Joint Learning of 3D Shape Retrieval and Deformation

Mikaela Angelina Uy[1] Vladimir G. Kim[2] Minhyuk Sung[3] Noam Aigerman[2]
Siddhartha Chaudhuri[2,4] Leonidas Guibas[1]
[1]Stanford University [2]Adobe Research [3]KAIST [4]IIT Bombay

S.1. 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 [28, 24, 5], atlases [12], volumetric grids [7, 40], point clouds [1, 43, 8], and meshes [37, 39]. 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 [22, 10, 25]. These generative techniques are commonly used jointly with 2D CNNs [29] or shape encoders [46] to enable creating a shape based on some partial observations, such as a natural image [11] or a point scan [6]. A simple shape retrieval [23] 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 [33].

Deformation-Aware Retrieval. Direct retrieval has the advantages of producing stock-quality meshes [34, 35], however, unless the database contains all possible objects, might not produce a good fit for an encoded target. Prior works [27, 30, 36] 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 [27], but is however computationally expensive. Schulz *et al*. [30] 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*. [36] use a deep embedding to retrieve a shape and then separately deform it to the target by directly optimizing the ARAP [16] 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 [15, 31, 19], as well as human-made [20, 9, 41, 47] shapes. One can only directly optimize if a target is a full shape, however if it of a different modality, such as image or partial scan, one needs to employ priors [14]. Neural techniques have been used to learn such deformation priors from collections of shapes, representing deformations as volumetric warps [17, 21, 45], cage deformations [44], vertex-based offsets [38, 13] or flow-based approaches [18]. 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 [17], cage mesh [44] or number of vertices [38]. 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 [26, 42, 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.

S.2. Implementation Details

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.

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 reach 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 $\mathbf{t}$, 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 $\mathcal{L}_{\text{CD}}$ 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 $\mathcal{L}_{\text{def}}$ is defined in Equation 4 in the main paper.

Connectivity constraint.

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, making the deformation functions $\{\mathcal{D}_{\mathbf{s}}\}$ even more source-dependent.

We precompute the constraint projection matrix for each source $\mathbf{s} \in \mathbf{S}$ in an automatic pre-processing step, where we first identify contacts based on the distance between the closest pairs of keypoints between pairs of parts $(\mathbf{s}^i_{\mathcal{D}}, \mathbf{s}^j_{\mathcal{D}})$. Parts $\mathbf{s}^i_{\mathcal{D}}$ and $\mathbf{s}^j_{\mathcal{D}}$ 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 $B_{\mathbf{s}}$ for source model $\mathbf{s}$. Let $Q_{\mathbf{s}}$ be the nullspace, *i.e.* columns representing the nullspace basis vectors, of $B_{\mathbf{s}}$ computed via SVD, then the constraint projection matrix of $\mathbf{s}$ is given by $Q_{\mathbf{s}} Q_{\mathbf{s}}^T$.

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.

S.3. Retrieval in Latent Space

The retrieval space $\mathcal{R}$ is defined similarly to Uy et al. [36], 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 $\mathbf{s}^v_{\mathcal{R}} \in \mathbb{R}^{n_4 \times n_4}$ 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:

$$d(\mathbf{s},\mathbf{t})_{\mathcal{R}} = \sqrt{(\mathbf{s}_{\mathcal{R}} - \mathbf{t}_{\mathcal{R}})^T \mathbf{s}^v_{\mathcal{R}} (\mathbf{s}_{\mathcal{R}} - \mathbf{t}_{\mathcal{R}})}. \quad (1)$$

During training we optimize the parameters of the encoder $E_{\mathcal{R}}(\mathbf{t})$ as well as latent codes and variances for each source, $\mathbf{s}_{\mathcal{R}}, \mathbf{s}^v_{\mathcal{R}}$. $\mathbf{s}_{\mathcal{R}}$ is obtained by feeding the default shape of source model $\mathbf{s}$ to encoder $E_{\mathcal{R}}(\mathbf{t})$. Different from Uy et al. [36], we optimize $\mathbf{s}^v_{\mathcal{R}}$ 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.

S.4. 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 [26], and we select random 10% of the data as our sources. Automatically-segmented shapes come from two pre-analyzed classes in ComplementMe [32] (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 points-to-points distances are used for training and evaluation.

S.5. 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 S1 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 S1 and Figure S2. 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 [36] we can 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**).

