Few-Shot Physically-Aware Articulated Mesh Generation via Hierarchical Deformation
Supplementary Materials

The appendix provides a list of supplemental materials to support the main paper.

- **Further Explanations on the Method –** We provide additional details for some components and algorithms to complement the main paper.
  - *Convex-Level Deformation Generative Model* (Sec. A.1). We include a more detailed explanation regarding the model including two tricks we use to parameterize deformations
  - *Convex Deformation Synchronization* (Sec. A.2). We explain our alternative optimization strategy to calculate synchronization matrices in more detail.
  - *Physics-Aware Deformation Correction* (Sec. A.3). We explain the calculation processes of the physical penalty $\mathcal{L}_{phy}$ and the projection loss $\mathcal{L}_{proj}$ in more detail.
  - *Additional Explanations* (Sec. A.4). We include more details regarding the practical implementation of the method.
- **Additional Experiments –** We present additional experiments to further prove the effectiveness of our method.
  - *Few-Shot Mesh Generation* (Sec. B.1). We further examine the few-shot generation ability of our hierarchical deformation strategy by testing it on different few-shot generation settings and also on rigid mesh categories.
  - *Transfer Learning for Convex Deformations* (Sec. B.2). We demonstrate why we choose to use convexes as intermediates to transfer cross-category shared shape patterns. Besides, we also explore the influence of source categories in the transfer learning on the few-shot generation performance.
  - *Generation via Deformation* (Sec. B.3). We leverage a different mesh generation technique and design a second approach trying to solve the problem. By comparing this strategy to our method, we cast some thoughts on the design philosophy of strategies to solve the few-shot and physically-aware generation challenges.
- **Experimental Settings –** We provide additional information about our experimental settings.
  - *Datasets* (Sec. C.1). We provide more information on our datasets for pre-training and evaluation.
  - *Baselines* (Sec. C.2). We explain our modifications and improvements on baseline methods so that we can adapt them to the articulated mesh generation problem.
  - *Metrics* (Sec. C.3). We provide additional calculation details of the evaluation metrics.
  - *Additional Experimental Settings* (Sec. C.4). We further discuss some additional experimental settings.

Project page with code is available at meowuu7.github.io/few-arti-obj-gen.

A. Further Explanations on the Method

A.1. Convex-Level Generative Model

In the method, we design a convex-level generative model to parameterize vertex-level deformation offset d_c into a low dimensional space. We leverage two tricks to parameterize d_c: 1) using cages to control per-vertex deformations and 2) using dictionaries to record common deformation patterns. We elaborate on details of the above tricks that are not covered in the main text in the following text.

Cages to control convex deformation. To form the cage t_c of the convex c containing N_c vertices, we deform a template mesh based upon the shape of the convex c. Specifically, given a template mesh, *i.e.,* a sphere surface mesh, t_c with $N_t(N_t \ll N_c)$ vertices, we deform t_c to form the cage of c via the following steps: 1) Assume the vertex sets of t_c and c are $\mathcal{V}_t$ and $\mathcal{V}_c$ respectively. Find a mapping from each cage vertex, saying $v_t \in \mathcal{V}_t$, to a vertex v_c in the convex c, *i.e.*, $m(v_t) = v_c, v_t \in \mathcal{V}_t, v_c \in \mathcal{V}_c(m(v_{t_1}) \neq m(v_{t_2}), \forall v_{t_1} \neq v_{t_2})$ such that we can minimize $\sum_{v_t \in \mathcal{V}_t} \|m(v_t) - v_t\|_2$. We use "linear_sum_assignment" function implemented in package "scipy" to find the mapping $m(\cdot)$. 2) Deform v_t to $\hat{v}_t =$

$v_t + (1-\epsilon)\cdot(m(v_t) - v_t)$, where ϵ is a hyper-parameter, which is set to 0.05 in our experiment.

Such a heuristic deformation strategy works well in our problem considering shapes we want to deform here are near-to-convex segments.

