JoinABLe: Learning Bottom-up Assembly of Parametric CAD Joints

Karl D.D. Willis[1] Pradeep Kumar Jayaraman[1] Hang Chu[1] Yunsheng Tian[2] Yifei Li[2]
Daniele Grandi [1] Aditya Sanghi[1] Linh Tran[1] Joseph G. Lambourne[1]
Armando Solar-Lezama[2] Wojciech Matusik[2]
[1]Autodesk Research [2]MIT CSAIL

Abstract

Physical products are often complex assemblies combining a multitude of 3D parts modeled in computer-aided design (CAD) software. CAD designers build up these assemblies by aligning individual parts to one another using constraints called joints. In this paper we introduce JoinABLe, a learning-based method that assembles parts together to form joints. JoinABLe uses the weak supervision available in standard parametric CAD files without the help of object class labels or human guidance. Our results show that by making network predictions over a graph representation of solid models we can outperform multiple baseline methods with an accuracy (79.53%) that approaches human performance (80%). Finally, to support future research we release the Fusion 360 Gallery assembly dataset, containing assemblies with rich information on joints, contact surfaces, holes, and the underlying assembly graph structure.

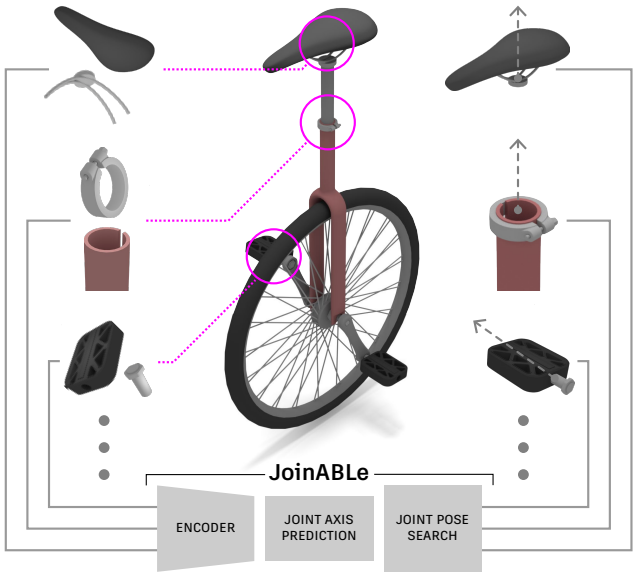

Figure 1. CAD assemblies contain valuable *joint* information describing how parts are locally constrained and positioned together. We use this weak supervision to learn a bottom-up approach to assembly. *JoinABLe* combines an encoder and joint axis prediction network together with a neurally guided joint pose search to assemble pairs of parts without class labels or human guidance.

1. Introduction

The physical products that surround us every day are often complex assemblies combining a multitude of parts modeled using computer-aided design (CAD) software. Well-designed assemblies are critical to ensure that products are cost-efficient, reliable, and easy to physically assemble. CAD designers build up assemblies by aligning individual parts to one another using constraints called *joints*. These joints determine the relative pose and allowed degrees of freedom (DOF) of parts in an assembly [35]. For example, a bolt can be constrained to a hole, then a nut constrained to the bolt, and so on until an entire assembly is designed. Assemblies may contain thousands of parts, represented as solid models in the boundary representation (B-Rep) format [27, 51], and are used for everything from furniture, to vehicles, to electronic devices. Defining individual global positions for each part without using joints quickly becomes cumbersome and prone to error. Joints enable designers to quickly make parametric changes to a design while preserving existing relationships between parts and maintaining design intent.

However, fully defining joints in assemblies is time-consuming and only partially automated in commercial tools [44], often resulting in missing or partly defined joint information. A learning-based approach capable of predicting joints could ease the burden of joint definition and enable other applications such as CAD assembly synthesis [46], robotic assembly [28], optimization of dynamic assemblies [61], part motion prediction [48], assembly-aware similarity search [3] and many more. Although joints for real world assemblies are configured in a bottom-up fashion, recent work largely takes a top-down approach to assembly related tasks [15, 17, 32]. Top-down approaches learn a global arrangement of parts from set object and

part classes in carefully annotated data. An open challenge remains to learn to assemble parts without relying on the strong object and part class priors provided in heavily annotated datasets. In this work we ask the following question, illustrated in Figure 1: Given a pair of parts, can we automatically assemble them without prior knowledge of the global design, class labels, or additional human input? Solving this problem is a fundamental building block for leveraging learning-based methods with assemblies. Our long-term motivation is to enable the next generation of assembly aware tools that can increase the reuse of existing components and streamline robotic assembly and disassembly – important steps in reducing the negative impact of physical products [5, 23, 29, 34, 36].

To begin to address this challenge we introduce *JoinABLe* (Joint Assembly Bottom-up Learning), our *bottom-up* approach to assembly that learns how parts connect locally to form parametric CAD joints. *JoinABLe* uses the weak supervision available in parametric CAD files, containing only partial joint labels, to automatically assemble pairs of parts. We make the following contributions:

- We propose a novel learning-based method to automatically assemble pairs of parts using the weak supervision available in parametric CAD files. We do this without the help of object or part class labels, human annotation, or user guidance for the first time.
- We create and release the *Fusion 360 Gallery* assembly dataset, containing CAD assemblies with rich information on joints, contact surfaces, holes, and the underlying assembly graph structure.
- We provide experimental results on both joint axis and joint pose prediction tasks, a human baseline study, and comparisons with multiple other methods.

Our results show that by making network predictions over a graph representation of solid models, we can outperform multiple baseline methods while using fewer network parameters. We demonstrate that our approach performs well with difficult cases where heuristic algorithms can struggle and achieves an accuracy (79.53%) that approaches human performance (80%) on the joint axis prediction task.

2. Related Work

Assemblies have been a critical part of design and engineering for centuries. Since the digitization of CAD in the 1980s, a number of research areas have been explored.

Shape Combination As early as 2004 the power of designing assemblies by combining and reusing existing parts was demonstrated in *Modeling by Example* [9]. Since then a body of work has focused on finding compatible parts to combine together into assemblies [16, 18, 56, 62]. The ability to parametrically assemble parts into novel designs has numerous applications in the media and entertainment industry, where digital worlds can be populated with novel content. Another line of work has focused on assemblies that can be physically fabricated [8, 26, 42, 43, 45, 49], as described in a recent survey [50]. Our work differs in that we automate the pair-wise assembly of real-world CAD parts using a learning based method without class labels or human guidance.