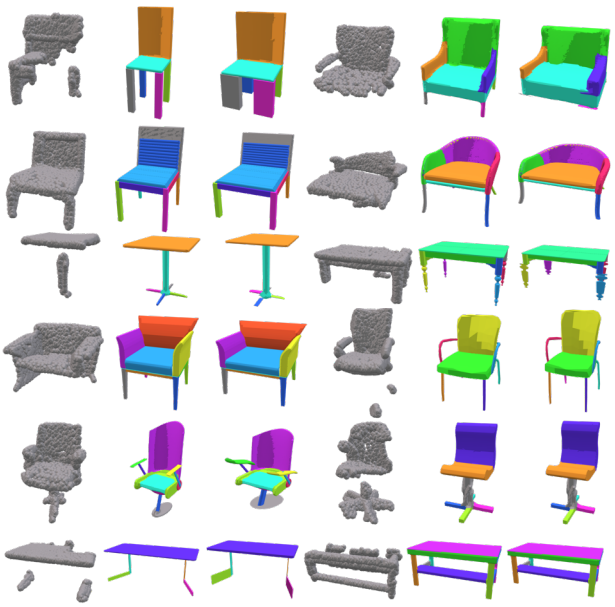


Figure S1. Our results on real scans from the Scan2CAD dataset [2].

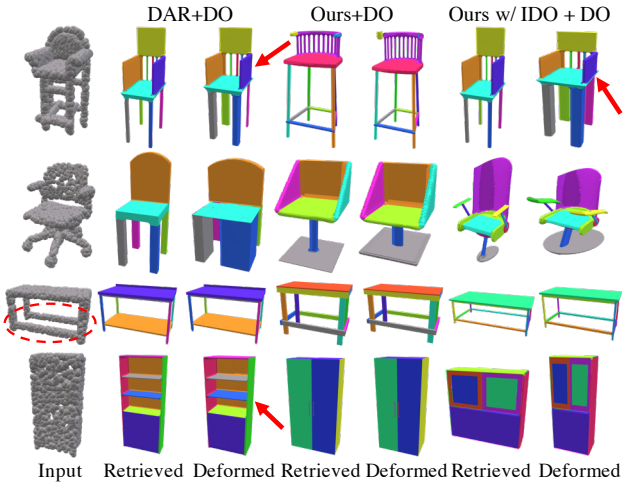


Figure S2. Comparison between our approach and baselines for the point-cloud-to-mesh experiment.

Deformation-Aware Retrieval Baseline. We compare to deformation-aware retrieval [36] (**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 3D scans or images. Our method outperforms this baseline with and without the direct optimization step (Table S1).

	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 S1. Comparing our method to various baselines and ablations on points-to-mesh benchmark (chamfer distances, $\times 10^{-2}$).

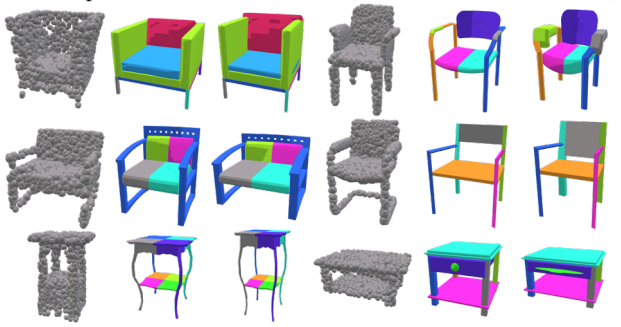


Figure S3. Fitting results using auto-segmented sources [32].

Qualitative results in Figure S2, 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 [9] (**Classif**). Instead of using a non-learnable defomation module via direct optimization of handcrafted templates as in [9], 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 S1. Note that this baseline is worse than our method and even [36].

Biased Sampling Ablation. As in the image target case, we demonstrate the importance of biased sampling in joint training (Table S1, **Uniform Sampling**).

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 [32] grouping connected components of meshes.

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

Table S2. Using auto-segmented models as the source database [32] (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 S3. Using our joint training with Neural Cages [44] deformation module (chamfer distances, $\times 10^{-2}$).

