

Joint Learning of 3D Shape Retrieval and Deformation

Mikaela Angelina Uy¹ Vladimir G. Kim² Minhyuk Sung³ Noam Aigerman²
 Siddhartha Chaudhuri^{2,4} Leonidas Guibas¹

¹Stanford University ²Adobe Research ³KAIST ⁴IIT Bombay

Abstract

We propose a novel technique for producing high-quality 3D models that match a given target object image or scan. Our method is based on retrieving an existing shape from a database of 3D models and then deforming its parts to match the target shape. Unlike previous approaches that independently focus on either shape retrieval or deformation, we propose a joint learning procedure that simultaneously trains the neural deformation module along with the embedding space used by the retrieval module. This enables our network to learn a deformation-aware embedding space, so that retrieved models are more amenable to match the target after an appropriate deformation. In fact, we use the embedding space to guide the shape pairs used to train the deformation module, so that it invests its capacity in learning deformations between meaningful shape pairs. Furthermore, our novel part-aware deformation module can work with inconsistent and diverse part-structures on the source shapes. We demonstrate the benefits of our joint training not only on our novel framework, but also on other state-of-the-art neural deformation modules proposed in recent years. Lastly, we also show that our jointly-trained method outperforms various non-joint baselines.

1. Introduction

Creating high-quality 3D models from a reference image or a scan is a laborious task, requiring significant expertise in 3D sculpting, meshing, and UV layout. While neural generative techniques for 3D shape synthesis hold promise for the future, they still lack the ability to create 3D models that rival the fidelity, level of detail, and overall quality of artist-generated meshes [38]. Several recent techniques propose to directly retrieve a high-quality 3D model from a database and deform it to match a target image or point cloud, thereby approximating the target shape while preserving the quality of the original source model. These prior methods largely focus on one of two complementary subproblems: either retrieving an appropriate mesh from a database [26, 6], or training a neural network to deform a source to a target [14, 43, 49, 36]. In most cases, the static database mesh most closely matching the target is retrieved,

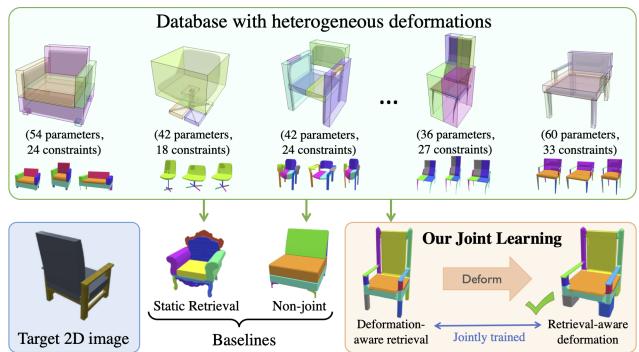


Figure 1. Given an input target we use jointly-learned retrieval and deformation modules to find a source model in a heterogeneous database and align it to the target. We demonstrate that our joint learning outperforms static retrieval and non-joint baselines.

and then deformed for a better fit [19]. The retrieval step is not influenced by the subsequent deformation procedure, and thus ignores the possibility that a database shape with different global geometry nevertheless possess local details that will produce the best match *after* deformation.

Only a few works explicitly consider *deformation-aware* retrieval [34, 41]. However, in these works the deformation module is a fixed, non-trainable black box, which requires complete shapes (and not e.g., natural images or partial scans) as targets, does not handle varying shape structures across the database, may necessitate time-consuming, manually-specified optimization of a fitting energy, exhaustive enumeration of deformed variants, and does not support back-propagating gradients through it for directly translating deformation error to retrieval error.

In this paper, we argue that retrieval and deformation should be *equal citizens in a joint problem*. Given a database of source models equipped with some parametric representation of deformations, our goal is to learn how to retrieve a shape from the database and predict the optimal deformation parameters so it best matches a given target. A key feature of our method is that both retrieval and deformation are *learnable* modules, each influencing the other and trained jointly. While the benefit of deformation-aware retrieval has been explored previously, we contribute the notion of *retrieval-aware deformation*: our learnable deforma-

tion module is optimized for fitting retrieved shapes to target shapes, and does not try to be a general-purpose algorithm for arbitrary source-target pairs. Thus, the retrieval module is optimized to retrieve sources that the deformation module can fit well to the input target, and the deformation module is trained on sources the retrieval module predicts for the input target, thereby letting it optimize capacity and learn only meaningful deformations.

The robustness of the joint training enables us to devise a more elaborate deformation space. Specifically, we devise a *differentiable, part-aware deformation function* that deforms individual parts of a model while respecting the part-to-part connectivity of the original structure (Figure 1). Importantly, it accommodates varying numbers of parts and structural relationships across the database, and does not require part labels or consistent segmentations. It can work with automatically-segmented meshes and even multiple differently segmented instances of the same source shape. We propose a way to encode each part in each source and to enable a general MLP to predict its deformation regardless of the part count. This holistic view of joint retrieval and deformation is especially important when considering heterogeneous collections of shapes “in the wild” that often vary in their part structure, topology, and geometry. These require different deformation spaces for different source models, which the retrieval module must be aware of.

We evaluate our method by matching 2D image and 3D point cloud targets. We demonstrate that it outperforms various baselines, such as vanilla retrieval [26], or deformation-aware retrieval using direct optimization for deformation [41], or a fixed, pre-trained, neural deformation module (*i.e.* omitting joint training). We also show that our method can be used even with imperfect and inconsistent segmentations produced automatically. Finally, we show that even with a different deformation module (e.g., Neural Cages [49]), our joint training leads to better results.

2. Related Works

Deep Learning for Shape Generation. Many neural techniques have been proposed recently for learning generative latent representations for 3D shapes, modeling geometry as implicit functions [31, 27, 5], atlases [13], volumetric grids [8, 45], point clouds [1, 48, 9], and meshes [42, 44]. These models tend to under-perform on topologically complex objects with intricate part structures. Thus, other techniques focus on factorized representation, where variations in structure are modeled separately from the geometry [25, 11, 28]. These generative techniques are commonly used jointly with 2D CNNs [33] or shape encoders [51] to enable creating a shape based on some partial observations, such as a natural image [12] or a point scan [7]. A simple shape retrieval [26] could also be viewed as the simplest version of such a shape generator, where the system simply

returns the nearest neighbor in the latent space, in fact, offering a strong baseline to other generative techniques [38].

Deformation-Aware Retrieval. Direct retrieval has the advantages of producing stock-quality meshes [39, 40], however, unless the database contains all possible objects, might not produce a good fit for an encoded target. Prior works [30, 34, 41] address this issue by additionally deforming, *i.e.* fitting, the retrieved shape to the desired target. One approach is to exhaustively deform all shapes in the database to the target and select the best fit [30], but is however computationally expensive. Schulz *et al.* [34] alleviates this by retrieving parametric models by representing each as a set of points and bounded tangent planes, thus enabling retrieval before the fitting process. Leveraging on deep networks, Uy *et al.* [41] use a deep embedding to retrieve a shape and then separately deform it to the target by directly optimizing the ARAP [18] loss. Their method is limited to full shapes as targets as direct optimization is not possible with partial scans [2] or natural images. They further observe that the retrieval network needs to be aware of the deformation step to retrieve a more appropriate source. We extend their approach in several ways. First, we demonstrate that one can use retrieve-and-deform method with a neural deformation technique, allowing us to handle natural images as inputs. Second, we propose a novel joint training process, which enables us to train our deformation module to be more suitable for the kind of pairs of shapes that are being retrieved. And third, we propose a novel neural deformation module that is especially suitable for heterogeneous shape collections with topological and structural variations.

3D Deformation. Deforming a source 3D model to a target is one of the fundamental problems in geometry processing. If target is a full shape, direct optimization techniques can be employed [17, 35, 22], as well as human-made [23, 10, 46, 52] shapes. One can only directly optimize if a target is a full shape, however if it is of a different modality, such as image or partial scan, one needs to employ priors [15]. Neural techniques have been used to learn such deformation priors from collections of shapes, representing deformations as volumetric warps [20, 24, 50], cage deformations [49], vertex-based offsets [43, 14] or flow-based approaches [21]. To make learning easier, these techniques typically assume homogeneity in the sources and represent the deformation with the same number of parameters for each source, *i.e.* grid control points [20], cage mesh [49] or number of vertices [43]. These assumptions make them less suitable for heterogeneous databases of sources with significant structural variations at the part level. We extend the part-level reasoning that proved to be effective for other problems [29, 47, 4] to neural deformation, by proposing a novel module that can learn source-specific deformations, and handle cases when sources can have different number of deformation parameters to account for part variability.

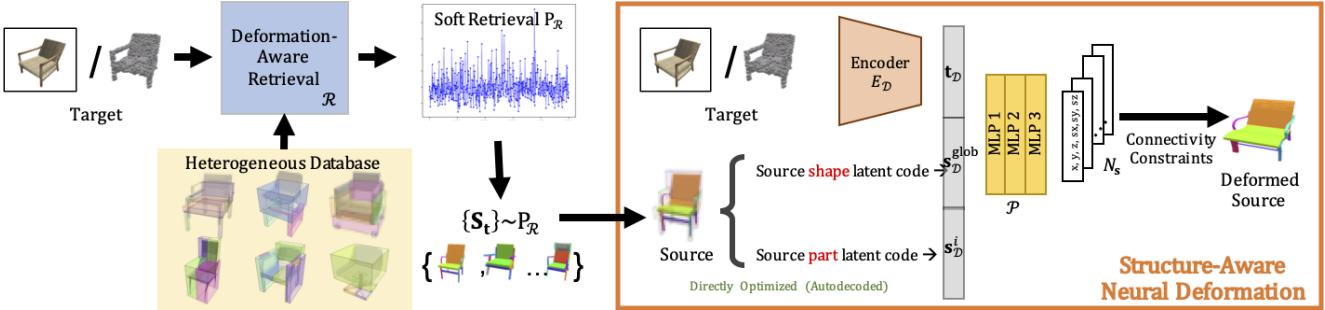


Figure 2. During training, given a target image or a point cloud and a database of deformable sources, we retrieve a subset of source models based on their proximity in the retrieval space, and use the structure-aware deformation module (right) to fit each source. Our deformation module uses encoded target, global and per-part source codes to predict per-part deformation parameters.

