Point-cloud registration methods. NSM outperforms registration methods by a large margin on all metrics. This may be surprising as shape assembly and point cloud registration are similar problems. In fact, shape assembly reduces to point cloud alignment given an interface segmentation. Despite this, these results suggest that existing point cloud registration methods are insufficient for the assembly task. In our qualitative results, we can see registration methods often attempt to overlay pieces rather than assemble them and this matches our hypothesis that the poor performance of registration methods is due to their correspondence assumptions. In point cloud registration, it is assumed that the inputs correspond usually to a rigid transformation and some observation noise. Even with outlier handling, they are unable to leave the non-interface portion of the surface out of correspondence in order to precisely align the interface portions. More surprisingly, this may be true even for learning-based methods like DCP, where the interpolation of correspondences may force consideration of non-interface points. This is not a flaw in these methods, but highlights that shape assembly is a distinct problem from registration, requiring more specialized method design.

Part Assembly. NSM outperforms GNN Assembly (Li et al., 2020) on rotation prediction and performs similarly on translation prediction. The GNN Assembly architecture is designed for the part assembly task where semantic cues are available and fine-grained surface details are not as important for alignment. We hypothesize that our adversarial loss and the transformer architecture are better-suited to the shape assembly task which relies on these details. These results support our conviction that the shape assembly problem is distinct from point cloud registration and part assembly, and that progress will require further investigation into the assembly task specifically.

E2. Generalization to unseen categories and cut types

Unseen categories. To test the generalization across categories, we train on the `box` and `bag` categories and evaluate on the `jar` category. (See Table 1 for category details.) Table 3 presents results on solid shapes and results on shell shapes are available in the appendix (Table 6). Notably, NSM is category agnostic, and relies mainly on aligning surface geometry details than class-specific semantic cues, we expected strong generalization. Compared to in-category testing, while the performance degrades slightly for both solid shape and shell shapes, the overall trends remain similar.

Unseen cut types. To test the generalization across different cut types, we test on parabolic cuts and train on the remaining 4 cut types. Table 4 presents results on solid shapes and the results on

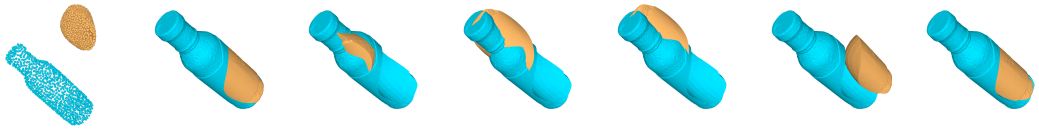

Figure 5: **Visual results on noisy point cloud pairwise 3D geometric shape assembly.**

Table 5: **Ablation study on Neural Shape Mating model design choices.** Testing performance with each loss removed. The training and test sets remain the same as in the main experiment (as presented in Table 2).

Method	MSE (R)	RMSE (R)	MAE (R)	MSE (T)	RMSE (T)	MAE (T)
Solid Objects					($\times 10^{-3}$)	
NSM	49.45	7.03	5.63	9.17	98.42	73.20
NSM w/o $\mathcal{L}_{\text{adv}}$	202.52	14.23	12.08	10.00	100.02	99.88
NSM w/o $\mathcal{L}_{\text{SDF}}$	137.03	11.71	10.78	11.43	106.92	100.04
NSM w/o $\mathcal{L}_{\text{pose}}$	10 678.33	103.34	97.46	894.33	945.69	852.27
Shell/Hollow Objects						
NSM	96.39	9.82	8.78	15.62	124.98	97.63
NSM w/o $\mathcal{L}_{\text{adv}}$	154.93	12.45	11.03	20.78	144.16	129.08
NSM w/o $\mathcal{L}_{\text{SDF}}$	176.60	13.29	11.44	18.33	135.4	109.77
NSM w/o $\mathcal{L}_{\text{pose}}$	6349.38	79.68	71.44	749.55	865.77	810.03

shell shapes are available in the appendix (Table 7). See Table 1 for cut type details. As with unseen categories, the performance degrades somewhat for solid shape assembly and only slightly for shell shape assembly. Otherwise the pattern of results remains similar.

