The Shape Part Slot Machine: Contact-based Reasoning for Generating 3D Shapes from Parts

Kai Wang[1], Paul Guerrero[2], Vladimir Kim[2], Siddhartha Chaudhuri[2], Minhyuk Sung[2,3], and Daniel Ritchie[1]

[1] Brown University
[2] Adobe Research
[3] KAIST

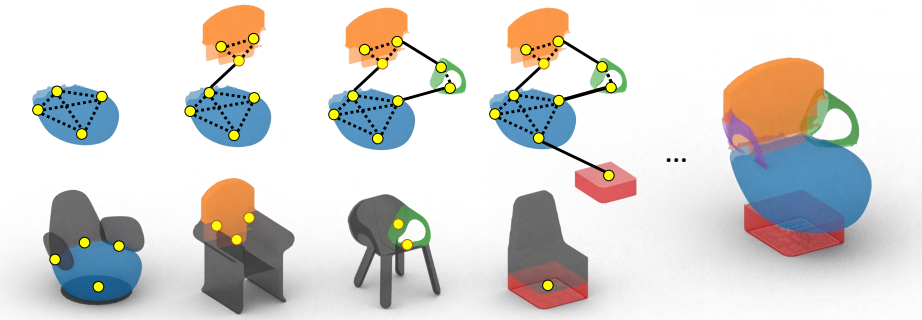

Fig. 1: Our system synthesizes novel 3D shapes by assembling them from parts. Internally, it represents shapes as a graph of the regions where parts connect to one another (which we call slots). It generates such a graph by retrieving part subgraphs from different shapes in a dataset. Once a full graph has been generated, the system then optimizes for affine part transformations to produce a final output shape.

Abstract. We present the Shape Part Slot Machine, a new method for assembling novel 3D shapes from existing parts by performing contact-based reasoning. Our method represents each shape as a graph of "slots," where each slot is a region of contact between two shape parts. Based on this representation, we design a graph-neural-network-based model for generating new slot graphs and retrieving compatible parts, as well as a gradient-descent-based optimization scheme for assembling the retrieved parts into a complete shape that respects the generated slot graph. This approach does not require any semantic part labels; interestingly, it also does not require complete part geometries—reasoning about the slots proves sufficient to generate novel, high-quality 3D shapes. We demonstrate that our method generates shapes that outperform existing modeling-by-assembly approaches regarding quality, diversity, and structural complexity.

1 Introduction

There is increasing demand for high-quality 3D object models across multiple fields: gaming and virtual reality; advertising and e-commerce; synthetic training data for computer vision and robotics; and more. The traditional practice of manual 3D modeling is

time-consuming and labor-intensive and is not well-suited to scaling to this demand. Thus, visual computing researchers have pursued data-driven methods which can augment human creative capabilities and accelerate the modeling process.

One promising technology in this space are generative models of 3D shapes. Such generative models could suggest new, never-before seen shapes, freeing users from tedious and time-consuming low-level geometric manipulations to focus on high-level creative decisions. Recent work in this space has focused on deep generative models of shapes in the form of volumetric occupancy grids, point clouds, or implicit fields. While these methods demonstrate impressive abilities to synthesize the bulk shape of novel objects, the local geometry they produce often exhibits noticeable artifacts: oversmoothing, bumpiness, extraneous holes, etc. At present, none of these generative models has achieved geometric output quality resembling the shapes they are trained on. An alternative approach would be to avoid synthesizing novel geometry altogether and instead learn how to re-combine existing high-quality geometries created by skilled modeling artists. This paradigm is known in the computer graphics literature as *modeling by assembly*, where it once received considerable attention. Since the deep learning revolution, however, the focus of most shape generation research has shifted to novel geometry synthesis. The few post-deep-learning methods for modeling by assembly have shown promise but have not quite lived up to it: handling only coarse-grained assemblies of large parts, as well as placing parts by directly predicting their world-space poses (leading to 'floating part' artifacts).

In this paper, we present a new generative model for shape synthesis by part assembly which addresses these issues. Our key idea is to use a representation which focuses on the connectivity structure of parts. This choice is inspired by several recent models for novel geometry synthesis which achieve better structural coherence in their outputs by adopting a part-connectivity-based representation [15, 10, 25]. In our model, the first-class entities are the regions where one part connects to another. We call these regions *slots* and our model the *Shape Part Slot Machine*.

In our model, a shape is represented by a graph in which slots are nodes and edges denote connections between them. We define shape synthesis as iteratively constructing such a graph by retrieving parts and connecting their slots together. We propose an autoregressive generative framework for solving this problem, composed of several neural network modules tasked with retrieving compatible parts and determining their slot connections. Throughout the iterative assembly process, the partial shape is represented only by its slot graph: it is not necessary to assemble the retrieved parts together until the process is complete, at which point we use a gradient-descent-based optimization scheme to find poses and scales for the retrieved parts which are consistent with the generated slot graph.

We compare the Shape Part Slot Machine to other modeling-by-assembly and part-connectivity-based generative models. We find that our approach consistently outperforms the alternatives in its ability to generate visually and physically plausible shapes.

In summary, our contributions are:

- The *Slot graph* representation for reasoning about part structure of shapes.
- An autoregressive generative model for slot graphs by iterative part retrieval and assembly.

– A demonstration that local part connectivity structure is enough to synthesize globally-plausible shapes: neither full part geometries nor their poses are required.