3. Method

Overview. We assume to possess a database of parametric *source* models $s \in S$, and we aim to jointly train a deformation and retrieval module to choose a source and deform it to fit a given target t (an image or a point cloud), with respect to a fitting metric \mathcal{L}_{fit} (we use chamfer in all experiments). Each source also has parameters defining its individual deformation space, that are optimized during training.

Our deformation module is designed to enable a different deformation function \mathcal{D}_s for each source s , based on its parts. The retrieval module uses embeddings of the sources and the target into a latent space \mathcal{R} , where it retrieves based on a distance measure $d_{\mathcal{R}}$, which enables the retrieval of the source shape that best fits to the target *after* deformation.

The training consists of optimizing the latent retrieval space \mathcal{R} and the deformation functions $\{\mathcal{D}_s\}$:

$$\min \mathcal{L}_{\text{fit}}(\mathcal{D}_{s'}(t), t_{\text{true}}),$$

where s' is the closest source to target t in latent space, w.r.t the distance measure $d_{\mathcal{R}}(s', t)$, and t_{true} is the corresponding true shape.

We progress by first explaining in Section 3.1 how we design our framework and optimization in a way that is differentiable and enables the deformation and retrieval modules to propagate information from one to the other. We then detail our novel deformation module and how it enables source-specific deformations in Section 3.2, and conclude by describing the retrieval module in Section 3.3.

3.1. Joint Deformation and Retrieval Training

It is critical for our approach to optimize the parameters of \mathcal{R} and $\{\mathcal{D}_s\}$ jointly. First, it enables the deformation module of each source to efficiently utilize its capacity and specialize on relevant targets that it could fit to. Second, it allows the retrieval module to create a deformation-aware latent space where sources are embedded closer to the targets they can deform to.

Soft Retrieval for Training. The retrieval module embeds the sources and the target in the latent retrieval space

\mathcal{R} . The proximity in latent space is used to define a biased distribution that can be loosely interpreted as the probability of source s being deformable to t :

$$P_{\mathcal{R}}(s, t) = p(s; t, S, d_{\mathcal{R}}, \sigma_0), \quad (1)$$

where

$$p(s; t, \tilde{S}, \tilde{d}, \tilde{\sigma}) = \frac{\exp(-\tilde{d}^2(s, t)/\tilde{\sigma}^2(s))}{\sum_{s' \in \tilde{S}} \exp(-\tilde{d}^2(s', t)/\tilde{\sigma}^2(s))},$$

$\tilde{d} : (S \times T) \rightarrow \mathbb{R}$ is a distance function between a source and a target (T is the target space), and $\tilde{\sigma} : S \rightarrow \mathbb{R}$ is a potentially source-dependent scalar function. Though, $\sigma_0(\cdot) = 100$ is a constant set for all experiments.

Instead of choosing the highest-scoring source according to the probability $P_{\mathcal{R}}$, we perform soft retrieval and sample $K = 10$ retrieval candidate sources from the distribution:

$$s_i \sim P_{\mathcal{R}}(s, t), \forall i \in \{1, 2, \dots, K\}.$$

The candidates $S_t = \{s_1, \dots, s_K\}$ sampled via our soft retrieval are then used to train both our retrieval module to learn \mathcal{R} and deformation module for source-dependent deformation functions $\{\mathcal{D}_s\}$.

The soft retrieval is crucial for our training: 1) adding randomness to the retrieval ensures that the latent space is optimized with respect to both high-probability instances and low-probability ones, that may reverse roles as the deformation module improves. 2) On the other hand, training the deformation module with a bias towards high-probability sources and not random ones ensures it is aware of the retrieval module and expands its capacity on meaningful matches.

Training. We train the two modules jointly in an alternating fashion, keeping one module fixed when optimizing the other, and vice versa, in successive iterations. To train the retrieval module, we deform the candidate sources and compute their fitting losses to the target. We update our latent space \mathcal{R} by penalizing the discrepancy between the distances in the retrieval space $d_{\mathcal{R}}$ and the post-deformation fitting losses \mathcal{L}_{fit} using softer probability measures estimated from the distances of the sampled candidates:

$$\mathcal{L}_{\text{emb}} = \sum_{k=1}^K |\mathbf{p}(\mathbf{s}_k, \mathbf{t}, \mathbf{S}_t, d_{\mathcal{R}}, \sigma_0) - \mathbf{p}(\mathbf{s}_k, \mathbf{t}, \mathbf{S}_t, d_{\text{fit}}, \sigma_k)|, \quad (2)$$

where

$$d_{\text{fit}}(\mathbf{s}, \mathbf{t}) = \mathcal{L}_{\text{fit}}(\mathcal{D}_{\mathbf{s}}(\mathbf{t}), \mathbf{t}_{\text{true}}), \quad (3)$$

and σ_k is a source-dependent scalar representing the predicted range of variations of each source model $\mathbf{s} \in \mathbf{S}$, which is also learned. For the deformation module, we update the deformation functions $\{\mathcal{D}_{\mathbf{s}_k}\}$ for the K biased samples by minimizing the post-deformation fitting losses weighted by their soft probability measures:

$$\mathcal{L}_{\text{def}} = \sum_{k=1}^K \mathbf{p}(\mathbf{s}_k, \mathbf{t}, \mathbf{S}_t, d_{\mathcal{R}}, \sigma_0) \mathcal{L}_{\text{fit}}(\mathcal{D}_{\mathbf{s}_k}(\mathbf{t}), \mathbf{t}_{\text{true}}). \quad (4)$$

This weighting scheme puts greater weight on sources that are closer to the target in the embedding space, thus further making the deformation module aware of the retrieval module, and allowing it to specialize on more amenable sources with respect to the training target.

Inner Deformation Optimization. To enforce the deformation module to perform more significant deformations, at each training iteration we use the deformation network’s current output for the given source and target that consists of parameters for the deformation, and directly run SGD on the deformation parameters until convergence of the fitting loss. We then measure the least-square error between the deformation network’s output and the optimized parameters, and train the module by backpropagating this error, hence enabling the network to learn stronger deformations and getting a better estimate for how well the source could be aligned to the target after the deformation. See the supplementary for the full details.

3.2. Structure-Aware Neural Deformation

While our joint training approach described in Section 3.1 is generic and can work well with different parameterization of deformations, its greatest advantage is that it enables our deformation space to vary greatly between each source without having the deformation module learn subpar deformations. We thus devise a deformation module with a heterogeneous space of part-based deformations as shown in Figure 1, which vary per each source, a necessary feature if one wants to tailor the deformations to be restricted to preserve and adjust part structures.

To get meaningful parts, we use manual segmentations from PartNet [29] or automatic segmentations (preprocessing) of ComplementMe [37], produced by grouping connected components in raw meshes. Our deformation module predicts a simple deformation consisting of translation and axis-aligned scaling for each part in a source model. See supplementary for the details on the prediction. The

number of parts for different sources vary, making the deformation functions source-dependent $\{\mathcal{D}_{\mathbf{s}}\}$. We abuse the notation a bit and let \mathcal{D} denote our deformation module.

We propose to use a neural network which can be applied to each part separately, thus making it applicable to models with varying part-constellations, as opposed to previous methods. Namely, we assign to each source a global code $\mathbf{s}_{\mathcal{D}}^{\text{glob}} \in \mathbb{R}^{n_1}$, and for each part within the shape, we assign a local code $\mathbf{s}_{\mathcal{D}}^{i=1 \dots N_s} \in \mathbb{R}^{n_2}$. The target is encoded via an encoder (PointNet [32] for point clouds and ResNet [16] for images) into a latent vector $\mathbf{t}_{\mathcal{D}} = E_{\mathcal{D}}(\mathbf{t}) \in \mathbb{R}^{n_3}$. We set $n_1 = n_3 = 256$ and $n_2 = 32$ for all experiments. The global, local, and target codes are concatenated and fed to a lightweight 3-layer MLP (512, 256, 6), \mathcal{P} , which outputs the deformation parameters of the corresponding part. The deformation parameters of all parts are then used to obtain the final deformed source shape. Each source’s global and local codes are optimized in an auto-decoder fashion during the training of the deformation module. Figure 2 (right) illustrates our module. We additionally add a symmetry loss in training our deformation module to enforce bilateral symmetry of the output deformed shapes as regularization, more details are found in the supplementary.

Connectivity Constraints. We further take advantage of our joint-training’s robustness to heterogeneous deformation spaces, and add part-connectivity constraints. We achieve this by introducing a layer that receives a deformation and projects it onto the space of contact-preserving deformations, via a simple linear transformation. Contacts are defined between pairs of connected parts where each pair introduces a set of constraints. The source models have different sets of connected parts, and hence a different number and set of constraints, as illustrated in Figure 1, making the deformation functions $\{\mathcal{D}_{\mathbf{s}}\}$ even more source-dependent. More details are found in the supplementary.