E3. Evaluation on noisy point clouds

Real-world point-cloud data, e.g., captured by depth cameras, contains measurement error that the point clouds in our training set do not. For our framework to be applicable to real world problems, it must be robust to noise in the point cloud observations. To test robustness to noise, we train and test the model on a noise-augmented version of our dataset. Gaussian noise with mean 0.0 and standard deviation 0.05 is added to each point. While the performance of all methods, including ours, does decline, NSM is still able to predict reasonable assembled poses as can be seen in Figure 5. Full metrics on solid and shell shapes are available in the appendix (Table 8). The performance of correspondence-based methods (ICP, Sparse ICP, and even learning-based DCP) all show large drops in performance even when ground truth normals are provided.

E4. Ablation Study: Contribution of loss functions

To evaluate our design choices, we conduct an ablation study by removing a loss function at a time. Table 5 presents the results of the solid shape and the shell shape assembly settings, respectively. The training and test sets remain the same as in the main experiment (as presented in Table 2). Performance declines significantly without the adversarial loss, confirming our hypothesis that the adversarial loss prior can serve as a pose refinement or regularizer and improve predictions even when we have ground truth available. Performance also declines without the implicit reconstruction loss, suggesting that there are useful synergies between shape assembly and geometry reconstruction. Without the pose loss, the model does not learn to assemble at all, which suggests adversarial training with implicit shape reconstruction alone is not sufficient.

7 Conclusions

This paper introduces Neural Shape Mating, a framework for learning pairwise reasoning about geometric shapes and the possible fits between them. Quantitative results show NSM is better suited to the geometric shape assembly task than point cloud registration or part assembly methods, and qualitative analysis suggests that this is because of the correspondence assumptions relied on by registration methods and the differing representational requirements of semantic part assembly (which, unlike geometric shape assembly, does not rely on local geometric details). Since NSM learns to align geometric features, rather than semantic ones, it is able to generalize across categories and across surface cut types. An ablation study suggests that the adversarial loss significantly improves performance (even though ground truth is available) and the performance benefits of an auxiliary implicit representation task suggest synergies between shape reconstruction and assembly. We hope that this paper can convincingly establish geometric shape assembly as a meaningful task, distinct from semantic part assembly, and that releasing our dataset and the details of our baseline will allow the community to form around this important problem. Natural extensions of NSM would go beyond pairwise assembly to consider the problem of assembling n-parts.

REPRODUCIBILITY

The source code and the dataset will be made publicly available upon paper acceptance. We document all details of the implementation and evaluation setting in section 6.1 and describe all necessary components of our loss in the main text. We use the open source implementations of baselines for ICP, Sparse ICP, and DCP for comparative evaluation.

REFERENCES

Maneesh Agrawala, Doantam Phan, Julie Heiser, John Haymaker, Jeff Klingner, Pat Hanrahan, and Barbara Tversky. Designing effective step-by-step assembly instructions. *ACM TOG*, 2003.

Gavin Barill, Neil G Dickson, Ryan Schmidt, David IW Levin, and Alec Jacobson. Fast winding numbers for soups and clouds. *ACM TOG*, 2018.

Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. *TPAMI*, 1992.

Sofien Bouaziz, Andrea Tagliasacchi, and Mark Pauly. Sparse iterative closest point. In *Computer graphics forum*, 2013.

Xuelin Chen, Hao Zhang, Jinjie Lin, Ruizhen Hu, Lin Lu, Qi-Xing Huang, Bedrich Benes, Daniel Cohen-Or, and Baoquan Chen. Dapper: decompose-and-pack for 3d printing. *ACM TOG*, 2015.

Haowen Deng, Tolga Birdal, and Slobodan Ilic. Ppfnet: Global context aware local features for robust 3d point matching. In *CVPR*, 2018.

Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *ICCV*, 2019.

Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3d shape. In *CVPR*, 2020.

Zan Gojcic, Caifa Zhou, Jan D Wegner, and Andreas Wieser. The perfect match: 3d point cloud matching with smoothed densities. In *CVPR*, 2019.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *NeurIPS*, 2014.

GoogleResearch. Google scanned objects, 2021.

Ankit Goyal and Jia Deng. Packit: A virtual environment for geometric planning. In *ICML*, 2020.

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. *arXiv*, 2020.

Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *ECCV*, 2018.

Alec Jacobson. Generalized matryoshka: Computational design of nesting objects. In *Computer Graphics Forum*, 2017.