2 Related Work

Modeling by Part Assembly: The Modeling By Example system pioneered the paradigm of modeling-by-assembly with interactive system for replacing parts of an object by searching in database [4]. The Shuffler system added semantic part labels, enabling automatic 'shuffling' of corresponding parts [13]. Later work handled more complex shapes by taking symmetry and hierarchy into account [9]. Other modes of user interaction include guiding exploration via sketches [23] or abstract shape templates [1], searching for parts by semantic attributes [2], or having the user play the role of fitness function in an evolutionary algorithm [24]. Probabilistic graphical models have been effective for suggesting parts [3] or synthesizing entire shapes automatically [12]. Part-based assembly has also been used for reconstructing shapes from images [18].

Our work is most closely related to ComplementMe, which trains deep networks to suggest and place unlabeled parts to extend a partial shape [19]. Our model is different in that we use a novel, part-contacts-only representation of shapes, which we show enables handling of more structurally complex shapes.

Deep Generative Models of Part-based Shapes: Our work is also related to deep generative models which synthesize part-based shapes. One option is to make voxel-grid generative models part-aware [20, 22]. Many models have been proposed which generate sets of cuboids representing shape parts [27]; some fill the cuboids with generated geometry in the form of voxel grids [14] or point clouds [15, 10, 11]. Other part-based generative models skip cuboid proxies and generate part geometries directly, as point clouds [17], implicit fields [21], or deformed genus zero meshes [5, 25]. All of these models synthesize part geometry. In contrast, our model synthesizes shapes by retrieving and assembling existing high-quality part meshes.

Estimating Poses for 3D Parts: Many part-based shape generative models must pose the generated parts. Some prior work looks at this problem on its own: given a set of parts, how to assemble them together? One method predicts a 6DOF pose for each part such that they become assembled [8]; another predicts per-part translations and scales and also synthesizes geometry to make the transformed parts connect seamlessly [26]. Rather than predict part poses directly, we solve for per-part poses and scales that satisfies contact constraints encoded in a slot graph. This approach has its root in early work in modeling by assembly [12] but without the need for part labels and separate steps computing how parts should attach. It is also similar in spirit to that of ShapeAssembly [10], working with part meshes rather than cuboid abstractions.

3 Overview

Assembling novel shapes from parts requires solving two sub-problems: finding a set of compatible parts, and computing the proper transforms to assemble the parts. These tasks depend on each other, e.g. replacing a small chair seat with a large one will shift

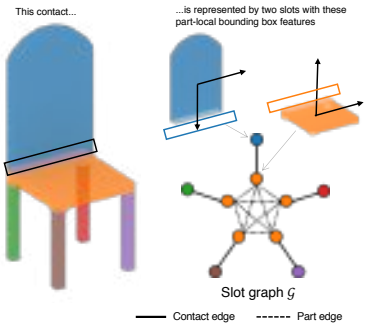

Fig. 2: A slot graph. Nodes are part-to-part contact regions called *slots* and describe the contact geometry with bounding boxes. *Contact edges* connect two slots on two adjacent parts, while *part edges* connect all slots of the same part.

the chair legs further away from the center. Instead of solving these sub-problems separately, we propose a system for solving them jointly. Specifically, our system synthesizes shapes by iteratively constructing a representation of the contacts between parts. The assembly transformations for each part can then be computed directly from this representation.

In our system, a shape is represented as a *slot graph*: each node corresponds to a "slot" (part-to-part contact region) on a part; each edge is either a *part edge* connecting slots of the same part or a *contact edge* connecting slots on two touching parts. Section 4 defines this graph structure and describes how we extract them from available data.

This reduces the task of assembling novel shapes to a graph generation problem: retrieving sub-graphs representing parts from different shapes and combining them into new graphs. We solve this problem autoregressively, assembling part sub-graphs one-by-one into a complete slot graph. At each iteration, given a partial slot graph, our system inserts a new part using three neural network modules: the first determines *where* a part be should connect to the current partial graph, the second decides *what* part to connect, and third determines *how* to connect the part. We describe this generation process in Section 5.

Finally, given a complete contact graph, the system runs a gradient-based optimization process that assembles parts into shapes by solving for poses and scales of the individual parts such that the contacts implied by the generated slot graph are satisfied. We describe the process in Section 6.

4 Representing Shapes with Slot Graphs

In this section, we define slot graphs, describe how we extract them from segmented shapes, and how we encode them with neural networks.

4.1 Slot-based Graph Representation of Shapes

A good shape representation that models how parts connect allows the generative model to reason independently about part connectivity and part geometry. Given a shape S and

its part decomposition $\{P_1 \dots P_N\}$, we call regions where parts connect "slots", and use them as nodes in a graph $\mathcal{G} = (V, E_c, E_p)$, as illustrated in Figure 2. Each pair of parts may be connected with multiple slots, and each slot $\mathbf{u}_{ij} \in V$ on part P_i that connects to P_j has a corresponding slot $\mathbf{u}_{ji}$ on part P_j that connects back to P_i. Each node $\mathbf{u}_{ij}$ stores the following properties:

- The axis-aligned bounding box (AABB) of the slot, in a coordinate frame centered on P_i.
- The same AABB, but normalized such that the bounding box of the entire part is a unit cube. This provides a scale-invariant view of how parts connect.

A slot graph $\mathcal{G}$ has two types of edges: *contact edges* $\mathbf{e}^c_{ij} \in E_c$ connect every pair of contacting slots $\mathbf{u}_{ij}, \mathbf{u}_{ji}$ and *part edges* $\mathbf{e}^p_{ijk} \in E_p$ connect every pair of slots $\mathbf{u}_{ij}, \mathbf{u}_{ik}$ in the same part P_i.