3.3. Retrieval in Latent Space

The retrieval space \mathcal{R} is defined similarly to Uy et al. [41], and we provide relevant technical details in this section for completeness. We use a PointNet or ResNet encoder to get the latent code of the target: $\mathbf{t}_{\mathcal{R}} = E_{\mathcal{R}}(\mathbf{t}) \in \mathbb{R}^{n_4}$ with $n_4 = 256$. The sources are represented as regions in the latent space, defined by a center code $\mathbf{s}_{\mathcal{R}} \in \mathbb{R}^{n_4}$ and a variance matrix $\mathbf{s}_{\mathcal{R}}^v \in \mathbb{R}^{n_4 \times n_4}$ that defines the egocentric distance field. The variance matrix is diagonal positive definite, with the positivity enforced by the sigmoid activation function. We define the distances in the retrieval space as:

$$d(\mathbf{s}, \mathbf{t})_{\mathcal{R}} = \sqrt{(\mathbf{s}_{\mathcal{R}} - \mathbf{t}_{\mathcal{R}})^T \mathbf{s}_{\mathcal{R}}^v (\mathbf{s}_{\mathcal{R}} - \mathbf{t}_{\mathcal{R}})}. \quad (5)$$

During training we optimize the parameters of the encoder $E_{\mathcal{R}}(\mathbf{t})$ as well as latent codes and variances for each



Figure 3. We test our trained method on online product images.

source, \mathbf{s}_R , \mathbf{s}_R^v . \mathbf{s}_R is obtained by feeding the default shape of source model \mathbf{s} to encoder $E_R(\mathbf{t})$. Different from Uy et al. [41], we optimize \mathbf{s}_R^v in an auto-decoder fashion, since we want to represent the deformation space of the source rather than its geometry. This allows us to handle sources with similar geometry but different parameterizations.

4. Results

In this section we discuss our data sources and evaluation metric and provide thorough experiments with image (Sec 4.1) and point cloud (Sec 4.2) targets.

Datasets and Evaluation Metric. We evaluate our method on the three furniture categories in the ShapeNet dataset [3] chairs (6531), tables (7939) and cabinets (1278). For our database of deformable source models, we use manually- and automatically-segmented shapes from two different datasets. Manually-segmented shapes come from the finest level of PartNet hierarchy [29], and we select random 10% of the data as our sources. Automatically-segmented shapes come from two pre-analyzed classes in ComplementMe [37] (chairs and tables), and we pick 200 random models for each. We remove the selected sources from the database, and use remaining models as training (80%) and testing (20%) targets. To demonstrate the practical utility of our method, we also test our trained networks on product images and 3D scans.

We represent the shapes by uniformly sampling 2048 points. For the image experiments, we render 24 uniformly-sampled viewpoints, and pick a random view at each iteration during training. In all cases our true targets and deformed sources are represented as point clouds, and point-to-points distances are used for training and evaluation.

4.1. Image-to-Mesh

We first test our system on product images “in the wild” as well as images from our test set and show qualitative results for retrieval and deformation in Figures 3 and 4. Note

	Chair	Table	Cabinet
R	1.926	2.235	2.228
R+DF	1.969	2.705	2.035
DAR (Retrieval Only)	1.345	2.058	3.489
DAR+DF	1.216	1.621	1.333
Uniform Sampling	1.118	1.486	1.318
Ours	1.005	0.970	1.220
Ours w/ IDO	0.976	0.935	1.141

Table 1. Comparing our method to various baselines and ablations on image-to-mesh benchmark (chamfer distances, $\times 10^{-2}$).

how retrieved results have compatible structure to the input, which then enables the deformation technique to match the source to the target. We quantitatively evaluate performance of our method and report chamfer distances in Table 1 (**Ours**) together with the chamfer distances with the inner deformation optimization (**Ours w/ IDO**). Since IDO step described significantly increases training time, we do not use it in ablations and comparisons.

Retrieval Baselines. We compare our method to a vanilla image-to-shape retrieval technique [26] (denoted by **R**). This baseline first constructs the latent space by projecting shape-to-shape chamfer distance matrix to 256-dimensional space via MDS, and then trains a ResNet [16] encoder to map images to that latent space with $L2$ -loss. Since any retrieval baseline can also work with a pre-trained neural deformation, we also train our structure-aware deformation module on random pairs of shapes (i.e., ablating the joint training procedure) and report results with neural deformation applied to the retrieved results (**R+DF**). Since this vanilla baseline retrieves only based on geometric similarity and does not account for deformation, the retrieved shapes may not deform to targets well. Hence, there is no improvement when deforming with the pre-trained deformation function.

The second retrieval baseline is the deformation-aware retrieval [41], where we also use our structure-aware deformation module pre-trained on random pairs. For this baseline we report results for retrieval (**DAR**) as well as deformation (**DAR+DF**). Our results show that being deformation-aware is not sufficient, and it is important for deformation module to be trained with retrieved shapes.

Biased Sampling Ablation. Our joint training benefits from biasing sampling of retrieval targets (Eq. 1). To ablate this, we sample from a uniform distribution, i.e., each source is sampled with equal probability during training. In this setting, while the retrieval and deformation modules are still trained together, they are less aware of which samples are most relevant at inference time and thus yield higher errors (see **Uniform Sampling** in Table 1).

Improvement in Deformation Module. In addition to holistic improvement to the final output, we would like to

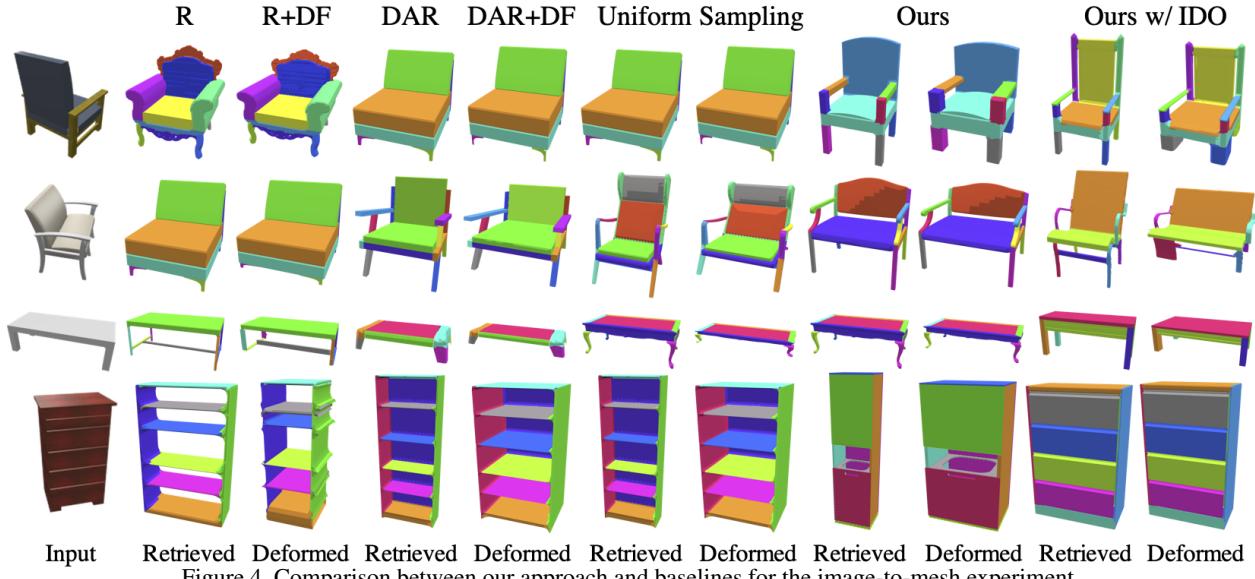


Figure 4. Comparison between our approach and baselines for the image-to-mesh experiment.

	Chair	Table	Cabinet
DF	0.748	0.702	0.706
Uniform Sampling	0.755	0.690	0.701
Ours	0.681	0.584	0.675
Ours w/ IDO	0.669	0.533	0.689

Table 2. Improvement in deformation module for image-to-mesh task with oracle retrieval due to joint training (chamfer $\times 10^{-2}$).

evaluate the effect of joint training on deformation module. To do this, we use *oracle retrieval* where for each test target, we deform all sources and pick the one with the smallest fitting error. Our joint training allows the deformation module to specialize on targets that are a good fit. Thus, as shown in Table 2, our method achieves the lowest fitting error for the best-fit sources with respect to the deformation module trained on all pairs (**DF**), and the deformation module trained without the biased sampling (**Uniform Sampling**).

4.2. Points-to-Mesh

We also test our method on point cloud targets. We first show qualitative results with *real* noisy and partial 3D scans in Scan2CAD dataset [2]. Figure 5 show some examples, and more are in the supplementary. As shown, given an incomplete scan, with missing parts and a noise, our approach still correctly retrieves and deforms a source database model to output a clean and complete mesh to match the scan. Our structure-aware neural deformation leverages learned shape priors to complete missing regions.

We also provide qualitative and quantitative results on our test set of point clouds sampled from ShapeNet meshes in Table 3 and Figure 6. As in the previous section, we report our results (**Ours**) along with our method with the inner direct optimization step (**Ours w/ IDO**). Since our input are point clouds, similar to prior work [41] we can

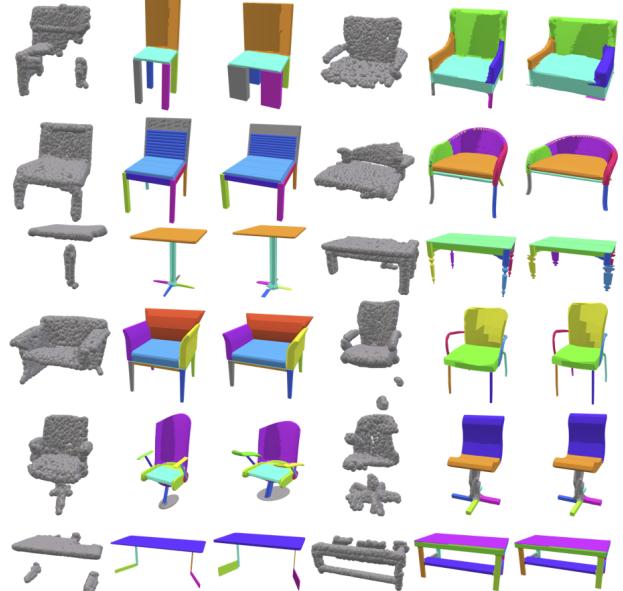


Figure 5. Our approach tested on real scans from the Scan2CAD dataset [2].

also directly optimize the chamfer distance to make our output fit better to the inputs, and we report results with this post-process as well (**Ours + DO**, **Ours w/ IDO + DO**).