Alec Jacobson, Daniele Panozzo, et al. libigl: A simple C++ geometry processing library, 2018.

Zhenyu Jiang, Yifeng Zhu, Maxwell Svetlik, Kuan Fang, and Yuke Zhu. Synergies between affordance and geometry: 6-dof grasp detection via implicit representations. *arXiv*, 2021.

R Kenny Jones, Theresa Barton, Xianghao Xu, Kai Wang, Ellen Jiang, Paul Guerrero, Niloy J Mitra, and Daniel Ritchie. Shapeassembly: Learning to generate programs for 3d shape structure synthesis. *ACM TOG*, 2020.

Korrawe Karunratanakul, Jinlong Yang, Yan Zhang, Michael Black, Krikamol Muandet, and Siyu Tang. Grasping field: Learning implicit representations for human grasps. In *3DV*, 2020.

Salman H Khan, Yulan Guo, Munawar Hayat, and Nick Barnes. Unsupervised primitive discovery for improved 3d generative modeling. In *CVPR*, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014.

Hei Law, Yun Teng, Olga Russakovsky, and Jia Deng. Cornernet-lite: Efficient keypoint based object detection. *arXiv*, 2019.

Youngwoon Lee, Edward S Hu, Zhengyu Yang, Alex Yin, and Joseph J Lim. Ikea furniture assembly environment for long-horizon complex manipulation tasks. *arXiv*, 2019.

Honghua Li, Ibraheem Alhashim, Hao Zhang, Ariel Shamir, and Daniel Cohen-Or. Stackabilization. *ACM TOG*, 2012.

Yichen Li, Kaichun Mo, Lin Shao, Minhyuk Sung, and Leonidas Guibas. Learning 3d part assembly from a single image. In *ECCV*, 2020.

Alice Lucas, Santiago Lopez-Tapia, Rafael Molina, and Aggelos K Katsaggelos. Generative adversarial networks and perceptual losses for video super-resolution. *TIP*, 2019.

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv*, 2014.

Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In *CVPR*, 2019.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019a.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019b.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017a.

Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv*, 2017b.

Vincent Sitzmann, Eric R Chan, Richard Tucker, Noah Snavely, and Gordon Wetzstein. Metasdf: Meta-learning signed distance functions. *arXiv*, 2020.

Shubham Tulsiani, Hao Su, Leonidas J. Guibas, Alexei A. Efros, and Jitendra Malik. Learning shape abstractions by assembling volumetric primitives. In *CVPR*, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

Chen Wang, Roberto Martín-Martín, Danfei Xu, Jun Lv, Cewu Lu, Li Fei-Fei, Silvio Savarese, and Yuke Zhu. 6-pack: Category-level 6d pose tracker with anchor-based keypoints. In *ICRA*, 2020.

Fan Wang and Kris Hauser. Stable bin packing of non-convex 3d objects with a robot manipulator. In *ICRA*, 2019.

He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *CVPR*, 2019a.

Yue Wang and Justin M Solomon. Deep closest point: Learning representations for point cloud registration. In *ICCV*, 2019.

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*, 2019b.

Yang You, Yujing Lou, Chengkun Li, Zhoujun Cheng, Liangwei Li, Lizhuang Ma, Cewu Lu, and Weiming Wang. Keypointnet: A large-scale 3d keypoint dataset aggregated from numerous human annotations. In *CVPR*, 2020.

Kevin Zakka, Andy Zeng, Johnny Lee, and Shuran Song. Form2fit: Learning shape priors for generalizable assembly from disassembly. In *ICRA*, 2020.

Andy Zeng, Kuan-Ting Yu, Shuran Song, Daniel Suo, Ed Walker, Alberto Rodriguez, and Jianxiong Xiao. Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge. In *ICRA*, 2017.

Kang Zhang, Wuyi Yu, Mary Manhein, Warren Waggenspack, and Xin Li. 3d fragment reassembly using integrated template guidance and fracture-region matching. In *ICCV*, 2015.

Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv*, 2016.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.

A APPENDIX: NETWORK ARCHITECTURES OF NSM

Point cloud encoder. We adopt DGCNN (Wang et al., 2019b) as our point cloud encoder.