This representation encodes neither the geometry nor the pose of each part. Omitting this information encourages generalization: the choice of parts will be based only on the compatibility of their attachment regions and connectivity structure; it will not be biased by a part's world-space position in its original shape nor its complete geometry.

This representation also does not encode part symmetries; nevertheless, we demonstrate in Section 7 that our model often generates appropriately symmetrical shapes. We can also optionally include logic that enforces symmetries at test time (see Section 5).

4.2 Extracting Slot Graphs from Data

Given a set of part-segmented shapes, we follow StructureNet [15] to extract part adjacencies and symmetries. We use adjacencies to define slots, and define connecting regions as points within distance τ of the adjacent part. Additionally, we ensure that symmetrical parts have symmetrical slots and prune excess slots where multiple parts overlap at the same region. See supplemental for details.

4.3 Encoding Slots Graphs with Neural Networks

We encode a slot graph into a graph feature $h_{\mathcal{G}}$ and per-slot features $h_{\mathbf{u}}$ using messaging passing networks [6].

Initializing Node Embeddings: We initialize slot features $h_{\mathbf{u}}$ using a learned encoding of the slot properties $x_{\mathbf{u}}$ (two six-dimensional AABBs) with a three-layer MLP f_{init}: $h^0_{\mathbf{u}} = f_{\text{init}}(x_{\mathbf{u}})$. As we discuss later, some of our generative model's modules also include an additional one-hot feature which is set to one for particular nodes that are relevant to their task (effectively 'highlighting' them for the network).

Graph Encoder: The node embeddings are then updated with our message passing network using an even number of message passing rounds. In each round, node embeddings are updated by gathering messages from adjacent nodes. We alternate the edge sets E during each round, using only part edges $E = E_p$ for odd rounds ($t = 1, 3, 5 \dots$) and only contact edges $E = E_c$ for even rounds ($t = 2, 4, 6 \dots$):

$$h_{\mathbf{t}} = f^t_{\text{update}}\Big(h^{t-1}_{\mathbf{u}}, \sum_{\mathbf{uv} \in E} f^t_{\text{msg}}(h^{t-1}_{\mathbf{u}}, h^{t-1}_{\mathbf{v}}, h^0_{\mathbf{u}}, h^0_{\mathbf{v}})\Big)$$

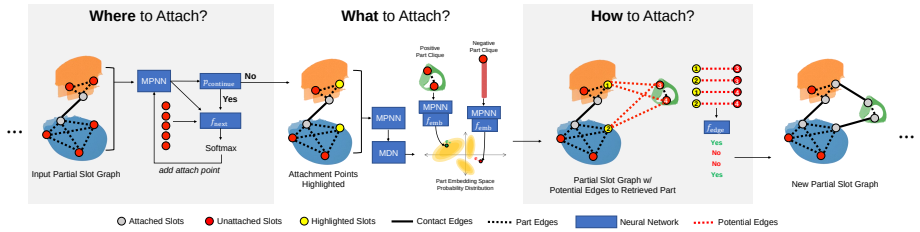

Fig. 3: Our slot graph generative model uses three neural network modules to build a graph step by step. *Where to Attach?*: Predicts which slots on the current partial shape the next-retrieved part should be attached to. *What to Attach?*: Learns an embedding space for part slot graphs and predicts a probability distribution over this space in which parts which are compatible with the highlighted slots have high probability. *How to attach?*: Determines which slots on the retrieved part should connect to which slots on the current partial shape.

where f_{msg} is a multi-layer perceptron (MLP) that computes a message for each pair of adjacent nodes, and f_{update} is a MLP that updates the node embedding from the summed messages. We also include skip connections to the initial node embeddings $h_{\mathbf{u}}^0$. All MLPs have separate weights for each round of message passing.

Gathering Information from the Graph: To obtain the final node features $h_{\mathbf{u}}$, we concatenate its initial embedding with as its embeddings after every even round of message passing (i.e. those using contact edges) and feed them into an MLP f_{node}:

$$h_{\mathbf{u}} = f_{\text{node}}(h_{\mathbf{u}}^0, h_{\mathbf{u}}^2 \dots h_{\mathbf{u}}^T)$$

To obtain the feature $h_{\mathcal{G}}$ of an entire graph, we first perform a graph readout over the embeddings at round t:

$$h_{\mathcal{G}}^t = \sum_{\mathbf{u} \in V} (f_{\text{project}}(h_{\mathbf{u}}^t) \cdot f_{\text{gate}}(h_{\mathbf{u}}^t))$$

where f_{project} projects node features into the latent space of graph features and f_{gate} assigns a weight for each of the mapped features. We then compute the final graph feature $h_{\mathcal{G}}$ similar to the way we compute the node features:

$$h_{\mathcal{G}} = f_{\text{graph}}(h_{\mathcal{G}}^0, h_{\mathcal{G}}^2 \dots h_{\mathcal{G}}^T)$$

In cases where we need a feature for a subset of nodes $V' \subset V$ in the graph, we simply perform the readout over V' instead of V.