Using dictionaries to record common deformation patterns. We record common cage deformation patterns using deformation bases. Following a previous work which also learns deformation bases to represent common deformation patterns [6], we wish the deformation basses predicted by our network for each cage should be able to cover the entire deformation space such that each possible cage deformation can be decomposed into a linear combination of these bases. It is encouraged by our learning objective guided via the convex deformation loss $\mathcal{L}_C$ that minimizes the Chamfer Distance between each deformed convex and the target convex. The Chamfer Distance between two convexes is defined on 4096 points sampled from their surfaces.

At the same time, we wish deformation bases predicted by our network to have the following two properties: 1) Deformation bases should be relatively orthogonal to each other to avoid recording redundant deformation patterns and to cover independent deformation patterns. 2) Deformation bases should be in a low dimensional space, activating as few vertices as possible. Therefore, we further add an orthogonal loss $\mathcal{L}_{orth}$ and a sparse loss $\mathcal{L}_{sp}$ for regularization purpose, following [6]. Among them, $\mathcal{L}_{orth}$ penalizes "dot products" between different deformation bases. And $\mathcal{L}_{sp}$ penalizes the $l1$-norm of each deformation basis. Such two penalties are added as additional regularization to optimize the hierarchical deformation-based generative model together with $\mathcal{L}_C$ and the physical penalty term $\mathcal{L}_{phy}$.

A.2. Convex Deformation Synchronization

After learning the conditional generative model for each convex c, we further design a convex deformation synchronization strategy to compose all the individual convex deformation spaces to the whole mesh-level deformation space. Given a set of articulated object meshes $\mathcal{A}$ from a certain category and an articulated mesh $a \in \mathcal{A}$, assuming the mesh a is segmented into M convexes and each convex is equipped with a deformation model $g_{\mathcal{C}}(\mathbf{z}_{c_m}|c_m)$, our goal is to replace $\mathbf{z}_{c_m}$ with $S_{c_m}\mathbf{z}$ so that sampling the shared noise parameter $\mathbf{z}$ results in a globally consistent mesh deformation. To compute the synchronization transformation S_{c_m}, we consider the deformation from a to other articulated meshes $a^i \in \mathcal{A}$. In particular, for each a^i, we optimize for a set of deformation coefficients $\{\mathbf{y}_m^i\}$ so that each convex c_m in mesh a could deform into the corresponding convex c_m^i in mesh a^i following the deformation model $g_{\mathcal{C}}(\mathbf{z}_{c_m}|c_m, \mathbf{z}_{c_m} = \mathbf{y}_m^i)$. We can then estimate the synchronization transformations $\{S_{c_m}\}$ by solving the following optimization problem:

$$\underset{\{S_{c_m}\},\{\mathbf{z}^i\}}{\text{minimize}} \sum_{i=1}^{|\mathcal{A}|}\sum_{m=1}^{M} \|B_{c_m}S_{c_m}\mathbf{z}^i - B_{c_m}\mathbf{y}_m^i\|_2, \quad (1)$$

where B_{c_m} is the deformation bases of convex c_m and $\mathbf{z}^i$ is a global deformation coefficient from mesh a to a^i shared across all convexes. We solve the above optimization problem via alternatively optimizing the synchronization transformations $\{S_{c_m}\}$ and the global deformation coefficients $\{\mathbf{z}^i\}$.

Specifically, we optimize equation 1 by alternatively taking the following two steps:

- Fix $\{S_{c_m}\}$, optimize each global deformation coefficient $\mathbf{z}^i$ from a to a^i via the global deformation coefficients optimization algorithm 2. The algorithm takes the convex deformation bases $\{B_{c_m}\}$, current synchronization transformations $\{S_{c_m}\}$, and convex deformation coefficients $\{\mathbf{y}_m^i\}$ as input, and outputs the optimized $\mathbf{z}^i$.
- Fix $\{\mathbf{z}^i\}$, optimize each synchronization transformation S_{c_m} for each convex c_m via the synchronization transformation matrices optimization algorithm 1. It takes the convex deformation bases $\{B_{c_m}\}$, current global deformation coefficients $\{\mathbf{z}^i\}$, and convex deformation coefficients $\{\mathbf{y}_m^i\}$ as input, and output the optimized S_{c_m}.