Transformer network. Our Transformer network consists of an encoder, a decoder, and a 4-head attention module. Both the encoder and the decoder consist of two fully connected layers.

Regressor. Our regressor consists of two fully connected layers.

Discriminator. We adopt DGCNN (Wang et al., 2019b) as our discriminator.

SDF network. Our SDF network consists of four fully connected layers.

B APPENDIX: ADDITIONAL RESULTS

Table 6: **Generalization: Unseen categories geometric shape assembly.** The training set contains shape pairs from the bag and box categories. The test set contains shape pairs from the jar category.

Method	MSE (R)	RMSE (R)	MAE (R)	MSE (T)	RMSE (T)	MAE (T)
Solid Objects					($\times 10^{-3}$)	
ICP (point-to-point) (Besl & McKay, 1992)	16 378.88	127.98	113.78	598.88	773.87	724.10
ICP (point-to-plane) (Besl & McKay, 1992)	13 140.04	114.63	109.71	702.56	838.19	774.47
Sparse ICP (point-to-point) (Bouaziz et al., 2013)	9145.48	95.63	90.45	697.70	853.29	741.78
Sparse ICP (point-to-plane) (Bouaziz et al., 2013)	5326.96	72.99	68.14	578.83	760.80	714.43
DCP (Wang & Solomon, 2019)	19 692.49	140.33	106.10	612.43	782.58	615.04
GNN Assembly (Li et al., 2020)	1060.15	32.56	29.98	8.59	92.67	72.18
Neural Shape Mating (NSM)	140.82	11.87	10.33	9.41	97.02	75.70
Shell/Hollow Objects						
ICP (point-to-point) (Besl & McKay, 1992)	12 915.19	113.64	107.11	710.02	842.63	759.93
ICP (point-to-plane) (Besl & McKay, 1992)	7280.53	85.33	81.04	698.83	835.96	749.98
Sparse ICP (point-to-point) (Bouaziz et al., 2013)	5680.34	75.37	68.07	491.83	646.78	587.09
Sparse ICP (point-to-plane) (Bouaziz et al., 2013)	4656.29	68.24	62.09	410.08	640.37	557.90
DCP (Wang & Solomon, 2019)	8706.93	93.31	74.61	611.24	781.82	616.52
GNN Assembly (Li et al., 2020)	898.80	29.98	22.32	13.14	114.64	88.22
Neural Shape Mating (NSM)	104.07	11.72	10.20	13.49	116.17	87.00

Table 7: **Generalization: Unseen cut types geometric shape assembly.** The training set contains the planar, sine, square and pulse cut types. The test set contains the parabolic cut type.

Method	MSE (R)	RMSE (R)	MAE (R)	MSE (T)	RMSE (T)	MAE (T)
Solid Objects					($\times 10^{-3}$)	
ICP (point-to-point) (Besl & McKay, 1992)	7436.99	86.24	80.09	441.31	664.31	570.90
ICP (point-to-plane) (Besl & McKay, 1992)	6171.52	78.56	72.30	401.14	633.43	583.41
Sparse ICP (point-to-point) (Bouaziz et al., 2013)	7053.14	83.98	77.04	704.81	839.53	679.91
Sparse ICP (point-to-plane) (Bouaziz et al., 2013)	3231.01	56.84	49.08	231.71	481.36	418.59
DCP (Wang & Solomon, 2019)	8538.16	92.40	73.51	544.28	737.75	576.42
GNN Assembly (Li et al., 2020)	1059.18	32.55	27.08	13.63	116.75	79.55
Neural Shape Mating (NSM)	135.82	11.65	10.21	13.41	115.82	79.63
Shell/Hollow Objects						
ICP (point-to-point) (Besl & McKay, 1992)	3288.33	57.34	51.09	643.02	801.89	749.34
ICP (point-to-plane) (Besl & McKay, 1992)	2125.47	46.10	40.33	547.10	739.66	613.38
Sparse ICP (point-to-point) (Bouaziz et al., 2013)	2409.10	49.08	42.94	332.10	576.28	497.70
Sparse ICP (point-to-plane) (Bouaziz et al., 2013)	1688.88	41.10	36.55	279.84	529.00	447.07
DCP (Wang & Solomon, 2019)	8493.38	92.16	73.56	570.54	755.34	586.35
GNN Assembly (Li et al., 2020)	1771.99	42.09	31.02	15.72	125.40	89.71
Neural Shape Mating (NSM)	119.14	10.91	9.43	16.21	127.33	90.72