5 Generating Slot Graphs

Our system casts shape synthesis by part assembly as a problem of generating novel slot graphs. To do so, we first extract a graph representation of each part: for every shape S in a dataset of shapes, and for every part $P_i \in S$ represented by a slot graph $\mathcal{G} = (V, E_c, E_p)$, we create a part clique $C_{P_i} \subseteq \mathcal{G}$ representing this part by taking all the slots $\mathbf{u}_{ij} \in V$ associated with this part, as well as all the part edges $\mathbf{e}_{ijk} \in E_p$. We

remove the all contact edges $\mathbf{e}_{ij} \in E_c$ that connects P_i to other parts in the shape. Our goal, then, is find a set of part cliques $\mathbf{C}$ that can be connected together into a novel slot graph $\mathcal{G}' = (V', E'_c, E'_p)$, where $V' = \{\mathbf{u} \in \mathbf{C}\}$, $E'_p = \{\mathbf{e} \in \mathbf{C}\}$, and E'_c is the set of contact edges that need to be added to make all the slots attached i.e. connected to exactly one other slot via a contact edge.

There can be thousands of parts in available shape datasets, each containing multiple slots that can be attached in different ways. Thus, it is infeasible to search this combinatorial space exhaustively. Instead, we *learn* how to build novel slot graphs autoregressively, attaching one part clique at a time to a partial slot graph, until it is complete (i.e. all slots are attached). We learn this autoregressive process with teacher forcing: for every training sample, we take a random, connected partial slot graph $\mathcal{G}'$ consisting of one or more part cliques from a graph $\mathcal{G} = (V, E_c, E_P)$ extracted from a dataset shape S. We then select a random part clique $C_{P_j} | P_j \in S$ (referred to as C_{target} in the following sections) that is attached to $\mathcal{G}'$ on one or more slots $V_{\text{target}} = \{\mathbf{u}_{ij} \mid \mathbf{u}_{ij} \in \mathcal{G}', \mathbf{u}_{ji} \in C_{P_j}\}$ via set of contact edges $E_{\text{target}} = \{\mathbf{e}^c_{ij} \mid \mathbf{u}_{ij} \in V_{\text{target}}\}$. The goal of a single generation step, then, is to maximize

$$p(V_{\text{target}}, C_{\text{target}}, E_{\text{target}} \mid \mathcal{G}')$$

Rather than learn this complex joint distribution directly, we instead factor it into three steps using the chain rule:

- **Where** to attach: maximizing $p(V_{\text{target}} \mid \mathcal{G}')$
- **What** to attach: maximizing $p(C_{\text{target}} \mid \mathcal{G}', V_{\text{target}})$
- **How** to attach: maximizing $p(E_{\text{target}} \mid \mathcal{G}', V_{\text{target}}, C_{\text{target}})$

In the remainder of this section, we detail the formulation for the networks we use for each of the three steps, as well as how we use them during test time. Figure 3 visually illustrates these steps.

Where to Attach?: Given a partial slot graph $\mathcal{G}'$, we first identify the slots V_{target} to which the next-retrieved part clique should attach. We predict each element of V_{target} autoregressively (in any order), where each step i takes as input $\mathcal{G}'$ and the already-sampled slots $V^i_{\text{target}} = \{V_{\text{target},0} \dots V_{\text{target},i-1}\}$ (highlighted in $\mathcal{G}'$ with a one-hot node feature). We first use a MLP f_{continue} to predict the probability p_{continue} that another slot should be added ($p_{\text{continue}} = 0$ if $V^i_{\text{target}} = V_{\text{target}}$ and 1 otherwise). If more slots should be included, then we use an MLP f_{next} to predict a logit for each of the unattached slots $\tilde{V}$ in $\mathcal{G}'$ that are not already in V^i_{target}. These logits are then fed into a softmax to obtain a probability distribution over possible slots:

$$p(V_{\text{target}} | \mathcal{G}') = \prod_{i=1}^{|V_{\text{target}}|} p^i_{\text{cont}} \cdot p^i_{\text{next}}[V_{\text{target},i}]$$

$$p^i_{\text{cont}} = \begin{cases} p_{\text{continue}}(h_{\mathcal{G}'} | V^i_{\text{target}}) & i < |V_{\text{target}}| \\ 1 - p_{\text{continue}}(h_{\mathcal{G}'} | V^i_{\text{target}}) & i = |V_{\text{target}}| \end{cases}$$

$$p^i_{\text{next}} = \text{softmax}(\left[f_{\text{next}}(h_{\mathbf{u}} | V^i_{\text{target}}) | \mathbf{u} \in \tilde{V} / V^i_{\text{target}}\right])$$

What to Attach?: Having selected the slots V_{target} to attach to, we then retrieve part cliques compatible with the partial graph $\mathcal{G}'$ and the selected slots. Similar to prior

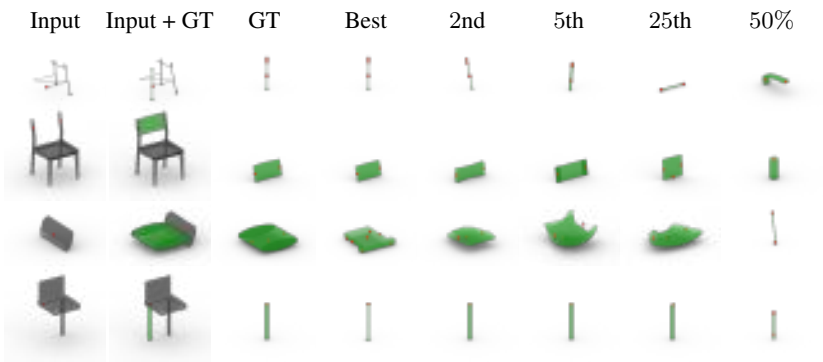

Fig. 4: Example outputs of the **What to Attach?** module. We visualize the input partial slot graph within the parts that contain them (grey) and the center of the selected slots (red), as well as the ground truth part (green, 2nd column). The parts and slots are in their ground truth world-space pose, which is *not* available to the neural network. We then visualize, individually, the ground truth part and the retrieved candidates ranked 1st, 2nd, 5th, 25th, and at the 50th percentile, respectively, along with all of their slots (red).