Deformation-Aware Retrieval Baseline. We compare to deformation-aware retrieval [41] (**DAR**) followed by either directly optimizing with respect to our per-part parameters (**DAR+DO**), or using our neural deformation module pre-trained on random pairs (**DAR+DF**). Note that the direct optimization step is only possible with complete inputs and cannot be employed with partial data such as

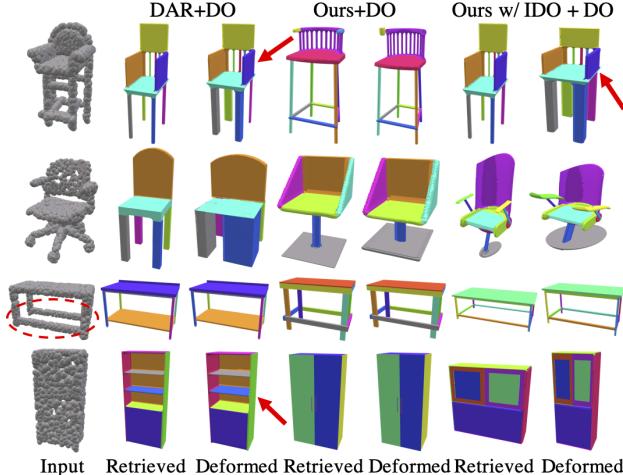


Figure 6. Comparison between our approach and baselines for the point-cloud-to-mesh experiment.

	Chair	Table	Cabinet
Classif.+DO	1.826	2.192	1.144
DAR+DO	0.584	0.452	0.633
Ours+DO	0.504	0.414	0.494
Ours w/ IDO+DO	0.484	0.407	0.485
Classif.+DO	3.199	4.518	1.661
DAR+DF	0.965	1.561	0.829
Uniform Sampling	0.998	1.502	0.767
Ours	0.763	0.696	0.715
Ours w/ IDO	0.691	0.670	0.696

Table 3. Comparing our method to various baselines and ablations on points-to-mesh benchmark (chamfer distances, $\times 10^{-2}$).

3D scans or images. Our method outperforms this baseline with and without the direct optimization step (Table 3). Qualitative results in Figure 6, also demonstrate that our method retrieves structurally similar shapes and deforms them to a better fit for the target. Even if retrieved shape is identical (chair in the first row), the deformation learned with our method is superior (e.g., see seat alignment).

Template-Classification Baseline. We also compare to a template-classification-based approach mimicked from [10] (**Classif**). Instead of using a non-learnable deformation module via direct optimization of handcrafted templates as in [10], we use our pre-trained neural deformation module (DF) to make the baseline computationally feasible. We treat every source shape as a template, deform it to each training target, and train a classifier based on the best match. We use this classifier instead of the retrieval module at inference time, and show the fitting error in Table 3. Note that this baseline is worse than our method and even [41].

Biased Sampling Ablation. As in the image target case, we demonstrate the importance of biased sampling (Equation 3.1) in joint training (Table 3, **Uniform Sampling**).

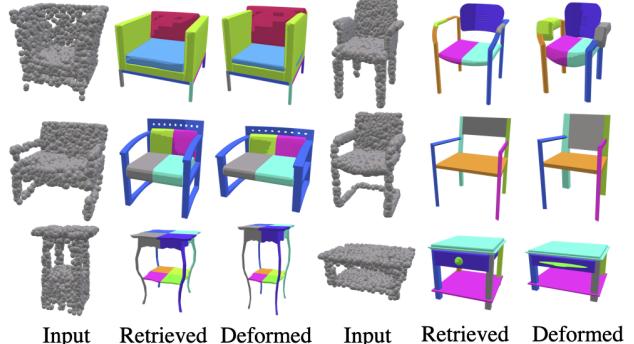


Figure 7. Fitting results using auto-segmented sources [37].

	DAR+DF	Uniform Sampling	Ours
Chair	1.118	1.077	0.990
Table	1.409	1.502	1.166

Table 4. Using auto-segmented models as the source database [37] (chamfer distance ($\times 10^{-2}$)).

	Chair	Table	Cabinet
DAR+NC	0.480	0.575	0.589
Ours NC	0.476	0.411	0.538

Table 5. Using our joint training with Neural Cages [49] deformation module (chamfer distances, $\times 10^{-2}$).

Performance on Auto-Segmented Data. Since manually segmenting a collection of source shapes is an expensive process, we test our method on automatically-segmented models. We use a heuristic method proposed in ComplementMe [37] grouping connected components of meshes. As shown in Figure 7, even though the models have inconsistent segmentations, our method can still successfully learn a meaningful deformation module. We also outperform the baseline (**DAR+DF**, **Uniform Sampling**) in the quantitative benchmark (Table 4).

Performance with Neural Cages [49]. Since our joint training is not restricted to our structure-aware deformation, we further evaluate the performance of our framework with an alternative neural deformation method. We pick Neural Cages [49], a state-of-the-art technique that parameterizes global warping as a cage-based deformation. We simply replace our structure-aware deformation with Neural Cages, without any other changes to our joint training process (**Ours NC**). We further compare to the baseline of running deformation-aware retrieval [41] with neural cage module that is pre-trained on random pairs (**DAR+NC**). Joint training offers an improvement with respect to our benchmark on all categories of shapes (see Table 5). Qualitative results in Figure 8 show that our joint training scheme can better retrieve shapes such as chairs with the right back and seat shape (first two rows), and a cabinet with shelves.

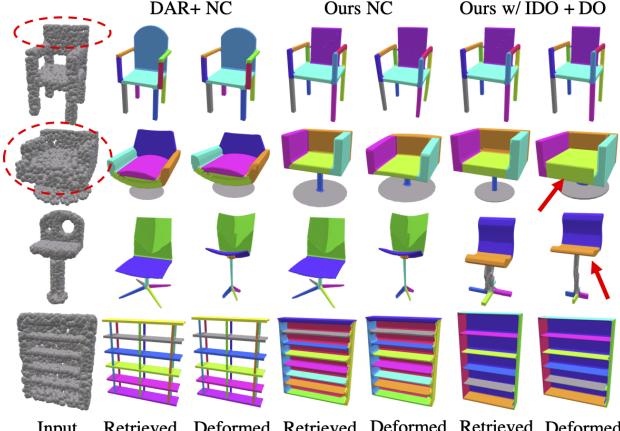


Figure 8. Using Neural Cages [49] as a deformation module in our joint training.

	Chair	Table	Cabinet
DF	0.712	0.703	0.549
Uniform Sampling	0.714	0.700	0.509
Ours	0.643	0.564	0.494
Ours w/ IDO	0.583	0.482	0.494

Table 6. Improvement in deformation module for points-to-mesh (with oracle retrieval) due to joint training (chamfer distances, $\times 10^{-2}$).

We remark that our joint training does not constrain the choice of the neural deformation module. One can choose any module based on its strengths and weaknesses. For instance, Neural Cages module often provides a tighter fit to the target, although it often results in bending/distortion of shapes (e.g., legs of the chair in the first row and the seat and legs of the chair in the third row of Figure 8). It also lacks the ability to change the geometry of individual local parts. In contrast, our deformation module allows thickening parts such as the seat of the chair in the second row of Figure 8. This implies that Neural Cages can be used when a tighter fit to the target is prioritized while our method can be used when it is more desired to preserve and manipulate part-level structure of the object. Our method is also more suitable for heterogeneous sources whose deformations need to be parameterized in different manners.

Improvement in Deformation Module. As in the image target case, we demonstrate the improvement in the deformation module alone using oracle retrieval with joint training (**Ours**), random pairs (**DF**), and without biased sampling (**Uniform Sampling**), see Table 6. We demonstrate a qualitative example in Figure 9 showing an example where all methods retrieve the same source model for the given target, but our joint approach achieves the best output as shown by the differences in the legs of the chair.

Performance for Different Database Sizes. We further evaluate the performance of different techniques while

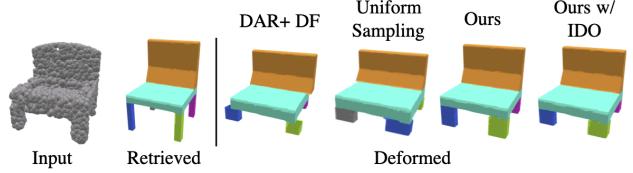


Figure 9. We pick an example where retrieved mesh is the same for all methods, and show that joint training also improves the quality of the neural deformation module on its own.

	DAR+DF	Uniform Sampling	Ours
$ S =50$	0.872	0.877	0.823
$ S =100$	0.858	0.860	0.803
$ S =200$	0.850	0.841	0.748
$ S =400$	0.938	0.985	0.784
$ S =800$	1.142	1.541	0.734

Table 7. Performance on of our method and various baselines with different source database sizes (chamfer distances, $\times 10^{-2}$).

varying the size of the database of source models. We randomly sample 50, 100, 200, 400, and 800 chair models from PartNet to construct the source databases. Table 7 shows that in all cases our joint training approach improves the performance over the baselines. The boost in the performance of our joint training is bigger in larger databases as there are combinatorially more random source-target pairs which may not be deformable.

5. Conclusion

To summarize, we propose a joint training for retrieval-and-deformation problem, where the neural modules inform one another, yielding better matching results with respect to image and point cloud targets. Our joint training procedure offers improvements regardless of the choice of the neural deformation module. We further propose a novel structure-aware deformation module that is especially suitable for heterogeneous datasets of source models with very diverse parameterizations of deformations. Our method does not require consistent manual segmentations or part labels and can work with imprecise automatic segmentations.

Limitations and Future Work. Our method is only supervised by chamfer distance, and thus might not favor semantic and structural similarity between the target and retrieved sources. We believe that improving the loss function to leverage manual part annotations can further remedy this issue. Our deformation module does not provide strong links between parts, and does not favor capturing part-to-part relations, which can be addressed by adding more constraints (e.g. symmetry) as well as improving our learning module with a more advanced graph-based neural architecture.