Table 8: **Experimental results of noisy point clouds geometric shape assembly.**

Method	MSE (R)	RMSE (R)	MAE (R)	MSE (T)	RMSE (T)	MAE (T)
Solid Objects					($\times 10^{-3}$)	
ICP (point-to-point) (Besl & McKay, 1992)	20 349.02	142.65	134.08	894.44	945.75	851.67
ICP (point-to-plane) (Besl & McKay, 1992)	15 400.81	124.10	117.98	780.34	883.37	820.14
Sparse ICP (point-to-point) (Bouaziz et al., 2013)	6780.86	82.35	74.91	653.35	808.30	749.91
Sparse ICP (point-to-plane) (Bouaziz et al., 2013)	4882.80	69.88	61.09	630.05	793.76	723.30
DCP (Wang & Solomon, 2019)	13 592.30	116.59	105.04	733.82	856.63	776.21
GNN Assembly (Li et al., 2020)	1386.37	37.23	33.41	157.83	396.66	318.09
Neural Shape Mating (NSM)	357.25	18.90	15.66	137.89	371.34	329.95
Shell/Hollow Objects						
ICP (point-to-point) (Besl & McKay, 1992)	16 869.59	129.88	120.12	884.07	940.25	857.73
ICP (point-to-plane) (Besl & McKay, 1992)	10 519.58	102.56	92.34	744.91	863.08	783.99
Sparse ICP (point-to-point) (Bouaziz et al., 2013)	8293.38	91.07	84.34	673.38	820.60	743.37
Sparse ICP (point-to-plane) (Bouaziz et al., 2013)	7584.84	87.09	80.10	579.01	760.93	697.71
DCP (Wang & Solomon, 2019)	7068.27	84.07	79.44	490.88	700.63	623.38
GNN Assembly (Li et al., 2020)	1582.77	39.78	31.00	42.17	205.35	183.98
Neural Shape Mating (NSM)	263.52	16.23	14.62	27.04	164.45	147.69

C Appendix: Additional Visual Examples of the Dataset

Figure 6, Figure 7 and Figure 8 present additional visual examples of the geometric shape assembly dataset.