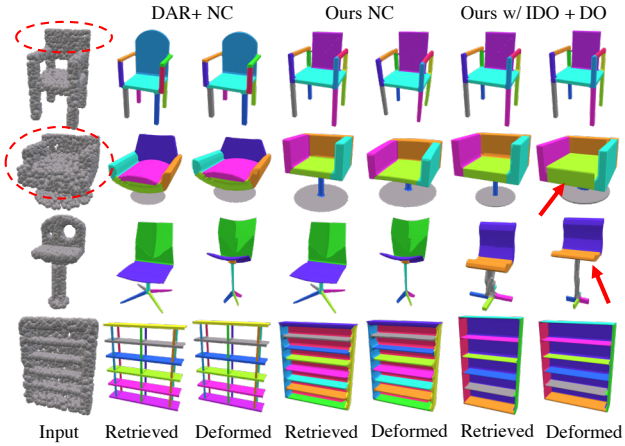


Figure S4. Using Neural Cages [44] as a deformation module in our joint training.

As shown in Figure S3, 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 S2).

Performance with Neural Cages [44]. 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 [44], 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 [36] 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 S3). Qualitative results in Figure S4 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.

We remark that our joint training does not constraint the choice of the neural deformation module. One can choose any module based on its strengths and weaknesses. For in-

	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 S4. Improvement in deformation module for points-to-mesh (with oracle retrieval) due to joint training (chamfer $\times 10^{-2}$).

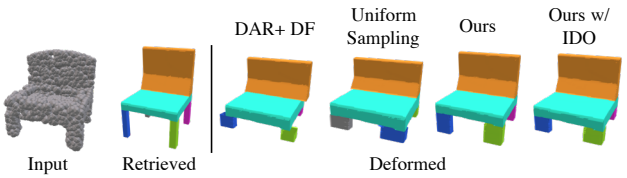


Figure S5. We pick an example where retrieved mesh is the same for all methods, and show that joint training also improves the quality of the neural deformation module on its own.

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

Table S5. Performance on of our method and various baselines with different source database sizes (chamfer distances, $\times 10^{-2}$).

stance, 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 S4). 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 S4. 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 S4. We demonstrate a qualitative example in Figure S5 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 varying the size of the database of source models. We randomly sample 50, 100, 200, 400, and 800 chair models from

	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 S6. Our approach compared to the baselines in the setup with no connectivity constraint.

PartNet to construct the source databases. Table S5 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.

S.6. 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 S6 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 $k = 1, 2, 3, 4, 5$.

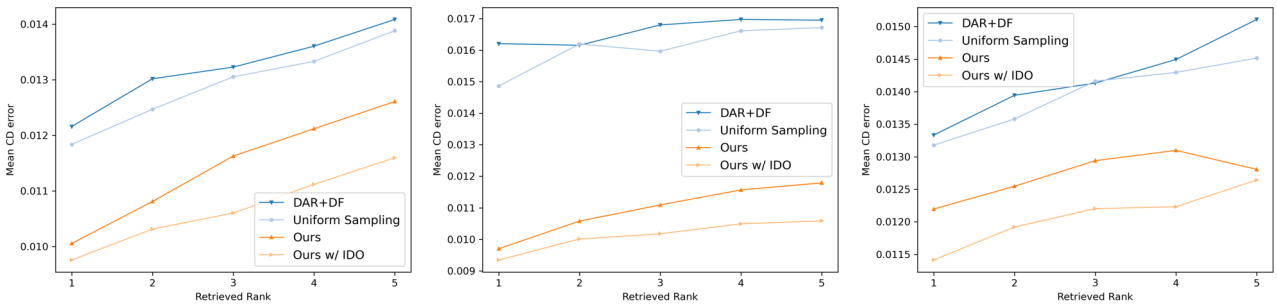
For image-to-mesh experiment, we show the result in Figure S6, which complements Table 1 of the main paper. For points-to-mesh experiment, we show the result in Figure S7, which complements Table S1 of the main paper. Note that in both cases the chamfer distance for up to top-5 retrieved results is consistently 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 *ranking evaluation* and *recall* similar to the metrics used in [36].

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 S7) and points-to-mesh (Table S8) 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



(a) Chair (b) Table (c) Cabinet

Figure S6. Quantitative evaluation of Image-to-Mesh.

(a) Chair (b) Table (c) Cabinet

Figure S7. Quantitative evaluation of Points-to-Mesh.

	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 S7. **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 S8. **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)

recall@1 and recall@5. We report the metrics for image-to-mesh (Table S9) and points-to-mesh (Table S10) 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 Table S11, 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 participants 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.

	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 S9. **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 S10. **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 S11. **Additional object categories.** Comparing our method to various baselines and ablations on additional object classes and mixture of categories (chamfer distances, $\times 10^{-2}$).