Structure Aware Deep Generative Models 3D shape synthesis has rapidly advanced with the use of structure aware deep generative models [6, 10, 11, 22, 30, 37, 41, 54, 59] that incorporate some notion of assembly structure to describe how the parts of a shape form a whole. Rather than synthesize the parts themselves, we focus instead on assembling existing parts in the industry-standard B-Rep format.

CAD Informed Robotic Assembly Prior knowledge of CAD assemblies has been leveraged for robotic assembly planning [12, 13] and sequencing [7, 20] to constrain the search process and validate assembly sequences. Although not addressed in the current work, we envision our approach can aid in improving the sampling efficiency of reinforcement learning based robotic assembly [47] by inferring joint information when it is absent or not fully specified.

Learning to Assemble Learning-based assembly methods from the literature largely follow a top-down approach that predicts the absolute pose of a set of parts to form an assembly [17, 32, 46, 60]. Predicting the absolute pose, however, can lead to noisy results where parts fail to completely align. To deal with this issue several recent works have leveraged supervision from local contact points between parts [14, 15]. We believe a bottom-up approach is a critical part of solving the assembly problem. Rather than rely on contact points, our work uses the joint information found in parametric CAD files as weak supervision. This allows the output of our method to be reconstructed as fully editable parametric CAD files.

Critical to prior work is training on synthetic assemblies [38, 55] that belong to set object classes, e.g. chairs, drawers, etc., and are manually segmented, annotated with part class labels, and oriented in a consistent manner. However, semantic segmentation is often incompatible with real-world CAD assemblies that segment parts by manufacturing process [35]. Moreover, while training on set object classes greatly improves within-class performance, generalization to unseen categories is an ongoing area of research [14]. Rather than rely on heavily annotated datasets with strong class priors, our work leverages the weak supervision readily available in standard parametric CAD files, and is trained *without* object classes.

Concurrent to our work, AutoMate [21] leverages similar joint information for use with a learning based recommendation system. Here the user selects an area on each part as guidance, and using those selections, AutoMate recom-

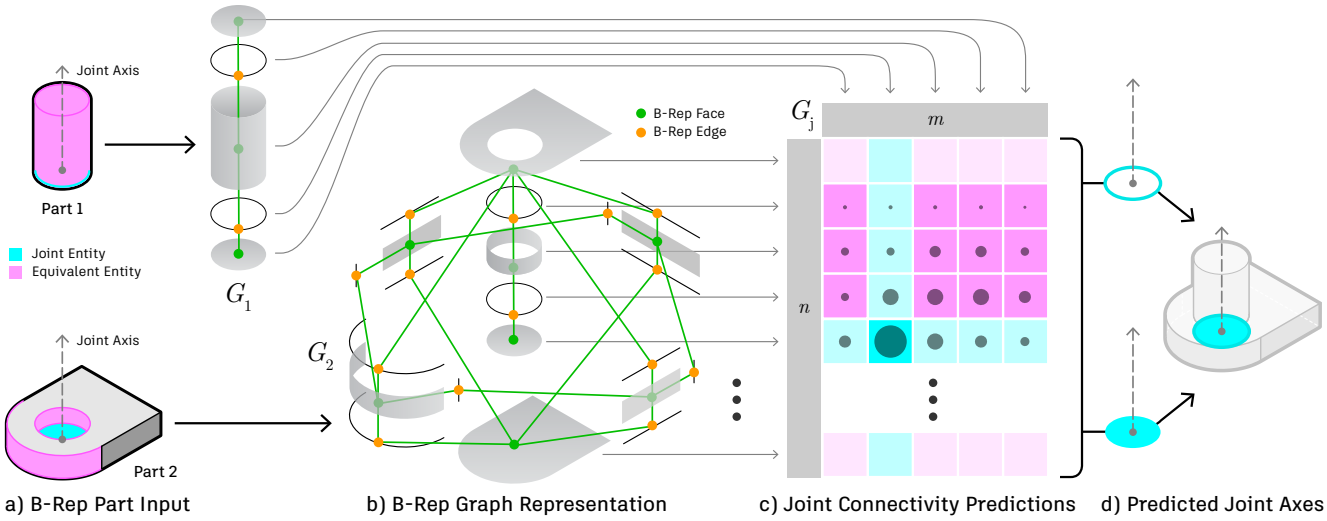

Figure 2. *JoinABLe* is used to assemble a pair of parts in the B-Rep format (a). We use supervision from parametric CAD files containing user selected B-Rep faces and edges that define joints (cyan). We also identify 'equivalent' faces and edges (pink) sharing the same joint axis for use during evaluation. Graphs for each part G_1, G_2 are constructed from adjacent B-Rep faces and edges (b), then joint connectivity predictions are made over a graph G_j containing dense connections between all graph vertices. G_j is shown as an $n \times m$ matrix (c) to visualize the prediction space. Finally, the parts are aligned along the predicted joint axes (d), ready for a subsequent search stage.

mends to the user multiple joint solutions confined to the user-selected input area. Similar to AutoMate, our method enables editable joints to be created in CAD, but we do so in an automated way that does not require user guidance and is not limited to a predefined area. We believe providing an automated solution is critical to enabling advanced assembly applications for CAD and robotics.

Part Mobility Understanding how assembled parts might move, i.e. *part mobility*, is an important problem in both CAD and robotics where the goal is to articulate a given part, such as a hinged door, without knowing the part mobility in advance. Most relevant to our work are systems that automatically predict the relative joint configurations between pairs of parts [31, 48, 58]. Here the input is a point cloud and the output joint axis parameters that define how the parts move in relation to one another. Again, these works rely on strong class priors and heavily annotated synthetic assembly data. We compare our method with adaptions of several part mobility baselines in Section 5.

3. Method

We now present our method, *JoinABLe*, for automatically assembling pairs of parts with joints.

3.1. CAD Joints

Assembly parts are typically represented in the B-Rep format, containing a watertight collection of trimmed parametric surfaces connected together by a well-structured graph [51]. Each face contains a parametric surface, and is bounded by edges that define the trimmed extent of the surface using parametric curves such as lines, arcs, and circles. The B-Rep format is used in all mechanical CAD tools and the selection of B-Rep entities, i.e. faces and edges, is a critical but time-consuming manual task required to set up joints. Our method proposes to learn from these user selections to automate the process of joint creation.