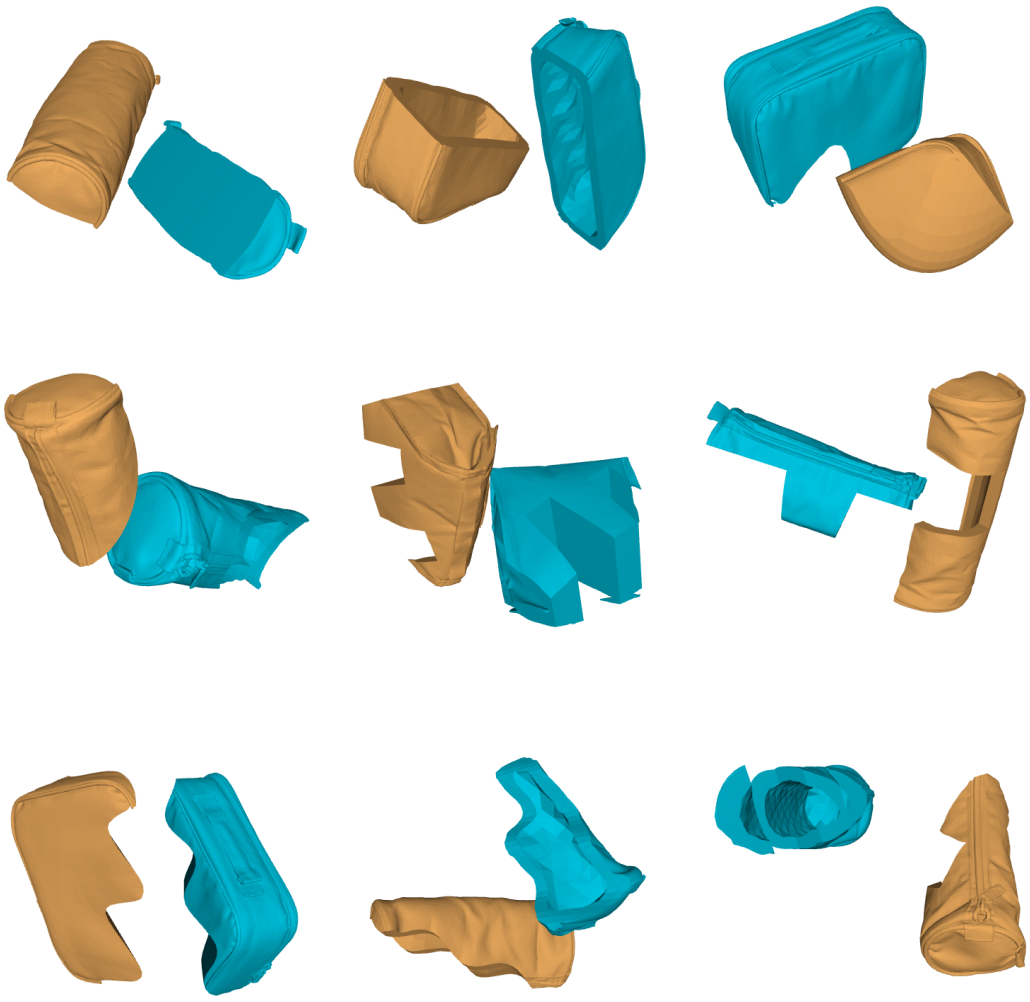

Figure 6: **Visual examples.** We present more visual examples of the shape pairs in the `bag` category.