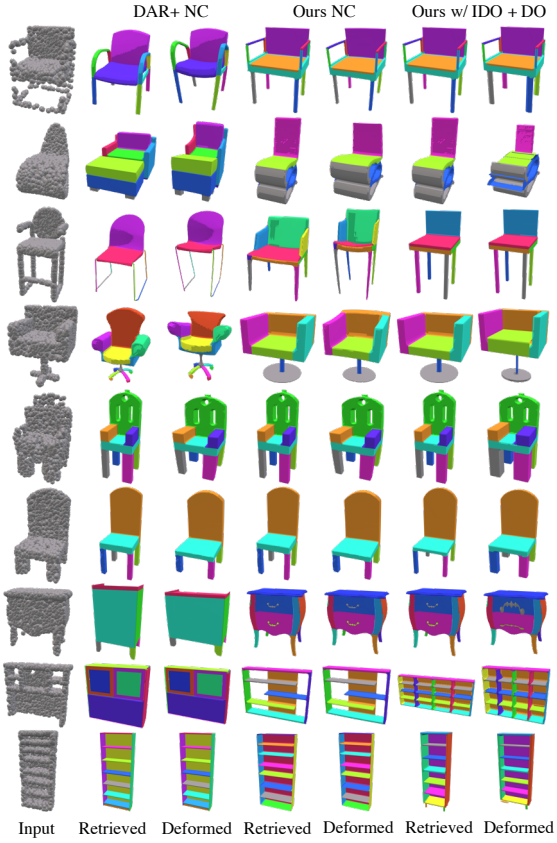


Figure S8. More qualitative results on Neural Cages [44].

S.6.1 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,

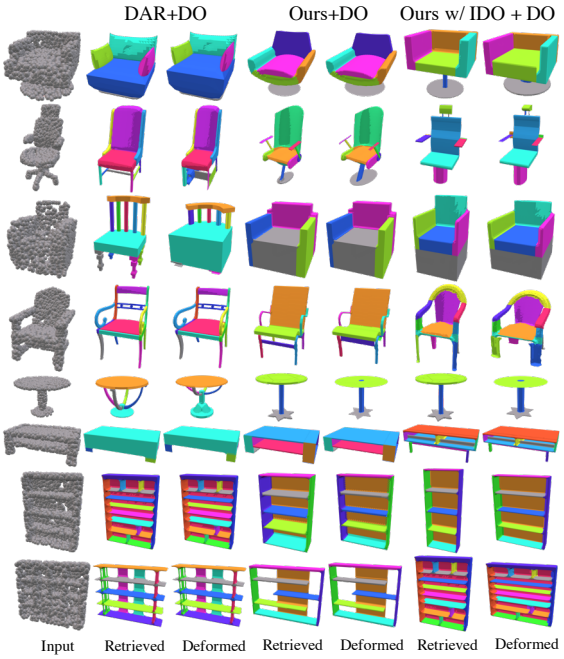


Figure S9. Additional qualitative results on comparisons between our approach and the baselines for the points-to-mesh experiments.

where segmentation is essential to the performance of the structure-aware neural deformation module.

Product images targets. Figure S12 shows additional qualitative results of our approach on product images.

Scan2CAD targets. Figure S10 shows additional results of our approach on real scans from the Scan2CAD [2] dataset using the manually segmented PartNet [26] database, while Figure S11 shows the results on real scans using the auto-segmented ComplementMe [32] database.

Image-to-Mesh baseline comparison. Figure S13 shows additional qualitative results on the image-to-mesh set-up that compares our method to the baselines.

Points-to-Mesh baseline comparison. Figure S9 shows additional qualitative results of our joint approach compared to the baselines on the points-to-mesh experiment.

Neural cages. Figure S8 shows additional qualitative results of our joint approach on Neural Cages [44].

Points-to-Mesh ablations qualitative results. Figure S14 shows qualitative results of ablations of our joint approach on the points-to-mesh experiment.

S.7. Discussion on [9]

The differences between our work and with [9] are as follows:

1. **Non-learnable deformations**: The fitting module of [9] 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 $\sim$ 8 years. Note that this limitation has been addressed in a recent work of Uy et al. [36] 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. [36]).
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., [20]).