Acknowledgements. This work is supported by a grant from the Samsung GRO program, a Vannevar Bush Faculty Fellowship, and gifts from Adobe, Autodesk, and Snap.

References

- [1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J Guibas. Learning representations and generative models for 3D point clouds. In *ICML*, 2018. [2](#)
- [2] Armen Avetisyan, Manuel Dahner, Angela Dai, Manolis Savva, Angel X Chang, and Matthias Nießner. Scan2CAD: Learning cad model alignment in RGB-D scans. In *CVPR*, 2019. [2](#), [6](#), [13](#), [15](#)
- [3] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An information-rich 3D model repository, 2015. [5](#)
- [4] Siddhartha Chaudhuri, Daniel Ritchie, Jiajun Wu, Kai Xu, and Hao Zhang. Learning to generate 3D structure. *Eurographics State-of-the-Art Reports (STAR)*, 2020. [2](#)
- [5] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *CVPR*, 2019. [2](#)
- [6] Manuel Dahner, Angela Dai, Leonidas Guibas, and Matthias Nießner. Joint embedding of 3d scan and cad objects. In *ICCV*, 2019. [1](#)
- [7] Angela Dai and M. Nießner. Scan2Mesh: From unstructured range scans to 3d meshes. In *CVPR*, 2019. [2](#)
- [8] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3D-encoder-predictor CNNs and shape synthesis. In *CVPR*, 2017. [2](#)
- [9] H. Fan, H. Su, and L. Guibas. A point set generation network for 3D object reconstruction from a single image. In *CVPR*, 2017. [2](#)
- [10] V. Ganapathi-Subramanian, O. Diamanti, S. Pirk, Chengcheng Tang, M. Nießner, and L. Guibas. Parsing geometry using structure-aware shape templates. In *3DV*, 2018. [2](#), [7](#), [14](#)
- [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. In *SIGGRAPH Asia*, 2019. [2](#)
- [12] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh R-CNN. In *ICCV*, 2019. [2](#)
- [13] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *CVPR*, 2018. [2](#)
- [14] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. Deep self-Supervised cycle-Consistent deformation for few-shot shape segmentation. In *Eurographics Symposium on Geometry Processing*, 2019. [1](#), [2](#)
- [15] Rana Hanocka, Noa Fish, Zhenhua Wang, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. ALIGNet: Partial-Shape agnostic alignment via unsupervised learning. *ACM Transactions on Graphics*, 2018. [2](#)
- [16] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. [4](#), [5](#)
- [17] Qixing Huang, B. Adams, Martin Wicke, and L. Guibas. Non-rigid registration under isometric deformations. *Computer Graphics Forum*, 2008. [2](#)
- [18] Takeo Igarashi, Tomer Moscovich, and John F. Hughes. As-rigid-as-possible shape manipulation. In *SIGGRAPH*, 2005. [2](#)
- [19] Vladislav Ishmitsev, Alexey Bokhovkin, Alexey Artemov, Savva Ignat'yev, Matthias Nießner, Denis Zorin, and Burnaev Evgeny. CAD-Deform: Deformable fitting of cad models to 3d scans. In *ECCV*, 2020. [1](#)
- [20] Dominic Jack, Jhony K. Pontes, Sridha Sridharan, Clinton Fookes, Sareh Shirazi, Frederic Maire, and Anders Eriksson. Learning free-Form deformations for 3D object reconstruction. In *ICCV*, 2018. [2](#)
- [21] Chiyu Jiang, Jingwei Huang, Andrea Tagliasacchi, and Leonidas Guibas. ShapeFlow: Learnable deformations among 3D shapes. In *NeurIPS*, 2020. [2](#)
- [22] Tao Ju, Scott Schaefer, and Joe Warren. Mean value coordinates for closed triangular meshes. In *SIGGRAPH*, 2005. [2](#)
- [23] Vladimir G. Kim, Wilmot Li, Niloy J. Mitra, Siddhartha Chaudhuri, Stephen DiVerdi, and Thomas Funkhouser. Learning part-based templates from large collections of 3D shapes. In *SIGGRAPH*, 2013. [2](#), [14](#)
- [24] Andrey Kurenkov, Jingwei Ji, Animesh Garg, Viraj Mehta, JunYoung Gwak, Christopher Bongsoo Choy, and Silvio Savarese. DeformNet: Free-Form deformation network for 3d shape reconstruction from a single image. In *WACV*, 2018. [2](#)
- [25] Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. Grass: Generative recursive autoencoders for shape structures. In *SIGGRAPH*, 2017. [2](#)
- [26] Yangyan Li, Hao Su, Charles Ruizhongtai Qi, Noa Fish, Daniel Cohen-Or, and Leonidas J. Guibas. Joint embeddings of shapes and images via cnn image purification. In *SIGGRAPH Asia*, 2015. [1](#), [2](#), [5](#)
- [27] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019. [2](#)
- [28] Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas Guibas. StructureNet: Hierarchical graph networks for 3D shape generation. In *SIGGRAPH Asia*, 2019. [2](#)
- [29] Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In *CVPR*, 2019. [2](#), [4](#), [5](#), [13](#), [15](#)
- [30] Liangliang Nan, Ke Xie, and Andrei Sharf. A search-classify approach for cluttered indoor scene understanding. *ACM Transactions on Graphics*, 2012. [2](#)
- [31] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. [2](#)
- [32] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *CVPR*, 2017. [4](#)

- [33] Charles Ruizhongtai Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas Guibas. Volumetric and multi-view CNNs for object classification on 3D data. In *CVPR*, 2016. 2
- [34] Adriana Schulz, Ariel Shamir, Ilya Baran, David I. W. Levin, Pitchaya Sitti-Amorn, and Wojciech Matusik. Retrieval on parametric shape collections. *ACM Transactions on Graphics*, 2017. 1, 2
- [35] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Eurographics Symposium on Geometry Processing*, 2007. 2
- [36] Minhyuk Sung, Zhenyu Jiang, Panos Achlioptas, Niloy J. Mitra, and Leonidas J. Guibas. DeformSyncNet: Deformation transfer via synchronized shape deformation spaces. In *SIGGRAPH Asia*, 2020. 1
- [37] Minhyuk Sung, Hao Su, Vladimir G. Kim, Siddhartha Chaudhuri, and Leonidas Guibas. ComplementMe: Weakly-supervised component suggestions for 3D modeling. In *SIGGRAPH Asia*, 2017. 4, 5, 7, 13, 15
- [38] Maxim Tatarchenko*, Stephan R. Richter*, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. What do single-view 3d reconstruction networks learn? In *CVPR*, 2019. 1, 2
- [39] Trimble. 3D warehouse. 2
- [40] TurboSquid. TurboSquid. 2
- [41] Mikaela Angelina Uy, Jingwei Huang, Minhyuk Sung, Tolga Birdal, and Leonidas Guibas. Deformation-Aware 3D model embedding and retrieval. In *ECCV*, 2020. 1, 2, 4, 5, 6, 7, 12, 14
- [42] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3D mesh models from single rgb images. In *ECCV*, 2018. 2
- [43] Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. 3DN: 3D deformation network. In *CVPR*, 2019. 1, 2
- [44] Chao Wen, Yinda Zhang, Zhuwen Li, and Yanwei Fu. Pixel2Mesh++: Multi-view 3D mesh generation via deformation. In *ICCV*, 2019. 2
- [45] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In *NeurIPS*, 2016. 2
- [46] Kai Xu, Honghua Li, Hao Zhang, Daniel Cohen-or, Yueshan Xiong, and Zhi-Quan Cheng. Style-content separation by anisotropic part scales. In *SIGGRAPH Asia*, 2010. 2
- [47] Kai Xu, Hanlin Zheng, Hao Zhang, Daniel Cohen-Or, Ligang Liu, and Yueshan Xiong. Photo-inspired model-driven 3d object modeling. In *SIGGRAPH*, 2011. 2
- [48] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. PointFlow: 3D point cloud generation with continuous normalizing flows. In *ICCV*, 2019. 2
- [49] Wang Yifan, Noam Aigerman, Vladimir Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. Neural cages for detail-preserving 3D deformations. In *CVPR*, 2020. 1, 2, 7, 8, 13, 14
- [50] Ersin Yumer and Niloy J. Mitra. Learning semantic deformation flows with 3d convolutional networks. In *ECCV*, 2016. 2
- [51] A. Khosla F. Yu L. Zhang X. Tang J. Xiao Z. Wu, S. Song. 3D ShapeNets: A deep representation for volumetric shapes. In *CVPR*, 2015. 2
- [52] Youyi Zheng, Hongbo Fu, Daniel Cohen-Or, Oscar Kin-Chung Au, and Chiew-Lan Tai. Component-wise Controllers for Structure-Preserving Shape Manipulation. *Computer Graphics Forum*, 2011. 2

Appendix

We provide additional implementation details (Section A.1), and additional quantitative evaluations (Section A.2.1) and qualitative results (Section A.2.2).

A.1. Implementation Details

Inner Deformation Optimization. We provide additional details for the inner deformation optimization step, as described in Section 3.1 of the main paper.

We initialize the inner deformation optimization with the parameters predicted by our deformation network. We propagate gradients directly to the parameters by minimizing the mean chamfer loss of the batch. We use the SGD optimizer with a learning rate of 0.05, and we terminate upon convergence (i.e., when the maximum loss change in a pair in the batch is less than 10^{-6} or it has reached the maximum number of iterations = 2000).

Structure-Aware Neural Deformation. We provide additional details for our structure-aware neural deformation as described in Section 3.2 of the main paper.

Our structure-aware neural deformation module predicts the deformation parameter offset from the default parameters of each source model. Specifically for a specific source-target pair, given network prediction p and default source parameter \bar{p} , our output parameters to obtain the deformed source model is given by $(\bar{p} + \alpha * p$, where $\alpha = 0.1)$ in all our experiments.