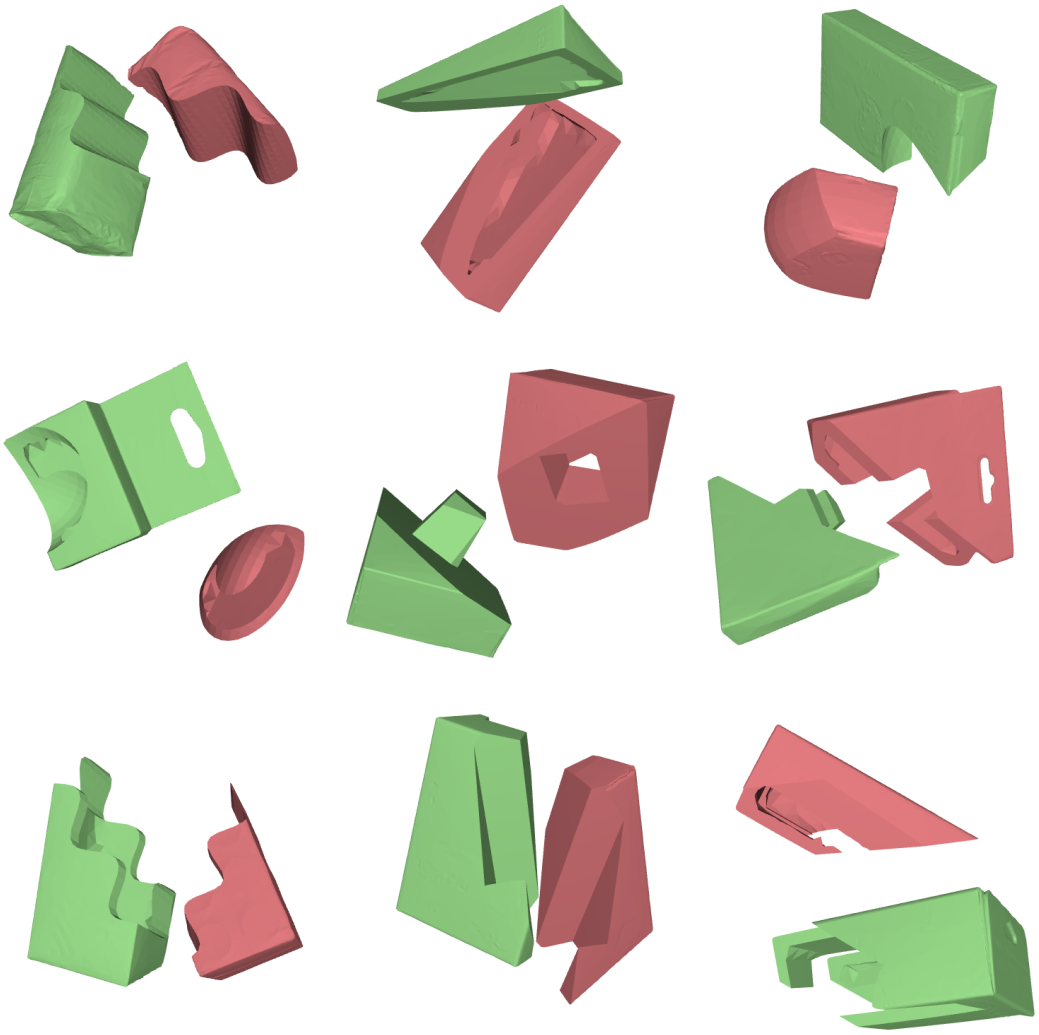

Figure 7: **Visual examples.** We present more visual examples of the shape pairs in the `box` category.

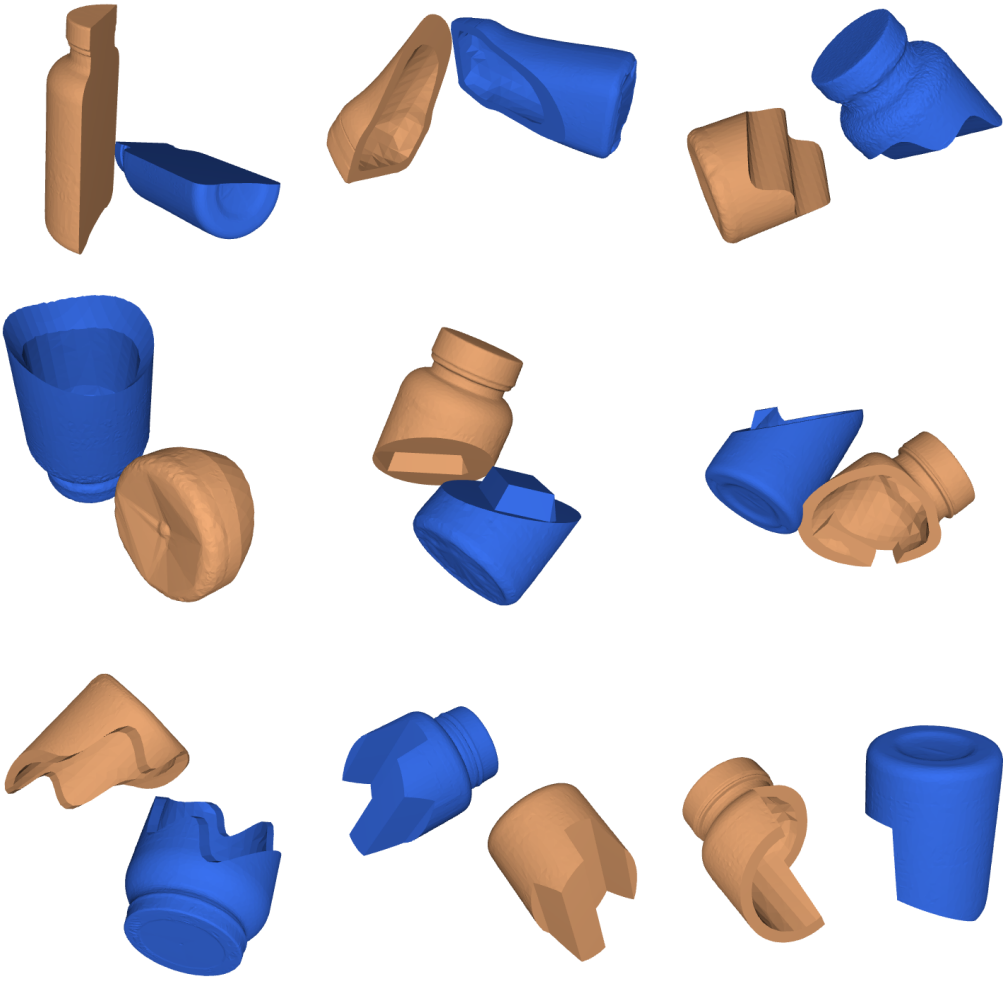

Figure 8: **Visual examples.** We present more visual examples of the shape pairs in the `jar` category.