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J Guibas. Learning representations and generative models for 3D point clouds. In *ICML*, 2018. 1

[2] Armen Avetisyan, Manuel Dahnert, Angela Dai, Manolis Savva, Angel X Chang, and Matthias Nießner. Scan2CAD: Learning cad model alignment in RGB-D scans. In *CVPR*, 2019. 1, 3, 7, 9

[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. 3

[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. 1

[5] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *CVPR*, 2019. 1

[6] Angela Dai and M. Nießner. Scan2Mesh: From unstructured range scans to 3d meshes. In *CVPR*, 2019. 1

[7] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3D-encoder-predictor CNNs and shape synthesis. In *CVPR*, 2017. 1

[8] H. Fan, H. Su, and L. Guibas. A point set generation network for 3D object reconstruction from a single image. In *CVPR*, 2017. 1

[9] V. Ganapathi-Subramanian, O. Diamanti, S. Pirk, Chengcheng Tang, M. Nießner, and L. Guibas. Parsing geometry using structure-aware shape templates. In *3DV*, 2018. 1, 4, 8

[10] 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. 1

[11] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh R-CNN. In *ICCV*, 2019. 1

[12] 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. 1

[13] 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

[14] 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. 1

[15] Qixing Huang, B. Adams, Martin Wicke, and L. Guibas. Non-rigid registration under isometric deformations. *Computer Graphics Forum*, 2008. 1

[16] Takeo Igarashi, Tomer Moscovich, and John F. Hughes. As-rigid-as-possible shape manipulation. In *SIGGRAPH*, 2005. 1

[17] 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. 1

[18] Chiyu Jiang, Jingwei Huang, Andrea Tagliasacchi, and Leonidas Guibas. ShapeFlow: Learnable deformations among 3D shapes. In *NeurIPS*, 2020. 1

[19] Tao Ju, Scott Schaefer, and Joe Warren. Mean value coordinates for closed triangular meshes. In *SIGGRAPH*, 2005. 1