The best practice for CAD assembly is to define relative relationships between pairs of parts, to form joints, also known as *mates*. Joints define the degrees of freedom between the two parts, the parameters for a rest state pose, and the overall motion limits. CAD users select B-Rep entities on each part (highlighted in cyan in Figure 2a) to define a per-part joint axis consisting of an origin point and a direction vector. The joint axes are determined by the type of geometry selection, for a circle the center point becomes the origin point and the normal becomes the direction vector. These two parts can then be aligned along their axes into an assembled state (Figure 2d).

3.2. Joint Prediction Problem Statement

Given a pair of parts (Figure 2a), we aim to create a parametric joint between them, such that the two parts are constrained relative to one another with the same joint axis and pose as defined by the ground truth (Figure 2d). Here the joint axis is defined by two joint origin points and joint direction vectors relative to each part, and the pose is defined by a single rigid transformation in absolute coordinates. We refer to the tasks of predicting these values as

joint axis prediction and *joint pose prediction*, respectively. We consider only pairs of parts that form rigid joints, and leave full multi-part assembly and non-rigid joints to future work. We assume that object or part class labels and any form of human guidance are unavailable. We train only using the weak supervision provided by standard parametric CAD files without any manual human annotation such as canonical alignment.

3.3. Input Representation

Our method takes a pair of parts in the B-Rep format (Figure 2a), building upon a line of recent work [19, 25, 52, 57] that utilizes the topology and geometry available within B-Rep CAD data. This approach enables us to make predictions over the exact entities used to define joints, rather than an intermediate representation such as a mesh or point cloud. Importantly, it allows us to frame the problem as a categorical one, by making predictions over the discrete set of B-Rep entities that contain ground truth information about the joint axis. Joints are commonly defined between *both* B-Rep face and edge entities, e.g. a cylinder (face) can be constrained to another cylinder (face) or a circle (edge). To accommodate this, for each part we build a graph representation, $G(V, E)$, from the B-Rep topology where graph vertices V are either B-Rep faces or edges, and graph edges E are defined by adjacency (Figure 2b).

For graph vertex features we use information about individual B-Rep faces and edges readily available in the B-Rep data structure. For B-Rep faces, we use a one-hot vector for the surface type (plane, cylinder, etc.) and a flag indicating if the surface is reversed with respect to the face. For B-Rep edges, we use a one-hot vector for the curve type (line, circle, etc.), the length of the edge, and a flag indicating if the curve is reversed with respect to the edge.

Finally, given the two graphs G_1, G_2 that we wish to assemble, with n and m vertices respectively, we form a third 'joint connectivity graph' G_j that densely connects the vertices between G_1 and G_2. G_j has $n \times m$ edges and allows us to formulate a link prediction problem [33], by identifying the connections between G_1 and G_2 that form a joint. G_j can be easily visualized as an $n \times m$ matrix (Figure 2c).

3.4. Weak Supervision from CAD Joints

A pair of parts in the B-Rep format have a finite number of faces and edges that can be paired to form a joint, specifically the $n \times m$ edges in G_j. Each ground truth joint results in a single positive label in the $n \times m$ prediction space and all remaining combinations are negative labels. For complex parts, such as mechanical gears that may contain thousands of discrete B-Rep entities, this results in an *extreme* imbalance between positive and negative labels.

The problem is further compounded by having only weak supervision available in standard parametric CAD files. This is due to several reasons: firstly, specifying joints between parts is time consuming and is often skipped by CAD designers; secondly, each CAD assembly is designed for a specific purpose, rather than to create an exhaustive set of assembly configurations. This weak supervision results in a positive and unlabeled (PU) learning problem [2] where the joints are known positive labels, but the remaining negative labels could be positive (i.e. an unseen but plausible joint) or negative (i.e. an implausible joint). To address the data imbalance and PU learning problem, we organize and augment our data using the following three techniques.

Joint Consolidation To increase the number of positive labels, we consolidate joints between identical pairs of parts into *joint sets*. Figure 4, right shows an example joint set where the same two parts are connected in multiple different ways. This approach allows us to present the network with a *single* data sample, i.e. a joint set, that contains all known joints between a pair of parts. Importantly, joint consolidation avoids presenting the network with multiple contradictory data samples, where a negative label in one sample may be a positive label in another sample.

Joint Equivalents To further counter the extreme data imbalance, we identify and label 'equivalent' entities that share the same joint axis as the ground truth. For example, if a circle is the labeled entity (highlighted in cyan in Figure 2a), its neighbouring faces, such as the cylinder highlighted in pink, will be labeled as equivalent. These entities represent the same user-selected joint axis and only differ by the origin point that locates the joint axis in 3D space. As we consider a predicted joint axis to be correct if it is co-linear with the ground truth joint axis, we include equivalent labels during *evaluation*.

Unambiguous Evaluation Sets A challenge with PU Learning is establishing a 'clean' test set to accurately measure network performance. Parts that have multiple plausible joints, such as a plate with multiple holes for fasteners, are problematic if only partial joint labels exist, leading to ambiguity at test time. We make a best effort to avoid positive unlabelled samples in the test and validation set by excluding geometrically similar, but unlabeled, 'sibling' entities, e.g. the faces and edges of an unlabeled hole with the same size as a labelled hole. We identify sibling entities by matching the entity type, area or length, and number of connected graph edges to the labeled entities.

3.5. JoinABLe Architecture

Our overall architecture is shown in Figure 3 and consists of an encoder module that outputs per-vertex embeddings for each B-Rep face and edge in our graph representation of

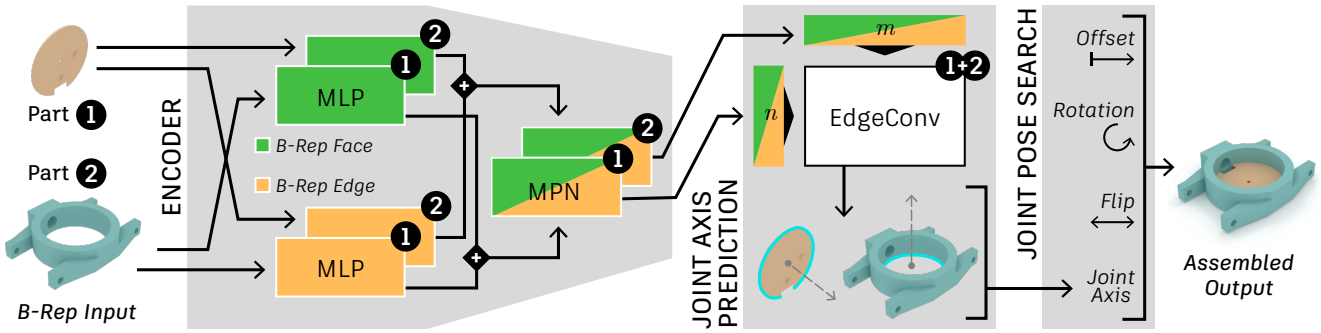

Figure 3. *JoinABLe* architecture. Given two B-Rep parts in our graph representation, the vertex features from B-Rep faces (green) and edges (orange) pass through separate multi-layer perceptrons (MLP) before being concatenated together and passed through a message passing network (MPN). This yields local vertex embeddings representing each B-Rep entity in the two parts. Our joint axis prediction branch then performs edge convolution *between* the two graphs to estimate the presence of joints over all possible pairs of connections. Finally, the joint parameters are discovered via search, with respect to the predicted joint axis, to complete the assembly.

the input parts. Using these embeddings we can predict a joint axis and then search for joint pose parameters.