work [19], we take a contrastive learning approach to this problem: the probability of the ground truth part clique should be greater than that of randomly sampled other part cliques (i.e. negative examples) by some margin m:

$$p(C_{\text{target}} \mid \mathcal{G}', V_{\text{target}}) > p(C_{\text{negative}} \mid \mathcal{G}', V_{\text{target}}) + m$$

We use two neural networks to enforce this property. The first maps part cliques C into an embedding space $\mathbb{R}_{\text{emb}}$.

$$X_C = f_{\text{emb}}(h_C)$$

where f_{emb} is the embedding MLP and h_C is the graph feature computed from C alone. The second network is a mixture density network (MDN) that outputs a probability distribution over this embedding space:

$$P(X|\mathcal{G}', V_{\text{target}}, X \in \mathbb{R}_{\text{emb}}) = \text{MDN}(h_{\mathcal{G}'}, h_{\mathcal{G}'_{\text{target}}})$$

Where V_{target} are highlighted in the input node features and $h_{\mathcal{G}'_{\text{target}}}$ is obtained by computing graph features using V_{target} only.

We visualize the behavior of this module trained on Chair in Figure 4. When the input demands a very specific type of structure (first 2 rows), our module can retrieve the part cliques that match such structure. When the input has fewer constraints (3rd row), our module retrieves a wide variety of partial cliques that can be attached. In the 4th row, our module retrieves chair legs that are *structurally* compatible. The legs are not necessarily *geometrically* compatible, as geometry information is not available to the module.

How to Attach?: The last module learns to connect the retrieved part clique C_{target} to the partial slot graph $\mathcal{G}'$. It predicts a probability for every pair of slots $\mathbf{u}_{ij} \in V_{\text{target}}, \mathbf{u}_{ji} \in C_{\text{target}}$ that could be connected via a contact edge:

$$p(\mathbf{e}^c_{ij} \mid \mathcal{G}, V_{\text{target}}, C_{\text{target}}) = f_{\text{edge}}(h_{\mathbf{u}_{ij}}, h'_{\mathbf{u}_{ji}})$$

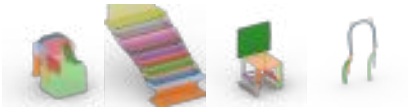

Fig. 5: Typical structural outliers detected at test time. From left to right: redundant component (chair back), repetition of structures, inability to resolve local connections (chair base), not enough slots to finish structure.

Where V_{target} are highlighted in input node features, f_{edge} is a MLP and $h_{\mathbf{u}_{ij}}$ and $h'_{\mathbf{u}_{ji}}$ are computed with two neural networks, one over $\mathcal{G}'$ and one over $\mathcal{C}_{\text{target}}$. $p(\mathbf{e}^c_{ij}) = 1$ if $\mathbf{e}^c_{ij} \in E_{\text{target}}$ and 0 otherwise. If both V_{target} and C_{target} contain one slot, then these slots must be connected, and this module can be skipped. To encourage the networks to learn more general representations, we augment V_{target} with random unattached slots in $\mathcal{G}'$.

Generating New Slot Graphs at Test Time: At test time, we generate new slot graphs by iteratively querying the three modules defined above. Although the modules we learn are probabilistic and support random sampling, we find MAP inference sufficient to produce a diverse range of shapes. We terminate the generation process when the slot graph is complete i.e. when all slots in the graph are attached to exactly one slot from a different part.

This stopping criterion, while simple, is not robust to errors: a random set of part cliques can occasionally form a complete slot graph by chance. To combat these errors, we reject any partial shapes for which the "how" module assigns low probabilities to all candidate slot pairs. We also use one-class Support Vector Machines to reject other structural outliers (see Figure 5 for examples).

Finally, we also include logic to enforce part symmetries. When retrieving a part to connect with slots that were part of a symmetry group in their original shape, we alter the rank order of parts returned by our "what" module to prioritize (a) parts that occurred in symmetry groups in the dataset, followed by (b) parts that are approximately symmetrical according to chamfer distance. See the supplemental for more details about these test-time sampling strategies.

6 Assembling Shapes From Slot Graphs

A generated slot graph defines connectivity between shape parts but does not explicitly give part poses. Thus, our system has an additional step to find world-space poses for all the retrieved parts. In this section, we describe a simple gradient-descent-based optimization procedure for doing so, which takes a generated slot graph $\mathcal{G} = (V, E_c, E_p)$ that describes N parts $P_1 \dots P_N$, and predicts an affine transformation matrix T_i for each part P_i.