We also add the symmetry loss to supervise the training of our structure-aware neural deformation. Note that all the source shapes in our databases have global reflective symmetry, and have been pre-aligned so that yz -plane aligns with the symmetry axis. Given the output deformed source shape, represented as a sampled point cloud O , for target point cloud T of given target t , we reflect each point O about the yz -plane to obtain reflected point cloud O' , then the symmetry loss is given by

$$\mathcal{L}_{\text{symm}} = \mathcal{L}_{\text{CD}}(O, O'),$$

where \mathcal{L}_{CD} is the chamfer distance. Then the loss we use to train our deformation module is given by

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{def}} + \mathcal{L}_{\text{symm}},$$

where \mathcal{L}_{def} is defined in Equation 4 in the main paper.

Connectivity constraint. We provide the details on how we obtain our connectivity constraint as described in Section 3.2 of the main paper.

We precompute the constraint projection matrix for each source $s \in S$ in an automatic pre-processing step, where we first identify contacts based on the distance between the closest pairs of keypoints between pairs of

	Chair	Table	Cabinet
DAR+DF (No Conn.)	1.107	1.728	1.480
Uniform Sampling (No Conn.)	1.129	1.655	1.358
Ours (No Conn.)	0.757	0.708	0.846

Table A1. Our approach compared to the baselines in the setup with no connectivity constraint.

parts (s_D^i, s_D^j). Parts s_D^i and s_D^j are deemed connected if the closest part of keypoints falls below a threshold $\tau = 0.05$. Part keypoints is the set of face centers, edge midpoints, and corners of each part’s axis-aligned bounding box. We then define contacts as the midpoint of the closest pair of keypoints of two connected parts, and obtain 3 linear constraints (one for each axis) for each pair of connected parts that enforces the contact point to maintain connectivity during deformation. We obtain a number of linear constraints from the collection of contacts that results in a different number of linear constraints for each source model. We concatenate all the linear constraints and represent these with constraint matrix B_s for source model s . Let Q_s be the nullspace, *i.e.* columns representing the nullspace basis vectors, of B_s computed via SVD, then the constraint projection matrix of s is given by $Q_s Q_s^T$.

Training details and training time. We alternately update the retrieval module and the deformation module at each iteration during our training procedure, and train for 300 epochs. To speedup training, we cache the distances to the sources for each target and update this cache every 5 epochs. We use a batch size of 16 targets in each iteration, the SGD optimizer with learning rate of 0.001, momentum of 0.9 and weight decay of 0.0005. For the inner deformation optimization, also use the SGD optimizer with a learning rate of 0.05 until the termination criteria is reached, which is when the fitting loss decreases by less than 10^{-5} or the maximum number of 5000 iterations is reached.

For our joint training module, we first train our Structure-Aware neural deformation module until convergence on random pairs, and also train our retrieval module on random pairs to initialize our joint training optimization scheme. Also note that when training image-based ResNet encoder for the retrieval and deformation modules, we warm-start with weights that are pre-trained on ImageNet, and only train the fourth block and the final fully-connected layers.

Training takes 18 and 40 hours on point clouds and images, respectively, for the chair class. With the inner loop direct optimization, the corresponding training time for chairs takes 3 days for both the point cloud and image experiments as the inner optimization dominates the runtime.

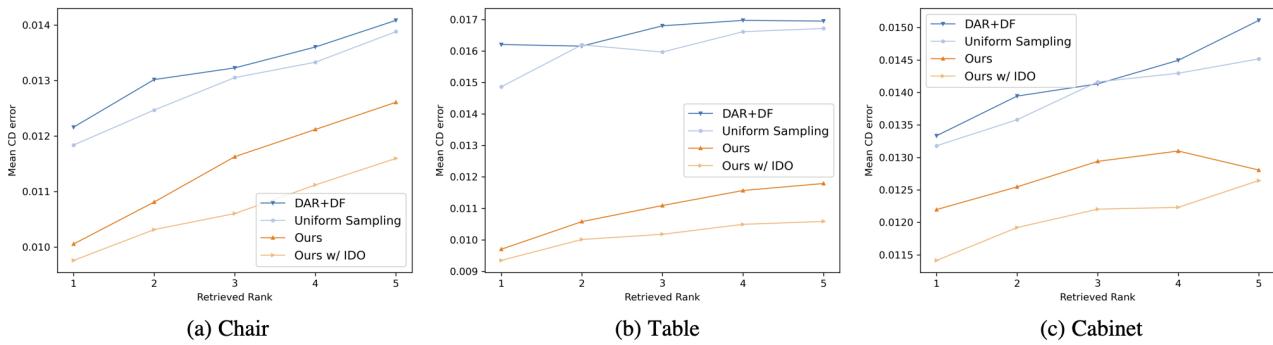


Figure A1. Quantitative evaluation of Image-to-Mesh.

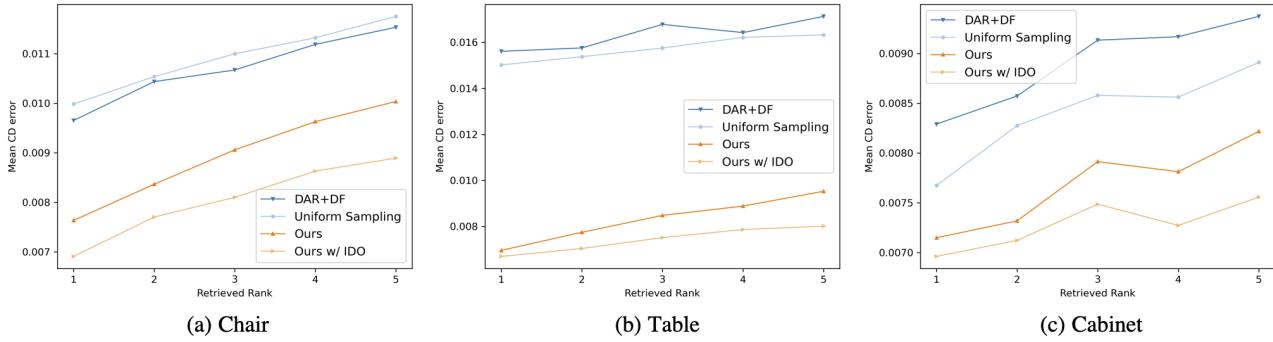


Figure A2. Quantitative evaluation of Points-to-Mesh.

A.2. Additional Results

A.2.1 Additional Quantitative Evaluations

No connectivity constraint ablation. We also test our joint training scheme in the setting where the source database models do not have connectivity constraints. In this set-up we do not use the constraint projection matrix. Table A1 shows that even in the set-up with no connectivity, our approach achieves the best results in all three object classes.

Retrieval-and-deformation results for different retrieved sources. We further evaluate how well our method works with other than top-1 retrieved source. In particular, we plot the mean chamfer distance for the k^{th} retrieved source, for $k = 1, 2, 3, 4, 5$.

For image-to-mesh experiment, we show the result in Figure A1, which complements Table 1 of the main paper. For points-to-mesh experiment, we show the result in Figure A2, which complements Table 3 of the main paper. Note that in both cases the chamfer distance for up to top-5 retrieved results is consistently lower than the baselines.

Retrieval module evaluation. We further evaluate the retrieval modules of our joint approach compared to the baselines. To evaluate the retrieval module, we report both *rank-*

ing evaluation and *recall* similar to the metrics used in [41].

One challenge in defining an evaluation metric is that we do not know which source model should be used for each target. Thus, to create the ground truth we use *oracle retrieval*, where we use the each method’s deformation module to deform each source to the target, and assume that if we sort the sources by the chamfer distance, it will give us the desired ground truth ordering for the retrieval.

Ranking evaluation reports the average rank of the top-1 retrieved model with respect to this ground truth. We report the metrics for image-to-mesh (Table A2) and points-to-mesh (Table A3) experiments, across all categories, and see consistent improvement with respect to the baselines.

We also report the recall of retrieval modules. For *recall@ N* , a correct match is defined as the case where at least one of the top- N retrieved models is in the top-5 ranks based on the oracle retrieval module. We report both *recall@1* and *recall@5*. We report the metrics for image-to-mesh (Table A4) and points-to-mesh (Table A5) experiments, across all categories, and see consistent improvement with respect to the baselines.

Additional object categories. We ran experiments on additional categories (vases, beds, trash cans), and a combination of categories (chairs+tables+cabinets). As shown in

	Chair	Table	Cabinet
DAR+DF	23.98	59.51	19.50
Uniform Sampling	20.88	53.01	23.39
Ours	15.35	22.19	21.70
Ours w/ IDO	21.94	36.92	16.89

Table A2. **Ranking evaluation for retrieval.** Comparing our method using the ranking evaluation metric on image-to-mesh benchmark. Numbers show the average rank of the retrieved model. (Lower is better)

	Chair	Table	Cabinet
DAR+DF	13.88	76.25	20.20
Uniform Sampling	18.27	72.44	23.44
Ours	6.37	6.97	17.91
Ours w/ IDO	6.62	18.03	18.22

Table A3. **Ranking evaluation for retrieval.** Comparing our method using the ranking evaluation metric on points-to-mesh benchmark. Numbers show the average rank of the retrieved model. (Lower is better)

	Chair		Table		Cabinet	
	recall@1	recall@5	recall@1	recall@5	recall@1	recall@5
DAR+DF	37.53	74.65	14.55	43.46	22.37	57.89
Uniform Sampling	38.94	75.56	21.90	54.79	21.05	53.81
Ours	53.60	81.03	53.81	82.93	30.70	61.40
Ours w/ IDO	45.65	77.30	35.83	69.35	35.96	65.79

Table A4. **Recall evaluation for retrieval.** Comparing our method using the ranking evaluation metric on image-to-mesh benchmark. Numbers show recall@1 and recall@5. A correct retrieval is when the top-1 and top-5 retrieved models is in the top-5 ranks based on the oracle retrieval. (Higher is better)

	Chair		Table		Cabinet	
	recall@1	recall@5	recall@1	recall@5	recall@1	recall@5
DAR+DF	61.56	93.54	23.57	54.54	39.83	72.29
Uniform Sampling	53.27	89.98	25.03	59.16	39.83	67.97
Ours	75.31	97.02	73.71	96.50	48.05	76.19
Ours w/ IDO	76.22	96.60	55.17	89.72	38.53	77.06

Table A5. **Recall evaluation for retrieval.** Comparing our method using the ranking evaluation metric on points-to-mesh benchmark. Numbers show recall@1 and recall@5. A correct retrieval is when the top-1 and top-5 retrieved models is in the top-5 ranks based on the oracle retrieval. (Higher is better)

	Vase	Bed	Trash Can	Combined
DAR+DF	1.538	4.498	0.889	1.968
Uniform Sampling	1.633	4.196	0.886	1.821
Ours	1.384	2.138	0.863	0.810

Table A6. **Additional object categories.** Comparing our method to various baselines and ablations on additional object classes and mixture of categories (chamfer distances, $\times 10^{-2}$).