3.5.1 Encoder

Our encoder neural network f_{enc} is a Siamese-style network with shared weights for the two parts. It firstly creates graph vertex embeddings by passing the vertex features $\mathbf{x}_1$ and $\mathbf{x}_2$ from the two graphs, through two separate multi-layer perceptrons (MLP). One MLP is used for vertices representing B-Rep faces and another for those representing B-Rep edges; the resulting vertex embeddings are then concatenated together. We next perform message passing *within* each part's graph using a two-layer Graph Attention Network v2 (GATv2) [4] to obtain the per-vertex embeddings $\mathbf{h}_1$ and $\mathbf{h}_2$ for both the graphs.

$$\mathbf{h}_1 = f_{\text{enc}}(\mathbf{x}_1, G_1), \quad \mathbf{h}_2 = f_{\text{enc}}(\mathbf{x}_2, G_2).$$

The idea here is to extract local features within each part that consider each B-Rep entity and its neighborhood.

3.5.2 Joint Axis Prediction

When creating a joint, a key piece of design intent is the definition of a joint axis by which two parts can be aligned and constrained to one another. The joint axis forms the basis for the degrees of freedom to be defined and enables downstream tasks such as assembly, part mobility, and animation. We formulate joint axis prediction as a link prediction problem, where the goal is to correctly identify a connection between G_1 and G_2 that aligns the two parts along a ground truth joint axis. This is done by aggregating information *between* parts using an edge convolution along the edges of G_j. The node features $\mathbf{x}_1$ and $\mathbf{x}_2$ from graphs G_1 and G_2, are passed through our shared encoder network f_{enc} to get 384-dimensional embeddings $\mathbf{h}_1$ and $\mathbf{h}_2$. Then for each edge (u, v) in the graph G_j which densely connects G_1 and G_2, we predict a logit indicating the presence of a joint:

$$\mathbf{h}_{uv} = \phi(\mathbf{h}_u \oplus \mathbf{h}_v),$$

where $\phi : \mathbb{R}^{768} \mapsto \mathbb{R}$ is a 3-layer MLP, $\oplus$ is the concatenation operator and $\mathbf{h}_u$ and $\mathbf{h}_v$ are gathered from $\mathbf{h}_1$ and $\mathbf{h}_2$ based on the source and target vertices for each edge in G_j.

We train the network with a loss function that has two terms. The first term $\mathcal{L}_{\text{CE}}$ is the cross-entropy between the edge predictions $\mathbf{h}_{uv}$ and the ground truth edge labels $\mathbf{j}_{uv} \in \{0, 1\}$ normalized into a probability distribution $\widehat{\mathbf{j}}_{uv}$.

$$\widehat{\mathbf{h}}_{uv} = \text{softmax}_{\text{all}}(\mathbf{h}_{uv}),$$
$$\mathcal{L}_{\text{CE}} = \text{CE}\left(\widehat{\mathbf{j}}_{uv}, \widehat{\mathbf{h}}_{uv}\right).$$

Here the subscript in the softmax operation indicates that it is applied over all edges in G_j, and $\text{CE}(\mathbf{p}, \mathbf{q}) = -\sum_i \mathbf{p}_i \log \mathbf{q}_i$. This loss encourages true joints to have higher values while simultaneously suppressing non-joints. We observe this is sub-optimal due to the sparsity of positive labels, where $\mathcal{L}_{\text{CE}}$ is summed over a large number of terms. To better focus the loss term so that the joints are better contrasted against more likely non-joints, we use a symmetric cross entropy loss $\mathcal{L}_{\text{Sym}}$ as the second term in the loss function.

$$\widehat{\mathbf{h}}_{\text{row}} = \text{softmax}_{\text{row}}(\mathbf{h}_{\text{2D}}), \quad \widehat{\mathbf{h}}_{\text{col}} = \text{softmax}_{\text{col}}(\mathbf{h}_{\text{2D}}),$$
$$\mathcal{L}_{\text{Sym}} = \text{CE}(\widehat{\mathbf{j}}_{\text{2D}}, \widehat{\mathbf{h}}_{\text{row}}) + \text{CE}(\widehat{\mathbf{j}}_{\text{2D}}, \widehat{\mathbf{h}}_{\text{col}}).$$

Here the subscript of the softmax indicates that it is taken over a single axis, and the 2D subscript instead of uv indicates that the predictions and ground truth labels on the edges of G_j are reshaped into $n \times m$ matrices.

3.5.3 Joint Pose Search

The B-Rep entities predicted by our network allow us to query the ground truth B-Rep data to obtain a joint axis prediction for each part. Once these axes are aligned together, three secondary parameters define a rigid joint and can be used for joint pose prediction. An *offset* distance along the joint axis, *rotation* about the joint axis, and a *flip* parameter to reverse the joint axis direction. We find these parameters using a neurally guided search that allows us to enumerate over the top-k joint axis predictions and directly consider interaction between both parts. To evaluate a candidate joint configuration, we propose a cost function $\mathcal{C}_{\text{joint}} = \mathcal{C}_{\text{overlap}} + \lambda \mathcal{C}_{\text{contact}}$ that considers two general criteria for well defined joints: overlap volume and contact area between parts, formulated as:

$$\mathcal{C}_{\text{overlap}} = \frac{\mathcal{V}_{1\cap 2}}{\min\left(\mathcal{V}_1, \mathcal{V}_2\right)}, \quad \mathcal{C}_{\text{contact}} = \frac{A_{1\cap 2}}{\min\left(A_1, A_2\right)}.$$

Here, $\mathcal{V}_1$ and $\mathcal{V}_2$ are the volume of the two parts, and $\mathcal{V}_{1\cap 2}$ represents their overlap volume. Similarly, A_1 and A_2 are the surface area of the two parts, and their contact area is $A_{1\cap 2}$. Intuitively, for the two parts to align closely to each other, minimizing the cost function should encourage a larger contact area while penalizing the overlap volume to prevent penetration. Therefore, we let $\lambda = -10$ if $\mathcal{C}_{\text{overlap}} < 0.1$. Otherwise, we set $\lambda = 0$ to increase the overlap penalty. Given this cost function, we search for the optimal joint pose using the Nelder-Mead algorithm [39] as a standard derivative-free optimization.