Objective Function: To assemble a shape from its slot graph, we want each slot to be connected in the same way as it was in its original shape, which we approximate by enforcing that the distance from any point on a slot to the contacting slot should stay the same in the assembled shape. Formally, for each slot $\mathbf{u}_{ij} \in V$, we select the set of points S_{ij} that the slot contains (from the point sample of P_i from Section 4). For each point $p \in S_{ij}$, we compute its distance $d_o(p)$ to the closest point on the slot that was originally connected to $\mathbf{u}_{ij}$ in the dataset. We then optimize for $T_1 \dots T_N$ via the

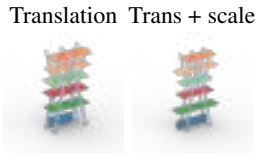

Fig. 6: Optimizing part affine transformations to satisfy a slot graph. We show the output of the initial translation-only phase of optimization & the final output with both translation and scale.

following objective: for every slot $\mathbf{u}_{ij} \in V$, every point sample $p \in S_{ij}$ should have the same distance to the connecting slot $\mathbf{u}_{ji}$ as the original distance $d_o(p)$:

$$f(T_1 \dots T_N) = \sum_{\mathbf{u}_{ij} \in V} \sum_{p \in S_{ij}} \Big(\big(\min_{q \in S_{ji}} d(T_i p, T_j q) \big) - d_o(p) \Big)^2$$

Optimization Process: We minimize this objective using gradient descent. Rather than full affine transformations, we optimize only translation and anisotropic scale. This prevents certain part re-uses from happening (e.g. re-using a horizontal crossbar as a vertical crossbar), but we find that the space of possible outputs is nonetheless expressive. To minimize unnecessary distortion, we prefer translation over scaling whenever possible: we optimize for translation only for the first 1000 iterations, and then alternate between translation and scaling every 50 iterations for the next 500 iterations. Optimizing for scales is essential in cases where translation alone cannot satisfy the slot graph. We show one such example in Figure 6, where the shelf layers are scaled horizontally to match the V shape of the frame.

7 Results & Evaluation

In this section, we evaluate our method's ability to synthesize novel shapes. We use the PartNet [16] dataset, segmenting shapes with the finest-grained PartNet hierarchy level and filtering out shapes with inconsistent segmentations and/or disconnected parts. More implementation details are in the supplemental.

Novel Shape Synthesis: Figure 7 shows examples of shapes our method is capable of assembling. Our model learns to generate shapes with complex structures specific to each shape categories. Though it does not use global part position information during graph generation, the resulting slot graphs lead to consistent global positions for the individual parts once optimized. Although we choose not to encode full geometries, our model is still able to generate shapes with interesting variations both structurally and geometrically. We further quantitatively compare the results generated by our model against against these alternatives:

- **ComplementMe** [19] is the previous state-of-art for modeling by part assembly. It retrieves compatible parts and places them together by predicting per-part translation vectors. ComplementMe also does not implement a stopping criteria for generation, so we train a network that takes a partial shape point cloud as input and predicts whether generation should stop. We also stop the generation early if the output of the part retrieval network does not change from one step to the next.

Fig. 7: Examples of range of shapes our method is able to generate. Each part has a different color that correlates with the order they are inserted. The blue part is used for initialization. See the supplementary material for more details about the color palette.

Fig. 8: (a): Chairs in the first row of Figure 7, where parts coming from the same source shape now have the same color. (b): Geometric nearest neighbor of the the same chairs in the training set. (c): Chairs generated without enforcing part symmetries. (d): Chairs generated with the explicit rule that no parts coming from the same source shape can be attached together. Our method uses parts from different shapes to generate novel shapes. It can generate approximately symmetric shapes without explicit rules, and can connect parts from different shapes together plausibly.

,

Table 1: Comparing our system to baselines and ablations on generating visually and physically plausible shapes.

Category	Method	Root↑	Stab↑	Fool↑	FD↓	Parts
Chair	Ours	98.1	70.1	6.4	61.1	7.8
	ComplementMe	90.2	41.1	6.6	83.0	5.7
	StructureNet	81.0	61.3	4.0	37.5	12.1
	Oracle	94.5	83.4	25.8	13.3	–
	ComplementMe (w/sym)	88.3	79.1	21.9	21.6	4.3
	Ground Truth	100.0	100.0	–	–	11.1
	Ours (no symmetry)	98.2	68.0	8.1	58.9	7.8
	Ours (no duplicate)	97.5	67.4	13.8	61.2	7.7
Table	Ours	98.2	82.8	10.6	61.6	6.8
	ComplementMe	90.2	62.0	7.7	93.5	4.7
	StructureNet	82.8	78.5	2.3	85.2	7.8
	ComplementMe (w/sym)	87.1	84.0	35.8	18.7	3.3
	Ground Truth	100.0	100.0	–	–	9.3
Storage	Ours	99.4	90.8	15.5	42.6	6.9
	ComplementMe	91.4	72.4	8.9	89.8	3.4
	StructureNet	89.6	82.2	6.8	105.5	8.3
	ComplementMe (w/sym)	85.3	70.6	11.5	71.4	3.3
	Ground Truth	100.0	99.4	–	–	13.6
Lamp	Ours	89.6	–	21.5	42.4	3.4
	ComplementMe	62.0	–	35.8	26.0	3.4
	Ground Truth	92.6	–	–	–	4.2

Finally, ComplementMe relies on a part discovery process where most groups of symmetrical parts are treated as a single part (e.g. four chair legs). We notice that, when trained on our data, ComplementMe suffers from a significant performance decrease on Chair and Table, and struggles to generate more complex Storage, as is evident from the average number of parts (See Table 1). Therefore, for these categories, we also include results where parts are grouped by symmetry (w/sym) for reference. We stress that, under this condition, both retrieving and assembling parts are significantly easier, thus the results are not directly comparable.

– **StructureNet** [15] is an end-to-end generative model outputs a hierarchical shape structure, where each leaf node contains a latent code that can either be decoded into a cuboid or a point cloud. We modify it to output meshes by, for each leaf node, retrieving the part in the dataset whose StructureNet latent code is closest to the leaf node's latent code and then transforming the part to fit the cuboid for that leaf node.
– We also include an **Oracle** as an upper bound on retrieval-based shape generation. The oracle is an autoregressive model that takes as input at each step (a) the bounding boxes for all parts in a ground-truth shape and (b) point clouds for parts

retrieved so far. Retrieved parts are scaled so that they fit exactly to the bounding box to which they are assigned.