Table A6, we got a comparable performance and improvement over baselines.

Perceptual Metric. We performed a user study comparing our approach to the DAR+DF baseline. We asked 60 partic-

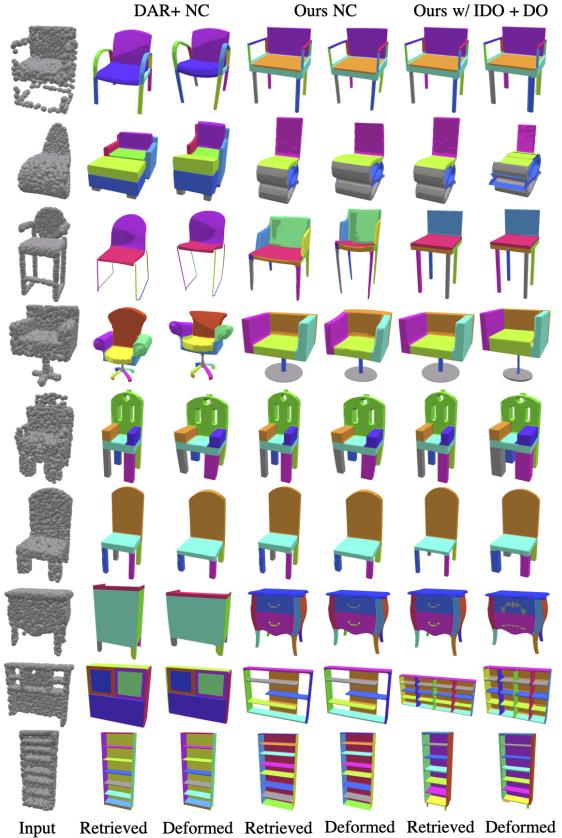


Figure A3. More qualitative results on Neural Cages [49].

ipants to pick the better match to input point clouds on 15 randomly selected targets from the test set, where an option of “no difference” can also be selected. Our approach got an average score of **8.02**, compared to 3.5 for the baseline and 3.48 abstain votes.

A.2.2 Additional Qualitative Results

We provide additional qualitative results using natural images, point cloud scans, and our benchmark as input targets. Note that in all visualizations, we use colors to indicate different segmentations of the source models, where segmentation is essential to the performance of the structure-aware neural deformation module.

Product images targets. Figure A7 shows additional qualitative results of our approach on product images.

Scan2CAD targets. Figure A5 shows additional results of our approach on real scans from the Scan2CAD [2] dataset using the manually segmented PartNet [29] database, while Figure A6 shows the results on real scans using the auto-segmented ComplementMe [37] database.

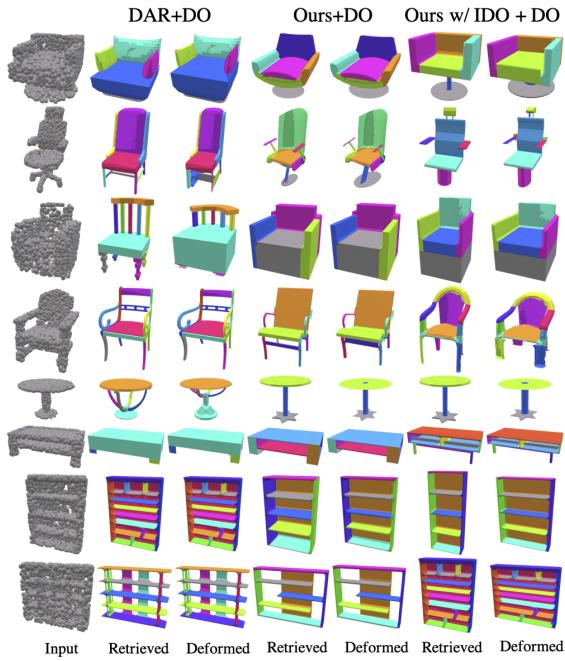


Figure A4. Additional qualitative results on comparisons between our approach and the baselines for the points-to-mesh experiments.

Image-to-Mesh baseline comparison. Figure A8 shows additional qualitative results on the image-to-mesh set-up that compares our method to the baselines.

Points-to-Mesh baseline comparison. Figure A4 shows additional qualitative results of our joint approach compared to the baselines on the points-to-mesh experiment.

Neural cages. Figure A3 shows additional qualitative results of our joint approach on Neural Cages [49].

Points-to-Mesh ablations qualitative results. Figure A9 shows qualitative results of ablations of our joint approach on the points-to-mesh experiment.

A.3. Discussion on [10]

The differences between our work and with [10] are as follows:

1. **Non-learnable deformations:** The fitting module of [10] is *not learnable*; they directly *optimize* parameters of a handcrafted template to fit to an input point cloud. Thus, one of our key contributions, a *retrieval-aware deformation*, is incompatible with their method.
2. **Infeasibility of image-to-mesh:** Without learnable deformations, their method cannot be used for the main application of our method, image-to-mesh generation.

3. **Manually-designed templates:** Designing templates is a tedious manual task that requires significant expertise. Their method requires users to pre-design a set of templates, hence they only use a small set of 21 templates.

4. **Non-scalable system:** While one could address solving our retrieval problem as a classification problem by treating every source shape as a template, this approach is not scalable. Their method requires a pre-process of matching every template to every input shape for training. Their optimization-based deformation module takes 2-3 mins for a single pair, and thus for all 500 sources and 4000 training targets as in our chair dataset, it would take ~ 8 years. Note that this limitation has been addressed in a recent work of Uy et al. [41] who propose to learn a *deformation-aware retrieval* latent space instead of the non-scalable hard shape-to-template assignment (and we extensively compared to Uy et al. [41]).

5. **Specific to template-based deformations:** Our key contribution, *joint* learning for retrieval and deformation, is not constrained to a specific choice of the deformation module.

We also remark that, while both ours and their method leverage on part bounding boxes for deformations, neither of these two were the first to use bounding boxes to deform the underlying geometry (e.g., [23]).

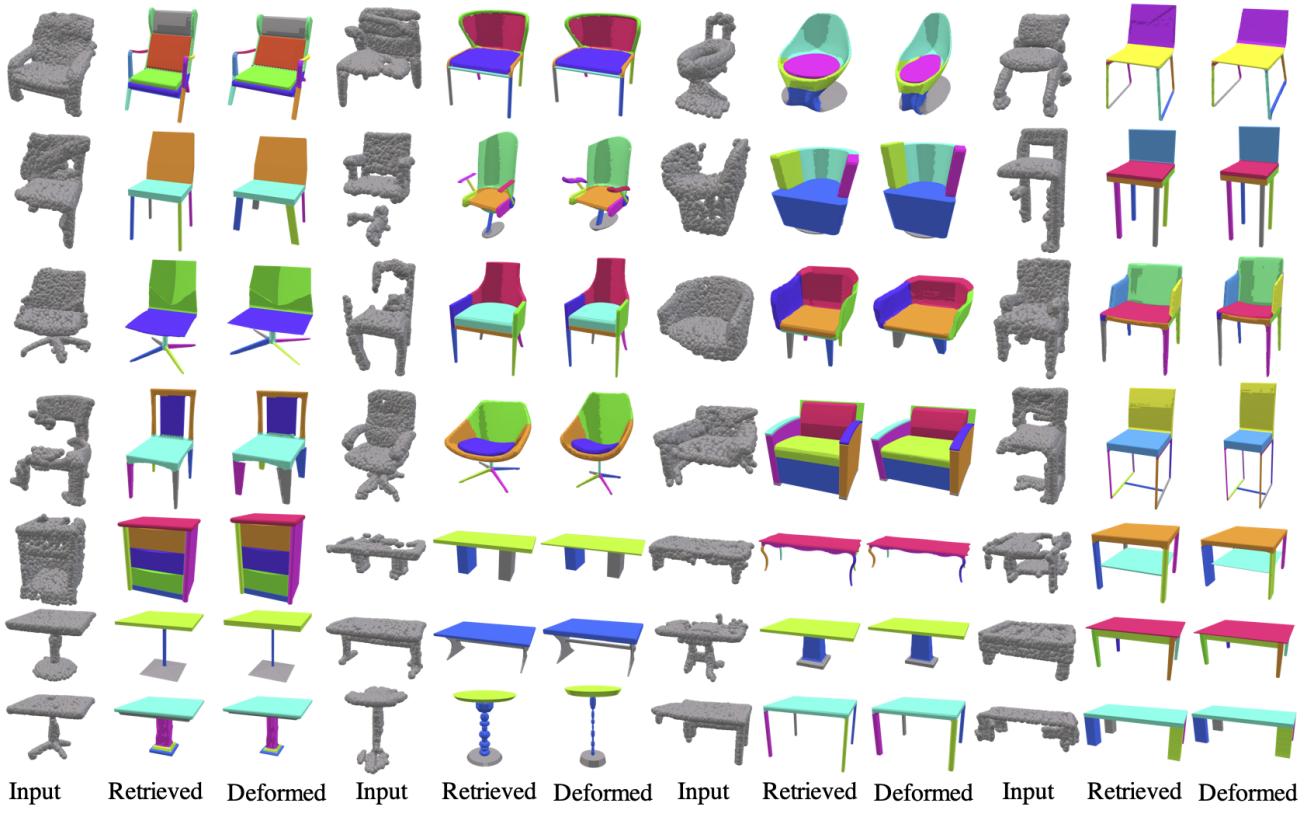


Figure A5. More qualitative results using the Scan2CAD [2] dataset using manually segmented shapes in PartNet [29].