4. Dataset

To evaluate the performance of our method we create the *Fusion 360 Gallery* assembly dataset, derived from designs created in Autodesk Fusion 360 and submitted to the publicly available Autodesk Online Gallery [1]. The dataset consists of two inter-related sets of data, *Assembly Data*, containing 8,251 assemblies with 154,468 separate parts, and *Joint Data* containing 32,148 joints defined between 23,029 different parts. The data and supporting code will be publicly available on GitHub[1] with a license allowing non-commercial research.

We now describe the joint data used in our experiments. Figure 4, left shows an overview of the joint data in our dataset. We consider a data sample to be a joint set, such as shown in Figure 4, right, containing a pair of parts with one or more joints defined between them. The user-selected B-Rep faces and edges form the ground truth labels together with the joint axis and pose information of each joint. We provide an approximate 70/10/10/10% data split, for the training, validation, test, and original distribution test sets respectively. The validation and test sets do not include samples with potentially ambiguous sibling entities, while the original distribution test set does.

[1] https://github.com/AutodeskAILab/Fusion360GalleryDataset

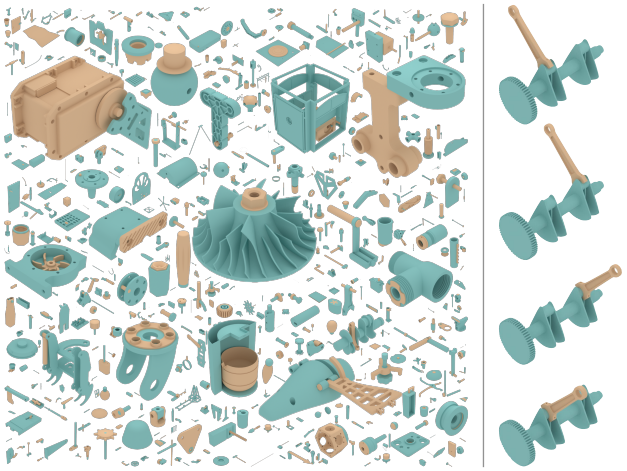

Figure 4. An overview of joint data from the *Fusion 360 Gallery* assembly dataset (left). Each sample consists of a unique pair of parts with *one or more* joints defining how they are locally constrained and positioned together (right).

5. Experiments

In this section we perform experiments to qualitatively and quantitatively evaluate our method on two tasks: joint axis prediction and joint pose prediction. We examine how our method compares with a human CAD expert and other methods from the literature. A key criteria for evaluating performance is to gauge how the network performs in scenarios that traditional algorithms find challenging. One such scenario involves designs that do not contain connections between cylindrical shafts and holes, such as a bolt and a hole similar to Figure 2a. Commercial products exist which infer joints of this type by searching for fasteners and holes with similar radii [44]. In our dataset we see that 82% of data samples contain holes and 47.5% of joints constrain circular or cylindrical entities on one part to a hole on the opposing part. In our experiments we report results that gauge the ability of our approach to correctly infer joints in the simple *Hole* case and the more complex *No Hole* case.

5.1. Human CAD Expert Baseline

Understanding how a human CAD expert performs in a similar setting is important to gauge the efficacy of each method. We conduct a study to establish a human baseline by recruiting a CAD expert, who works on commercial CAD design, and ask them to assemble pairs of parts from our dataset with a known ground truth joint. We use 100 data samples picked randomly from a distribution exclud-

	All Acc.% ↑	Hole Acc.% ↑	No Hole Acc.% ↑	Param. # ↓
Ours	**79.53**	**80.15**	**76.59**	**1.3M**
B-Dense	10.59	10.36	10.59	3.2M
B-Discrete	4.28	4.18	4.79	4.0M
B-Grid	65.21	65.09	65.81	3.1M
B-Heuristic	71.39	72.74	64.97	-
B-Random	21.55	21.92	23.29	-
Human	80.00	-	-	-

Table 1. Joint axis prediction accuracy results are shown for all data samples in the test set (All), the subset of data samples with holes (Hole) and without holes (No Hole). The number of network parameters is also shown (Param.). Finally, results from a human CAD expert on 100 test samples are shown.

ing the potentially ambiguous sibling entities. We randomly rotate and translate each part and conduct the study using Fusion 360. We compare the joint axis created by the CAD expert with the ground truth. We find that the CAD expert results match the ground truth 80% of the time. This shows that determining how two isolated parts should be assembled is challenging for CAD experts without the valuable context provided by the object assembly.

5.2. Joint Axis Prediction

Although there are no previous works that address the exact same setting as ours, we adapt several related methods to compare with our approach.

Point Cloud Baselines We adapt two point cloud based methods designed to predict a joint axis for part mobility. For each baseline we use a common architecture, based on a PointNet++ [40] encoder, and adapt the decoder strategy and loss functions from related work. **B-Dense** follows Li et al. [31] to densely regress a joint origin projection vector, projection distance, and joint direction for each point in the point cloud. **B-Discrete** follows Shape2Motion [48] and uses a hybrid of discrete classification and regression to predict the joint origin point and direction vector.

B-Rep Baselines We compare our method against several baseline methods that take B-Rep graphs as input. **B-Grid** follows UV-Net [19] and uses grid features (points, normals, trimming mask, and tangents) sampled on B-Rep faces and edges together with a CNN encoder. We use the same graph topology, prediction head, and loss as our network. **B-Heuristic** uses a rule-based approach that operates on B-Rep graphs and assigns a score to each B-Rep entity. Higher scores are assigned to entities that are similar, based on the entity type, area, and length information, and that match the training data distribution of entity type pairings. For cylinders and circles, the radius of the entity is also employed. A higher score is given to entity pairs where the radii match to within 5%. **B-Random** makes random predictions over all B-Rep entities and represents the lower bound for B-Rep performance.