See supplemental for more details about these baselines. We use an evaluation protocol similar to ShapeAssembly [10] which evaluates both the physical plausibility and quality of generated shapes:

- **Rootedness ↑ (Root)** measures the percentage of shapes for which there is a connected path between the ground to all parts;
- **Stability ↑ (Stable)** measures the percentage of shapes that remains upright under gravity and a small force in physical simulation, we do not report this for lamps because lamps such as chandeliers do not need to be stable;
- **Realism ↑ (Fool)** is the percentage of test set shapes classified as "generated" by a PointNet trained to distinguish between dataset and generated shapes;
- **Freschet Distance ↓ (FD)** [7] measures distributional similarity between generated and dataset shapes in the feature space of a pre-trained PointNet classifier.
- **Parts** is the mean number of parts in generated shapes.

Table 1 summarizes the results. By using a contact-based representation, our model is able to generate shapes that are more physically plausible (rooted and stable) than the baselines. while being comparable in terms of the overall shape quality, measured by Frechet distance and classifier fool percentage. Our model performs particularly well for storage furniture; we hypothesize rich connectivity information of this shape category allows our model to pick parts that match particularly well. Our model fares less well on lamps, where connectivity structure is simple and the geometric variability of parts (which out model does not encode) is highly variable. ComplementMe works well on lamps, thanks to its focus on part geometry. Its performance drops significantly on all other categories with more complicated shape structures. We provide more details, as well as random samples for all methods, in the supplementary material.

Generalization Capacity: It is important that a generative model that follows the modeling by assembly paradigm learns to recombine parts from different sources into novel shapes. We demonstrate our model's capacity for this in Figure 8: it is able to assemble parts from multiple source shapes together into novel shapes different from those seen during training, with or without explicit restrictions whether parts from the same source shape can be connected to each other. We also see that while including symmetry reasoning improves geometric quality, our method is able to generate shapes that are roughly symmetrical without it. This is also reflected in Table 1: removing symmetry or prohibiting using multiple parts from the same source shape has minimal impact on our metrics. We provide more analysis of generalization in the supplemental.

Performance of Individual Modules: Finally, we evaluate each individual model module, using the following metrics:

- **Attach Acc**: How often the "where" module correctly selects the slots to attach, given the first slot.
- **Average Rank**: Average percentile rank of ground truth next part according to the "what" module.
- **Edge Acc**: How often the "how" module recovers the the ground truth edge pairs.

Table 2 summarizes the results. Modules perform very well, with some lower numbers caused by inherent multimodality of the data.

Table 2: Evaluating our neural network modules in isolation.

	Attach Acc	Avg Rank	Edge Acc
Chair	96.70	99.05	94.79
Table	92.32	99.14	92.82
Storage	87.46	99.08	85.38
Lamp	98.87	91.36	91.89

Fig. 9: Typical failure cases of our method. From left to right: a chair with a tiny seat, two opposite-facing lamps attached together awkwardly, a chair with a implausible back, a chair that misses seat and legs completely.

Limitations: Even with outlier detection as mentioned in section 5, poor-quality outputs can still occur. Figure 9 shows typical examples. Most are caused by our model's lack of focus on geometry: chairs with a tiny seat, lamps that face opposite directions, and chair backs that block the seat completely. Incorporating additional geometric features when appropriate could help.

8 Conclusion

We presented the Shape Part Slot machine, a new modeling-by-part-assembly generative model for 3D shapes. Our model synthesizes new shapes by generating slot graphs describing the contact structure between parts; it then assembles its retrieved parts by optimizing per-part affine transforms to be consistent with this structure. The slot graph encodes surprisingly little information, yet we demonstrated experimentally that our model outperforms multiple baselines and prior modeling-by-assembly systems on generating novel shapes from PartNet parts.

There are multiple directions for future work. Parts could be repurposed in more diverse ways if we had a method to transfer slot graphs between geometrically- and contextually-similar parts (so e.g. a chair seat that had armrests originally does not have to have them in all synthesized results). More variety could also be obtained by optimizing for part orientations (so e.g. a vertical slat could be used as a horizontal one).

Acknowledgements This work was funded in part by NSF award #1907547 and a gift fund from Adobe. Daniel Ritchie is an advisor to Geopipe and owns equity in the company. Geopipe is a start-up that is developing 3D technology to build immersive virtual copies of the real world with applications in various fields, including games and architecture. Minhyuk Sung acknowledges the support of NRF grant (2022R1F1A1068681), NST grant (CRC 21011), and IITP grant (2022-0-00594) funded by the Korea government (MSIT), and grants from Adobe, KT, and Samsung Electronics corporations. Part of this work was done when Kai Wang interned at Adobe.