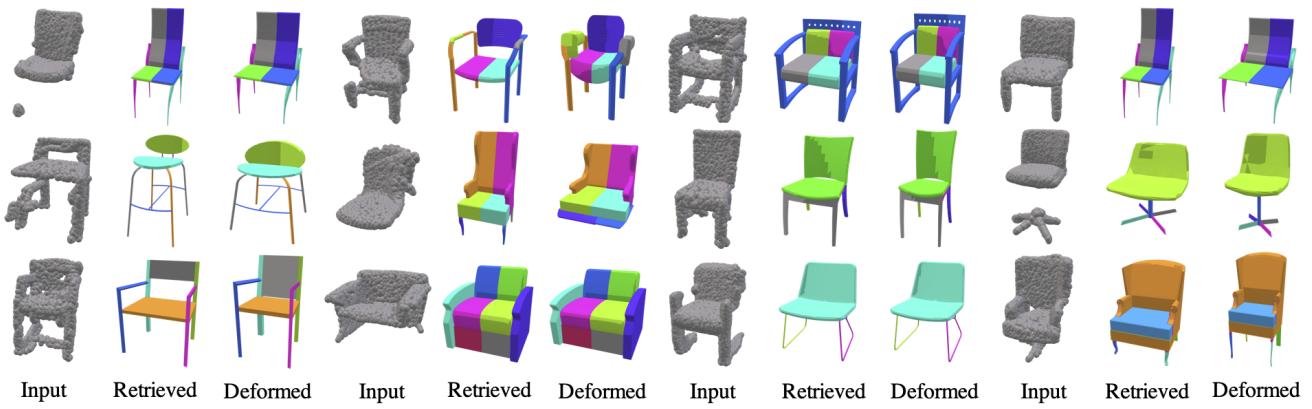
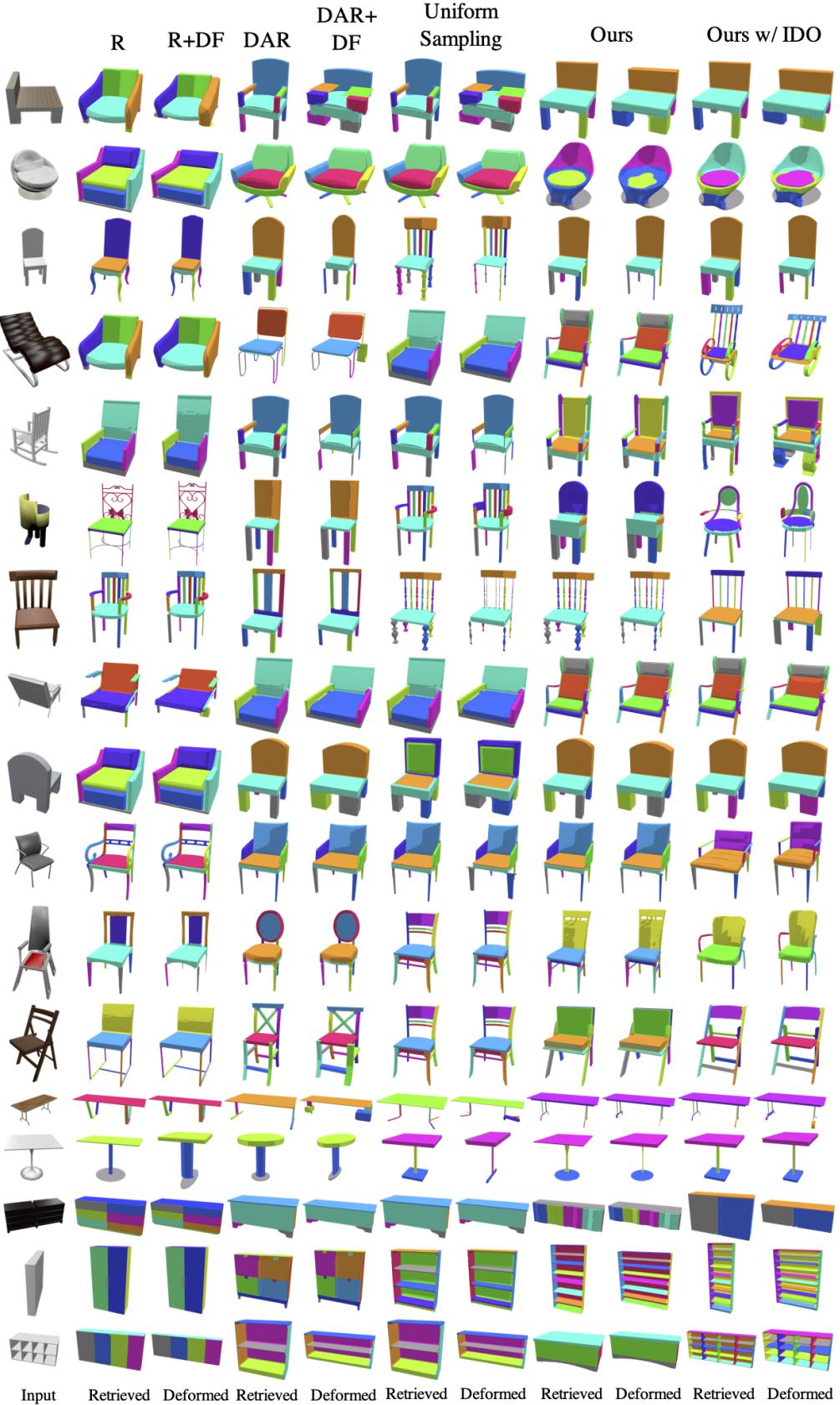


Figure A6. More qualitative results using the Scan2CAD [2] dataset using autosegmented shapes in ComplementMe [37].



Figure A7. More qualitative results on product images.



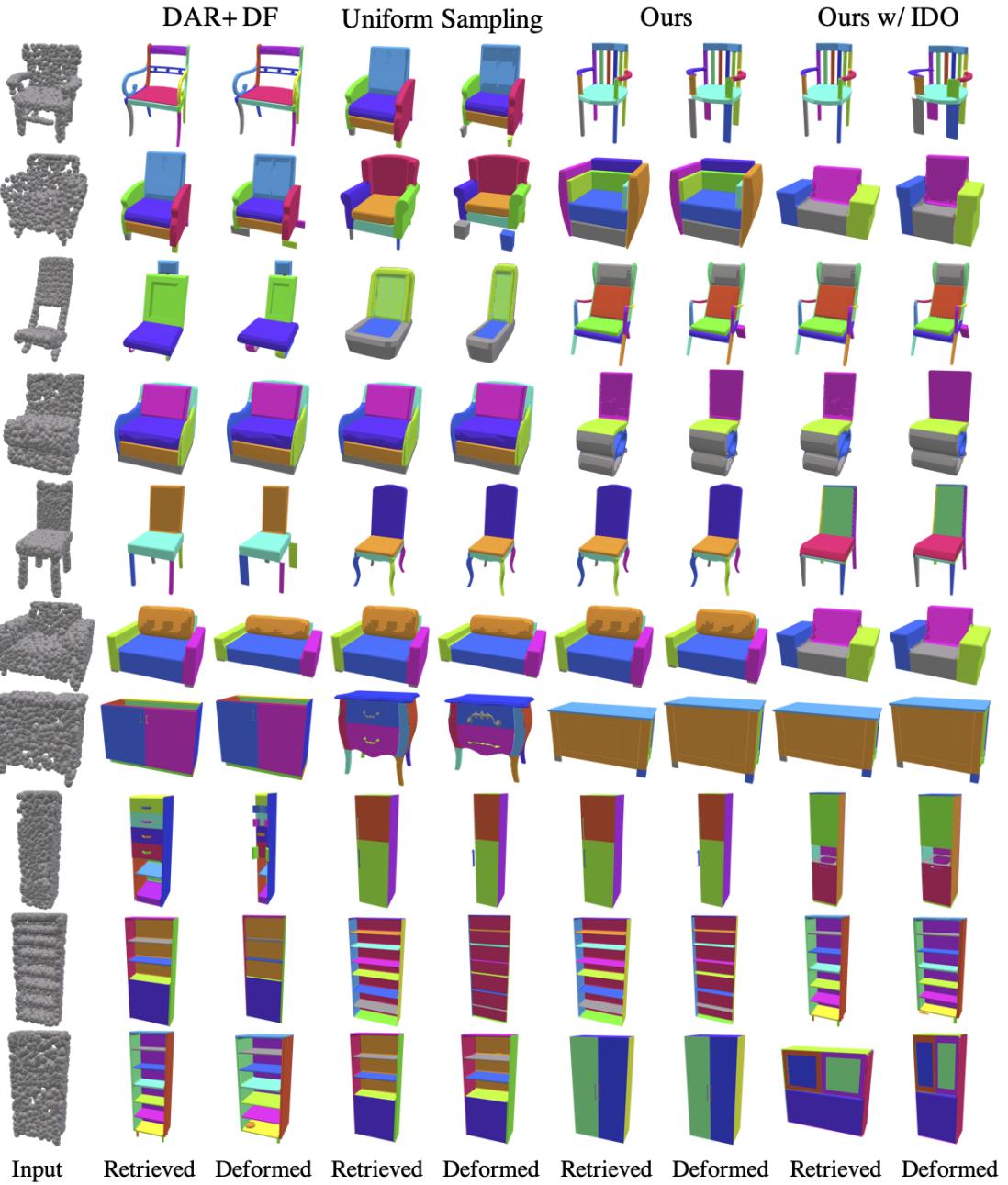


Figure A9. More qualitative results on Points-to-Mesh.