	All CD ↓	Hole CD ↓	No Hole CD↓	Param. # ↓
Ours + Search	**0.0580**	**0.0570**	**0.0628**	**1.3M**
Ours	0.0627	0.0624	0.0657	**1.3M**
B-Pose	0.0700	0.0693	0.0730	2.3M

Table 2. Joint pose prediction results using average chamfer distance (CD) where lower is better. We show results for all samples in the test set (All), and the subset of data samples with holes (Hole) and without holes (No Hole). The number of network parameters is also shown (Param.)

Table 1 shows results for the joint axis prediction task on the test set. We report the accuracy of regression based approaches by considering a joint axis prediction to be a 'hit' if it is collinear within a distance and angular threshold of 5%. For classification based approaches we report the top-1 accuracy. We also report accuracy for the subset of data samples that have holes (*Hole*) and those that do not (*No Hole*). Recall that traditional algorithms are good at working with the special case of matching fasteners to holes. We observe that the performance gap between our approach and the next highest performing B-Heuristic approach is 8.14%, however this widens to 11.62% for the important *No Hole* subset where traditional algorithms are known to struggle. We find that the B-Rep based approaches outperform those based on point clouds while also using fewer parameters. Although point cloud approaches perform well with axis aligned parts from the same object class [48], our results show that real world data is significantly more challenging. Finally we note that our approach is within 0.5% of the performance of a human CAD expert.

5.3. Joint Pose Prediction

For the joint pose prediction task we again adapt a baseline method from the literature to our setting. **B-Pose** follows Huang et al. [17] to regress a translation point and rotation quaternion using a combination of L2 and chamfer distance (CD) loss terms. Although a parametric joint is not created, B-Pose represents a common approach used with top-down assembly. We evaluate the performance of our method in two different configurations. **Ours** uses the joint axes derived from network predictions to align the two parts together without an offset, rotation, or flip. **Ours + Search** additionally performs joint pose search over the top 50 predictions to find suitable offset, rotation, and flip parameters.

Table 2 shows results for the joint pose prediction task. We record the minimum CD calculated between the ≥ 1

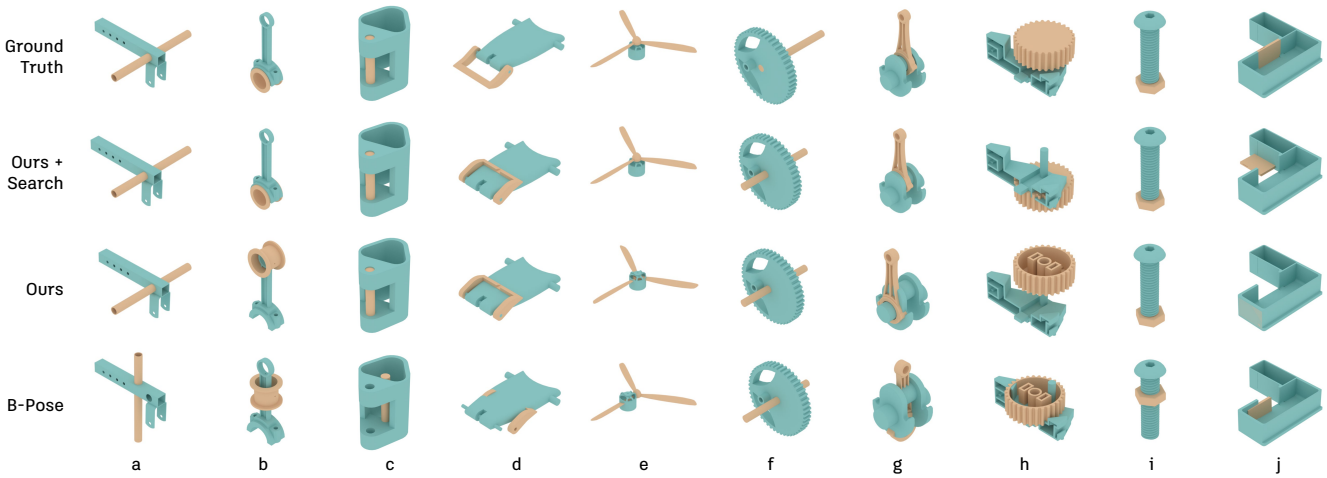

Figure 5. Qualitative comparison of joint pose prediction results comparing our method, with and without search, with the B-Pose baseline.

ground truth joints from the joint set and the predicted assembly. We then report the average CD across all samples in the test set. We find that using our network predictions alone (Ours) can better match the ground truth when compared with the B-Pose baseline. Introducing search (Ours + Search) can help resolve areas of overlap (Figure 5e) and in some cases resolve incorrect axis predictions (Figure 5b,g). It is important to note that the ground truth data only contains a finite set of discrete states (e.g. door open, door closed) rather than continuous states (e.g. door *opening*) that may also be valid. For example, our predictions for the belt buckle in Figure 5d do not match the ground truth state but appear plausible. As such, CD should be considered an approximate metric for comparing the relative performance of each method.

6. Discussion

Future Applications Our joint axis prediction network and search approach can serve as fundamental building blocks for a number of applications. One such application is the automated assembly of multiple parts in a design. As a preliminary demonstration we assemble a multi-part design given only the individual parts and the sequence of part pairs derived from our assembly dataset. We amend our search strategy to minimize the overlap volume between the new part and the partially assembled design at each assembly step and maximize the contact area between them using a similar cost function. Figure 6 shows an example sequence of parts that are assembled correctly in a bottom-up fashion.

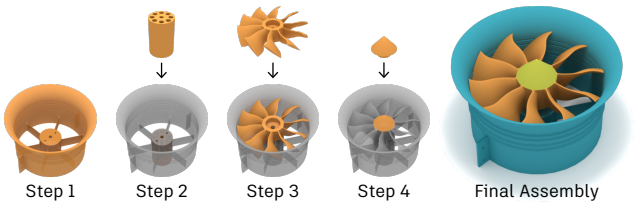

Figure 6. Multi-part assembly demonstration. Parts are aligned sequentially from a given assembly sequence using our joint axis prediction network and pose search.

Limitations A bottom-up approach to assembly may be limited when scaling to large assemblies where global composition is important. We believe a promising avenue to pursue in future research is a hybrid approach that combines the precision of bottom-up assembly with the compositional ability of top-down approaches. Reliance on B-Rep CAD data is another limitation of the current work, although mitigated by recent dataset releases [21, 24, 53].