[20] 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. 1, 8

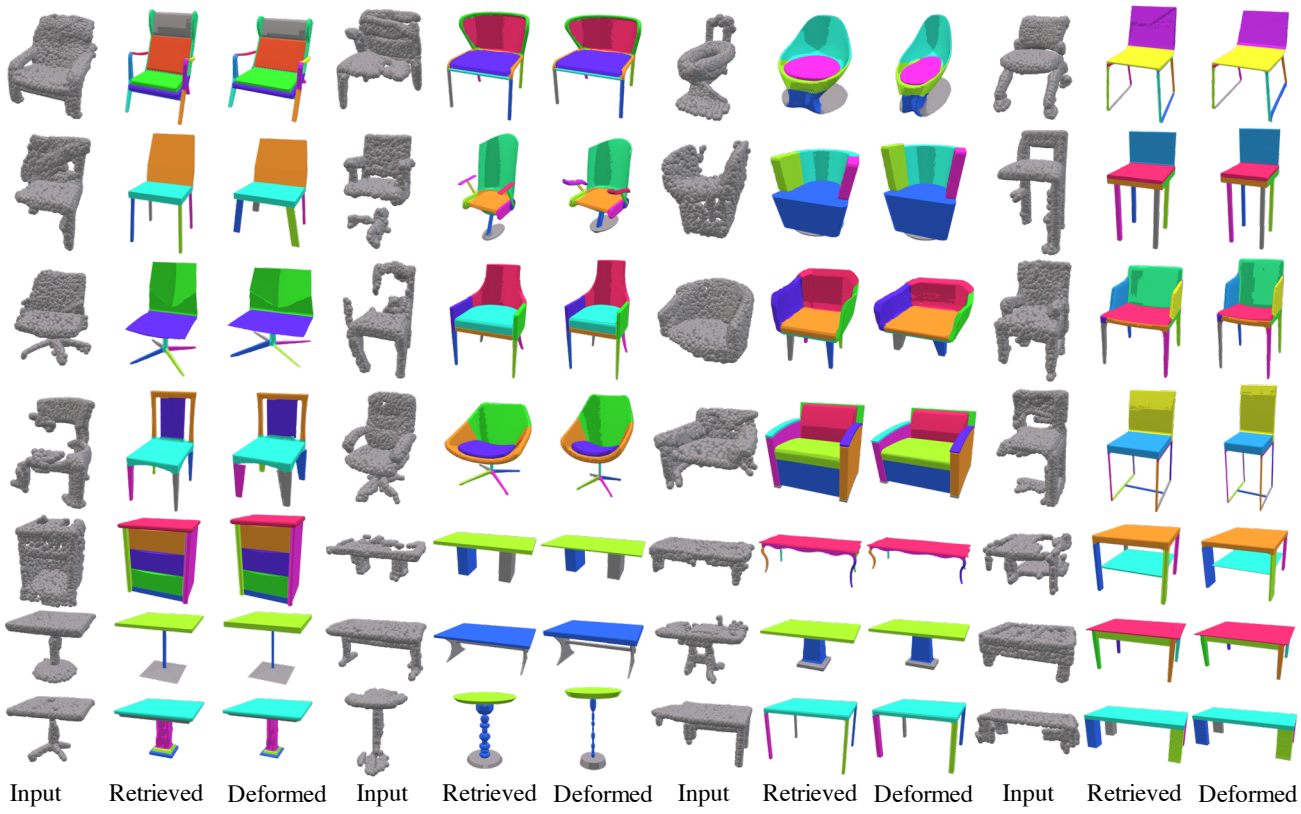


Figure S10. More qualitative results using the Scan2CAD [2] dataset using manually segmented shapes in PartNet [26].

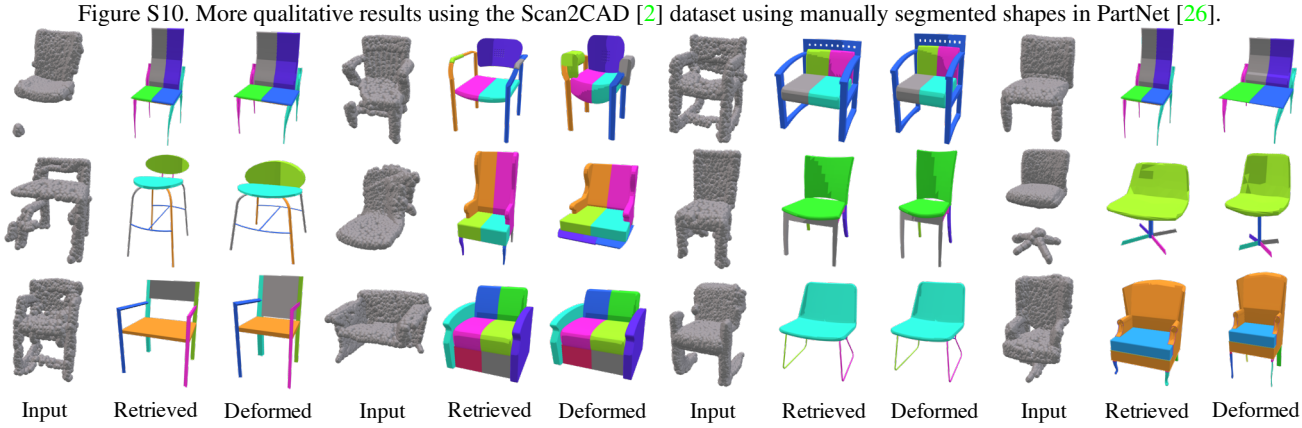


Figure S11. More qualitative results using the Scan2CAD [2] dataset using autosegmented shapes in ComplementMe [32].

[21] 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. 1

[22] Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. Grass: Generative recursive autoencoders for shape structures. In *SIGGRAPH*, 2017. 1

[23] 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

[24] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019. 1

[25] 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. 1

[26] 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. 1, 3, 7, 9

Figure S12. More qualitative results on product images.

[27] Liangliang Nan, Ke Xie, and Andrei Sharf. A search-classify approach for cluttered indoor scene understanding. *ACM Transactions on Graphics*, 2012. 1

[28] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. 1

[29] 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. 1

[30] Adriana Schulz, Ariel Shamir, Ilya Baran, David I. W. Levin, Pitchaya Sitthi-Amorn, and Wojciech Matusik. Retrieval on parametric shape collections. *ACM Transactions on Graphics*, 2017. 1

[31] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Eurographics Symposium on Geometry Processing*, 2007. 1

[32] Minhyuk Sung, Hao Su, Vladimir G. Kim, Siddhartha Chaudhuri, and Leonidas Guibas. ComplementMe: Weakly-supervised component suggestions for 3D modeling. In *SIGGRAPH Asia*, 2017. 3, 4, 7, 9

[33] 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

[34] Trimble. 3D warehouse. 1

[35] TurboSquid. TurboSquid. 1

[36] Mikaela Angelina Uy, Jingwei Huang, Minhyuk Sung, Tolga Birdal, and Leonidas Guibas. Deformation-Aware 3D model embedding and retrival. In *ECCV*, 2020. 1, 2, 3, 4, 5, 8