References

1. Averkiou, M., Kim, V., Zheng, Y., Mitra, N.J.: Shapesynth: Parameterizing model collections for coupled shape exploration and synthesis. Computer Graphics Forum (Special issue of Eurographics 2014) (2014)
2. Chaudhuri, S., Kalogerakis, E., Giguere, S., Funkhouser, T.: AttribIt: Content creation with semantic attributes. ACM Symposium on User Interface Software and Technology (UIST) (Oct 2013)
3. Chaudhuri, S., Kalogerakis, E., Guibas, L., Koltun, V.: Probabilistic reasoning for assembly-based 3d modeling. ACM Trans. Graph. **30**(4) (Jul 2011)
4. Funkhouser, T., Kazhdan, M., Shilane, P., Min, P., Kiefer, W., Tal, A., Rusinkiewicz, S., Dobkin, D.: Modeling by example. In: ACM SIGGRAPH 2004 Papers (2004)
5. Gao, L., Yang, J., Wu, T., Yuan, Y.J., Fu, H., Lai, Y.K., Zhang, H.R.: Sdm-net: Deep generative network for structured deformable mesh. In: SIGGRAPH Asia (2019)
6. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. CoRR **arXiv:1704.01212** (2017)
7. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: NeurIPS (2017)
8. Huang, J., Zhan, G., Fan, Q., Mo, K., Shao, L., Chen, B., Guibas, L., Dong, H.: Generative 3d part assembly via dynamic graph learning. In: The IEEE Conference on Neural Information Processing Systems (NeurIPS) (2020)
9. Jain, A., Thormählen, T., Ritschel, T., Seidel, H.P.: Exploring shape variations by 3d-model decomposition and part-based recombination. Comput. Graph. Forum **31**(2pt3), 631–640 (May 2012). https://doi.org/10.1111/j.1467-8659.2012.03042.x, https://doi.org/10.1111/j.1467-8659.2012.03042.x
10. Jones, R.K., Barton, T., Xu, X., Wang, K., Jiang, E., Guerrero, P., Mitra, N.J., Ritchie, D.: Shapeassembly: Learning to generate programs for 3d shape structure synthesis. ACM Transactions on Graphics (TOG), SIGGRAPH Asia 2020 **39**(6), Article 234 (2020)
11. Jones, R.K., Charatan, D., Guerrero, P., Mitra, N.J., Ritchie, D.: Shapemod: Macro operation discovery for 3d shape programs. In: SIGGRAPH 2021 (2021)
12. Kalogerakis, E., Chaudhuri, S., Koller, D., Koltun, V.: A probabilistic model for component-based shape synthesis. ACM Trans. Graph. **31**(4) (Jul 2012)
13. Kreavoy, V., Julius, D., Sheffer, A.: Model composition from interchangeable components. In: Proceedings of the 15th Pacific Conference on Computer Graphics and Applications. p. 129–138. PG '07, IEEE Computer Society, USA (2007). https://doi.org/10.1109/PG.2007.43, https://doi.org/10.1109/PG.2007.43
14. Li, J., Xu, K., Chaudhuri, S., Yumer, E., Zhang, H., Guibas, L.: GRASS: Generative Recursive Autoencoders for Shape Structures. In: SIGGRAPH 2017 (2017)
15. Mo, K., Guerrero, P., Yi, L., Su, H., Wonka, P., Mitra, N., Guibas, L.: StructureNet: Hierarchical graph networks for 3D shape generation. In: SIGGRAPH Asia (2019)
16. Mo, K., Zhu, S., Chang, A.X., Yi, L., Tripathi, S., Guibas, L.J., Su, H.: Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 909–918 (2019)
17. Schor, N., Katzir, O., Zhang, H., Cohen-Or, D.: Componet: Learning to generate the unseen by part synthesis and composition. In: The IEEE International Conference on Computer Vision (ICCV) (October 2019)
18. Shen, C.H., Fu, H., Chen, K., Hu, S.M.: Structure recovery by part assembly. ACM Trans. Graph. **31**(6) (Nov 2012). https://doi.org/10.1145/2366145.2366199, https://doi.org/10.1145/2366145.2366199

19. Sung, M., Su, H., Kim, V.G., Chaudhuri, S., Guibas, L.: ComplementMe: Weakly-supervised component suggestions for 3D modeling. ACM Transactions on Graphics (TOG) **36**(6), 226 (2017)
20. Wang, H., Schor, N., Hu, R., Huang, H., Cohen-Or, D., Huang, H.: Global-to-local generative model for 3d shapes. ACM Transactions on Graphics (Proc. SIGGRAPH ASIA) **37**(6), 214:1–214:10 (2018)
21. Wu, R., Zhuang, Y., Xu, K., Zhang, H., Chen, B.: Pq-net: A generative part seq2seq network for 3d shapes. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020)
22. Wu, Z., Wang, X., Lin, D., Lischinski, D., Cohen-Or, D., Huang, H.: Sagnet: Structure-aware generative network for 3d-shape modeling. ACM Transactions on Graphics (Proceedings of SIGGRAPH 2019) **38**(4), 91:1–91:14 (2019)
23. Xie, X., Xu, K., Mitra, N.J., Cohen-Or, D., Gong, W., Su, Q., Chen, B.: Sketch-to-design: Context-based part assembly. Computer Graphics Forum **xx**(xx), xx (2013)
24. Xu, K., Zhang, H., Cohen-Or, D., Chen, B.: Fit and diverse: Set evolution for inspiring 3d shape galleries. ACM Transactions on Graphics, (Proc. of SIGGRAPH 2012) **31**(4), 57:1–57:10 (2012)
25. Yang, J., Mo, K., Lai, Y.K., Guibas, L.J., Gao, L.: Dsm-net: Disentangled structured mesh net for controllable generation of fine geometry (2020)
26. Yin, K., Chen, Z., Chaudhuri, S., Fisher, M., Kim, V., Zhang, H.: Coalesce: Component assembly by learning to synthesize connections. In: Proc. of 3DV (2020)
27. Zou, C., Yumer, E., Yang, J., Ceylan, D., Hoiem, D.: 3D-PRNN: Generating Shape Primitives with Recurrent Neural Networks. In: ICCV 2017 (2017)