7. Conclusion

Our long-term motivation is to enable assembly-aware design tools, capable of suggesting and automatically placing parts. Such a system could enable greater reuse of existing physical components in new designs and potentially reduce the cost and environmental impact associated with manufacturing and associated supply chains [23]. Understanding how parts are assembled is also critical for robotic assembly and disassembly. CAD-informed robotic disassembly systems may enhance our ability to reuse and recycle components [5, 29, 34, 36]. In this work we have begun the first steps to address these challenges by learning the bottom-up assembly of parametric CAD joints. Our results show the promise of learning based methods to approach the performance of human CAD experts, and with the publication of our dataset we hope to further aid future research.

References

[1] Autodesk. *Autodesk Online Gallery*, 2015. 6

[2] Jessa Bekker and Jesse Davis. Learning from positive and unlabeled data: A survey. *Machine Learning*, 109(4):719–760, 2020. 4

[3] Flavien Boussuge, Christopher M Tierney, Trevor T Robinson, and Cecil G Armstrong. Application of tensor factorisation to analyze similarities in cad assembly models. In *International Meshing Roundtable and User Forum*, 2019. 1

[4] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv:2105.14491*, 2021. 5

[5] MML Chang, SK Ong, and AYC Nee. Approaches and challenges in product disassembly planning for sustainability. *Procedia Cirp*, 60:506–511, 2017. 2, 8

[6] Siddhartha Chaudhuri, Daniel Ritchie, Jiajun Wu, Kai Xu, and Hao Zhang. Learning generative models of 3d structures. In *Computer Graphics Forum (CGF)*, volume 39, pages 643–666. Wiley Online Library, 2020. 2

[7] LS Homem De Mello and Arthur C Sanderson. A correct and complete algorithm for the generation of mechanical assembly sequences. In *1989 IEEE International Conference on Robotics and Automation*, pages 56–57. IEEE Computer Society, 1989. 2

[8] Chi-Wing Fu, Peng Song, Xiaoqi Yan, Lee Wei Yang, Pradeep Kumar Jayaraman, and Daniel Cohen-Or. Computational interlocking furniture assembly. *ACM Transactions on Graphics (TOG)*, 34(4):1–11, 2015. 2

[9] Thomas Funkhouser, Michael Kazhdan, Philip Shilane, Patrick Min, William Kiefer, Ayellet Tal, Szymon Rusinkiewicz, and David Dobkin. Modeling by example. *ACM Transactions on Graphics (TOG)*, 23(3):652–663, 2004. 2

[10] Matheus Gadelha, Giorgio Gori, Duygu Ceylan, Radomir Mech, Nathan Carr, Tamy Boubekeur, Rui Wang, and Subhransu Maji. Learning generative models of shape handles. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 402–411, 2020. 2

[11] Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao Zhang. Sdm-net: Deep generative network for structured deformable mesh. *ACM Transactions on Graphics (TOG)*, 38(6), 2019. 2

[12] Somaye Ghandi and Ellips Masehian. Review and taxonomies of assembly and disassembly path planning problems and approaches. *Computer-Aided Design*, 67:58–86, 2015. 2

[13] Dan Halperin, J-C Latombe, and Randall H Wilson. A general framework for assembly planning: The motion space approach. *Algorithmica*, 26(3):577–601, 2000. 2

[14] Songfang Han, Jiayuan Gu, Kaichun Mo, Li Yi, Siyu Hu, Xuejin Chen, and Hao Su. Compositionally generalizable 3d structure prediction. *arXiv:2012.02493*, 2020. 2

[15] Abhinav Narayan Harish, Rajendra Nagar, and Shanmuganathan Raman. Rgl-net: A recurrent graph learning framework for progressive part assembly. *arXiv:2107.12859*, 2021. 1, 2

[16] Haibin Huang, Evangelos Kalogerakis, and Benjamin Marlin. Analysis and synthesis of 3d shape families via deep-learned generative models of surfaces. In *Computer Graphics Forum*, volume 34, pages 25–38. Wiley Online Library, 2015. 2

[17] 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. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 1, 2, 7

[18] Arjun Jain, Thorsten Thormählen, Tobias Ritschel, and Hans-Peter Seidel. Exploring shape variations by 3d-model decomposition and part-based recombination. In *Computer Graphics Forum*, volume 31, pages 631–640. Wiley Online Library, 2012. 2

[19] Pradeep Kumar Jayaraman, Aditya Sanghi, Joseph G. Lambourne, Karl D.D. Willis, Thomas Davies, Hooman Shayani, and Nigel Morris. Uv-net: Learning from boundary representations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11703–11712, June 2021. 4, 7

[20] Pablo Jiménez. Survey on assembly sequencing: a combinatorial and geometrical perspective. *Journal of Intelligent Manufacturing*, 24(2):235–250, 2013. 2

[21] Benjamin Jones, Dalton Hildreth, Duowen Chen, Ilya Baran, Vova Kim, and Adriana Schulz. Automate: A dataset and learning approach for automatic mating of cad assemblies. *arXiv:2105.12238*, 2021. 2, 8

[22] 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 Transactions on Graphics (TOG)*, 39(6), 2020. 2

[23] Wendy Kerr and Chris Ryan. Eco-efficiency gains from remanufacturing: A case study of photocopier remanufacturing at fuji xerox australia. *Journal of cleaner production*, 9(1):75–81, 2001. 2, 8

[24] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9601–9611, 2019. 8

[25] Joseph G. Lambourne, Karl D.D. Willis, Pradeep Kumar Jayaraman, Aditya Sanghi, Peter Meltzer, and Hooman Shayani. Brepnet: A topological message passing system for solid models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12773–12782, June 2021. 4

[26] Manfred Lau, Akira Ohgawara, Jun Mitani, and Takeo Igarashi. Converting 3d furniture models to fabricatable parts and connectors. *ACM Transactions on Graphics (TOG)*, 30(4):1–6, 2011. 2

[27] Sang Hun Lee and Kunwoo Lee. Partial entity structure: A compact non-manifold boundary representation based on partial topological entities. In *Proceedings of the Sixth ACM Symposium on Solid Modeling and Applications*, SMA '01, pages 159—170, New York, NY, USA, 2001. Association for Computing Machinery. 1

[28] Youngwoon Lee, Edward S Hu, Zhengyu Yang, Alex Yin, and Joseph J Lim. Ikea furniture assembly environment for long-horizon complex manipulation tasks. *arXiv:1911.07246*, 2019. 1

[29] J Li, M Barwood, and S Rahimifard. A multi-criteria assessment of robotic disassembly to support recycling and recovery. *Resources, Conservation and Recycling*, 140:158–165, 2019. 2, 8

[30] Jun Li, Chengjie Niu, and Kai Xu. Learning part generation and assembly for structure-aware shape synthesis. In *AAAI Conference on Artificial Intelligence*, volume 34, pages 11362–11369, 2020. 2