[37] 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. 1

[38] Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. 3DN: 3D deformation network. In *CVPR*, 2019. 1

[39] Chao Wen, Yinda Zhang, Zhuwen Li, and Yanwei Fu. Pixel2Mesh++: Multi-view 3D mesh generation via deformation. In *ICCV*, 2019. 1

[40] 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. 1

[41] 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. 1

[42] Kai Xu, Hanlin Zheng, Hao Zhang, Daniel Cohen-Or, Ligang Liu, and Yueshan Xiong. Photo-inspired model-driven 3d object modeling. In *SIGGRAPH*, 2011. 1

[43] 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. 1

[44] Wang Yifan, Noam Aigerman, Vladimir Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. Neural cages for detail-preserving 3D deformations. In *CVPR*, 2020. 1, 4, 7

[45] Ersin Yumer and Niloy J. Mitra. Learning semantic deformation flows with 3d convolutional networks. In *ECCV*, 2016. 1

[46] 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. 1

[47] 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. 1

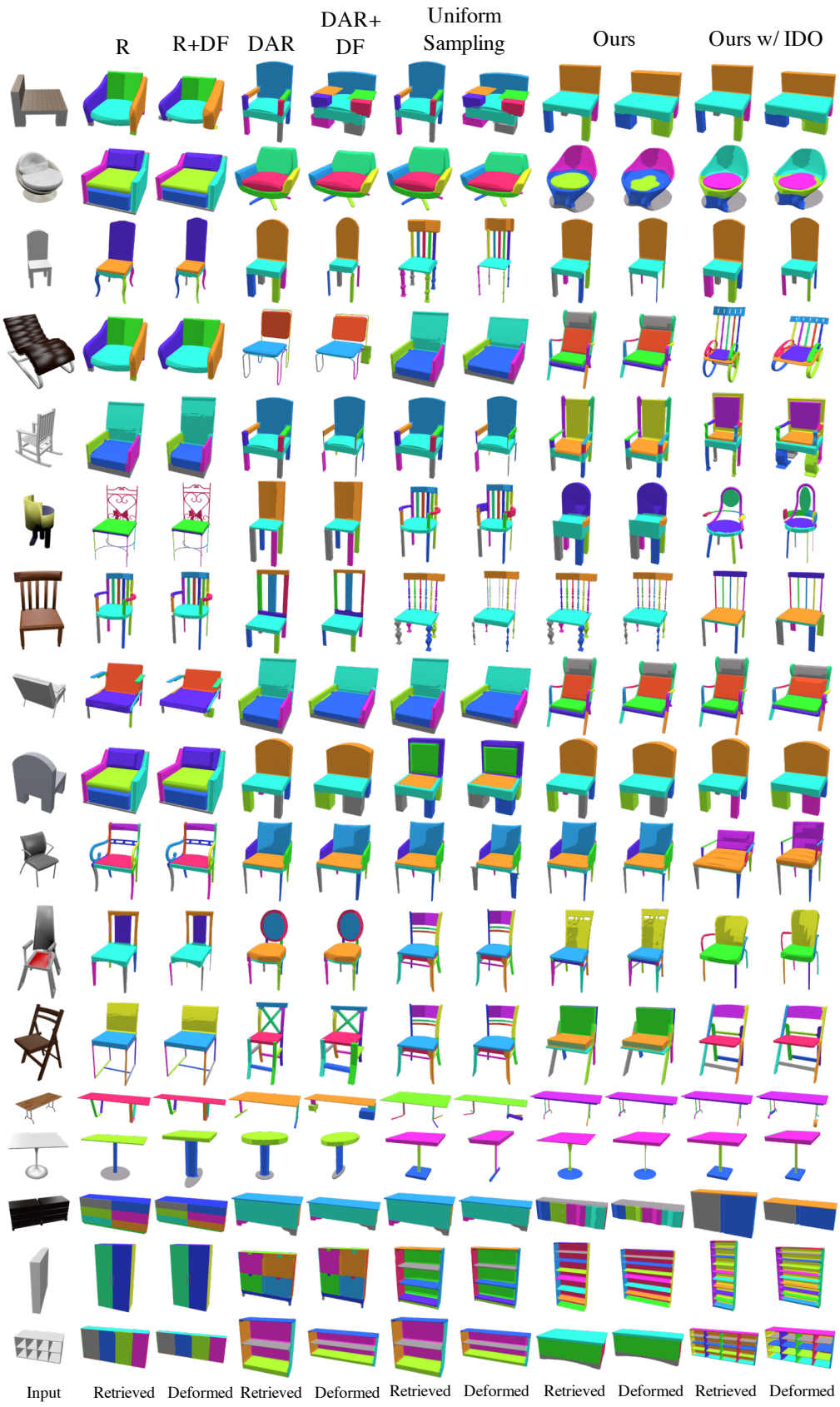


Figure S13. More qualitative results on Image-to-Mesh.