[31] Xiaolong Li, He Wang, Li Yi, Leonidas J Guibas, A Lynn Abbott, and Shuran Song. Category-level articulated object pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3706–3715, 2020. 3, 7

[32] Yichen Li, Kaichun Mo, Lin Shao, Minhyuk Sung, and Leonidas Guibas. Learning 3d part assembly from a single image. *European Conference on Computer Vision (ECCV)*, 2020. 1, 2

[33] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007. 4

[34] Quan Liu, Zhihao Liu, Wenjun Xu, Quan Tang, Zude Zhou, and Duc Truong Pham. Human-robot collaboration in disassembly for sustainable manufacturing. *International Journal of Production Research*, 57(12):4027–4044, 2019. 2, 8

[35] Katia Lupinetti, Jean-Philippe Pernot, Marina Monti, and Franca Giannini. Content-based cad assembly model retrieval: Survey and future challenges. *Computer-Aided Design*, 113:62–81, 2019. 1, 2

[36] Marco Marconi, Giacomo Palmieri, Massimo Callegari, and Michele Germani. Feasibility study and design of an automatic system for electronic components disassembly. *Journal of Manufacturing Science and Engineering*, 141(2), 2019. 2, 8

[37] Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas J Guibas. Structurenet: Hierarchical graph networks for 3d shape generation. *ACM Transactions on Graphics (TOG)*, 38(6), 2019. 2

[38] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 909–918, 2019. 2

[39] John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965. 6

[40] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5099–5108, 2017. 7

[41] Nadav Schor, Oren Katzir, Hao Zhang, and Daniel Cohen-Or. Componet: Learning to generate the unseen by part synthesis and composition. In *IEEE International Conference on Computer Vision (ICCV)*, pages 8759–8768, 2019. 2

[42] Adriana Schulz, Ariel Shamir, David IW Levin, Pitchaya Sitthi-Amorn, and Wojciech Matusik. Design and fabrication by example. *ACM Transactions on Graphics (TOG)*, 33(4):1–11, 2014. 2

[43] Tianjia Shao, Dongping Li, Yuliang Rong, Changxi Zheng, and Kun Zhou. Dynamic furniture modeling through assembly instructions. *ACM Transactions on Graphics (TOG)*, 35(6), 2016. 2

[44] SolidWorks. *Smart Fasteners*. 1, 6

[45] Peng Song, Chi-Wing Fu, Yueming Jin, Hongfei Xu, Ligang Liu, Pheng-Ann Heng, and Daniel Cohen-Or. Reconfigurable interlocking furniture. *ACM Transactions on Graphics (TOG)*, 36(6):1–14, 2017. 2

[46] Minhyuk Sung, Hao Su, Vladimir G Kim, Siddhartha Chaudhuri, and Leonidas Guibas. Complementme: weakly-supervised component suggestions for 3d modeling. *ACM Transactions on Graphics (TOG)*, 36(6):1–12, 2017. 1, 2

[47] Garrett Thomas, Melissa Chien, Aviv Tamar, Juan Aparicio Ojea, and Pieter Abbeel. Learning robotic assembly from cad. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3524–3531. IEEE, 2018. 2

[48] Xiaogang Wang, Bin Zhou, Yahao Shi, Xiaowu Chen, Qinping Zhao, and Kai Xu. Shape2motion: Joint analysis of motion parts and attributes from 3d shapes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8876–8884, 2019. 1, 3, 7

[49] Ziqi Wang, Peng Song, and Mark Pauly. Desia: a general framework for designing interlocking assemblies. *ACM Transactions on Graphics (TOG)*, 37(6):1–14, 2018. 2

[50] Ziqi Wang, Peng Song, and Mark Pauly. State of the art on computational design of assemblies with rigid parts. *Annual Conference of the European Association for Computer Graphics (EuroGraphics)*, 40(2), 2021. 2

[51] K.J. Weiler. *Topological structures for geometric modeling*. Technical report RPI, Center for Interactive Computer Graphics. University Microfilms, 1986. 1, 3

[52] Karl D. D. Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G. Lambourne, Armando Solar-Lezama, and Wojciech Matusik. Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences. *ACM Transactions on Graphics (TOG)*, 40(4), 2021. 4

[53] Rundi Wu, Chang Xiao, and Changxi Zheng. Deepcad: A deep generative network for computer-aided design models. In *IEEE International Conference on Computer Vision (ICCV)*, pages 6772–6782, October 2021. 8

[54] Rundi Wu, Yixin Zhuang, Kai Xu, Hao Zhang, and Baoquan Chen. Pq-net: A generative part seq2seq network for 3d shapes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 829–838, 2020. 2

[55] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. SAPIEN: A simulated part-based interactive environment. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[56] Kai Xu, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. Fit and diverse: set evolution for inspiring 3d shape galleries. *ACM Transactions on Graphics (TOG)*, 31(4):1–10, 2012. 2

[57] Xianghao Xu, Wenzhe Peng, Chin-Yi Cheng, Karl D.D. Willis, and Daniel Ritchie. Inferring cad modeling sequences using zone graphs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6062–6070, June 2021. 4

[58] Zihao Yan, Ruizhen Hu, Xingguang Yan, Luanmin Chen, Oliver van Kaick, Hao Zhang, and Hui Huang. Rpm-net: Recurrent prediction of motion and parts from point cloud. *Annual Conference on Computer Graphics and Interactive Techniques Asia (SIGGRAPH Asia)*, 38(6):240:1–240:15, 2019. 3

[59] Jie Yang, Kaichun Mo, Yu-Kun Lai, Leonidas J Guibas, and Lin Gao. Dsm-net: Disentangled structured mesh net for controllable generation of fine geometry. *arXiv:2008.05440*, 2020. 2

[60] Kangxue Yin, Zhiqin Chen, Siddhartha Chaudhuri, Matthew Fisher, Vladimir Kim, and Hao Zhang. Coalesce: Component assembly by learning to synthesize connections. *arXiv:2008.01936*, 2020. 2

[61] Allan Zhao, Jie Xu, Mina Konaković-Luković, Josephine Hughes, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. Robogrammar: graph grammar for terrain-optimized robot design. *ACM Transactions on Graphics (TOG)*, 39(6):1–16, 2020. 1

[62] Youyi Zheng, Daniel Cohen-Or, and Niloy J Mitra. Smart variations: Functional substructures for part

compatibility. In *Computer Graphics Forum*, volume 32, pages 195–204. Wiley Online Library, 2013. 2