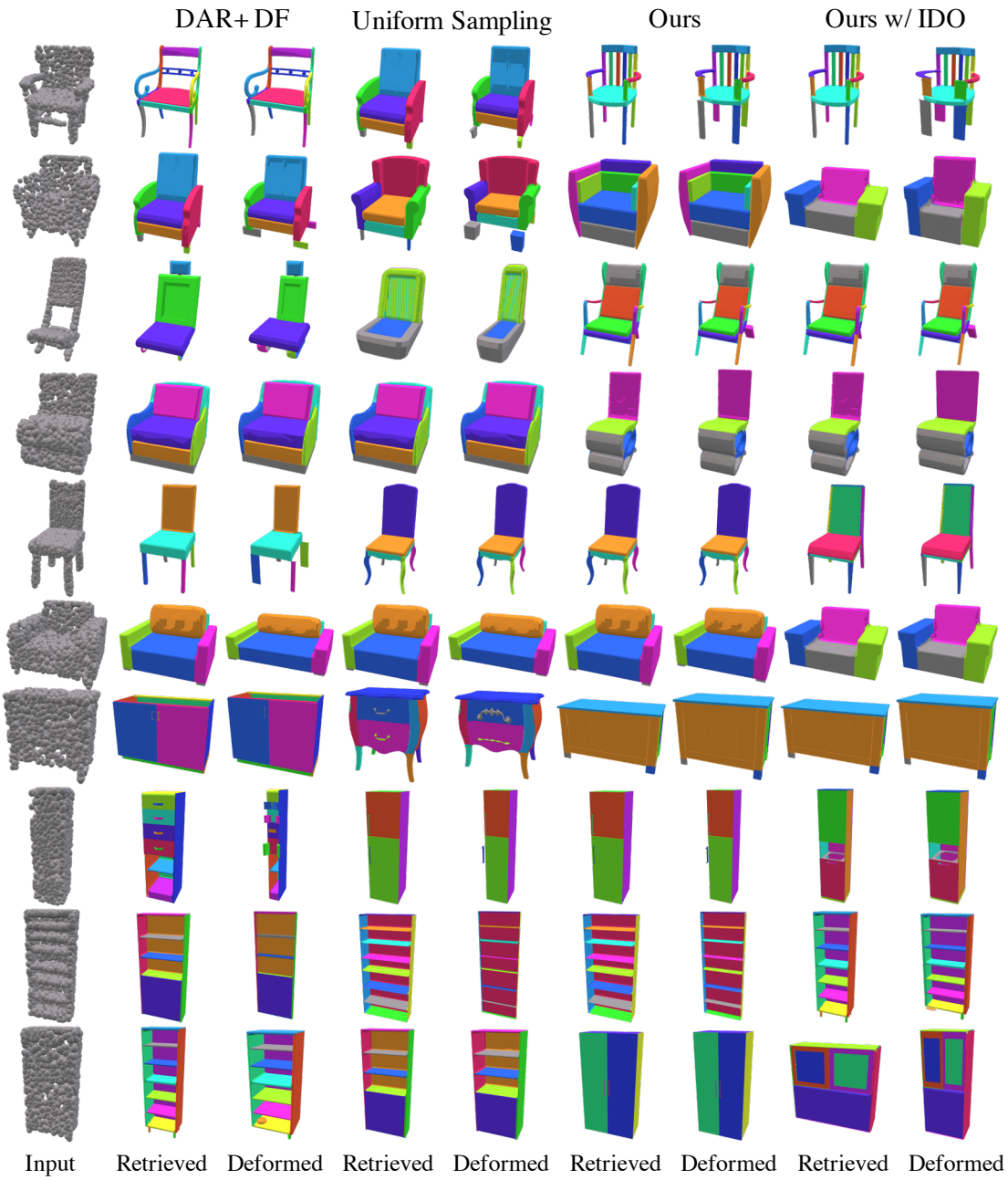


Figure S14. More qualitative results on Points-to-Mesh.