By looping the above two optimization steps several times (*i.e.*, 100 in our implementation), we finally get the optimized synchronization transformation matrices $\{S_{c_m}\}$ and global deformation coefficient $\{\mathbf{z}^i\}$. Then the distribution of the global shape deformation coefficient $\mathbf{z}$ is modeled by a mixture of Gaussian fit to the optimized $\{\mathbf{z}^i\}$.

Please note that the above approach (Algorithm 1, 2) is an approximate solution and is not affected by $\{B_{c_m}\}$. However, $\{B_{c_m}\}$ indeed influence the optimization objective outlined in Eq. 2 in the main text and cannot be omitted.

Algorithm 1 Synchronization transformation matrices optimization.

Input: Deformation bases for each convex $\{B_{c_m}\}$. Global deformation coefficients $\{\mathbf{z}^i\}$ from a to other articulated meshes $\{a^i\}$. Deformation coefficients $\{\mathbf{y}_m^i\}$ from each convex c_m to the corresponding convex of the articulated mesh a^i.

Output: Synchronization transformation matrix S_{c_m} of the convex c_m.

```
1: $\mathbf{Z} \leftarrow$ Stack($\{\mathbf{z}^i\}$)
2: $\mathbf{Y}_m \leftarrow$ Stack($\{\mathbf{y}_m^i\}$)
3: $[\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}^T] \leftarrow$ SVD($\mathbf{Z}$)
4: $[\mathbf{U}_m, \mathbf{\Sigma}_m, \mathbf{V}_m^T] \leftarrow$ SVD($\mathbf{Y}_m$)
5: $S_{c_m} \leftarrow \mathbf{U}_m\mathbf{\Sigma}_m\mathbf{V}_m^T\mathbf{V}\mathbf{\Sigma}^+\mathbf{U}^T$
6: return $S_{c_m}$
```

Algorithm 2 Global deformation coefficients optimization. "lsq" denotes the least square solver.

```
Input: Deformation bases for each convex $\{B_{c_m}\}$. Synchronization transformations $\{S_{c_m}\}$. Deformation coefficients $\mathbf{y}^i_m$ from each convex $c_m$ to the corresponding convex of the articulated mesh $a^i$.
Output: Global deformation coefficients $\mathbf{z}^i$ from $a$ to $a^i$.
1: $\mathcal{S}_{\mathbf{z}^i} \leftarrow \emptyset$
2: for $m = 1$ to $M$ do
3:     $\hat{\mathbf{z}}^i_m \leftarrow$ lsq$(S_{c_m}, \mathbf{z}^i_m)$
4:     $\mathcal{S}_{\mathbf{z}^i} \leftarrow \mathcal{S}_{\mathbf{z}^i} \cup \{\hat{\mathbf{z}}^i_m\}$
5: $\mathbf{z}^i =$ Average$(\mathcal{S}_{\mathbf{z}^i})$
6: return $\mathbf{z}^i$
```

A.3. Physics-Aware Deformation Correction

In our method, we further add a physics-aware deformation correction scheme to 1) encourage the hierarchical deformation model to generate physically-realistic deformations and 2) optimize synthesized articulated meshes such that they can support correct articulation functions.

We leverage physical simulation and a collision response-based shape optimization strategy to realize this vision. Two losses are involved in the correction strategy: 1) a physical penalty term $\mathcal{L}_{phy}$ measuring self-penetrations and 2) a projection loss $\mathcal{L}_{proj}$ guiding how to project penetrated vertices to resolve observed penetrations.

We implement $\mathcal{L}_{phy}$ and $\mathcal{L}_{proj}$ manually with no simulators. Given an articulated mesh a, we illustrate the details of their computing process in Algorithms 3 4.

For each category, the K articulation states is formed by C independent articulation chains. Each articulation chain consists of a moving part p_{mov} and other static parts in their specific articulation states. p_{mov} is articulated through the whole articulation range when articulating the object by the articulation chain. However, the valid articulated states for resting parts may vary across different categories. For eyeglasses, when articulating one leg, the other one should be put into 0 degree or 90 degree.

Algorithm 3 Single simulation. Single-part articulation simulation losses. "NONE_MOTION" indicates fixed parts.

```
Input: Part mesh $p_{mov} = (\mathbf{V}_{mov}, \mathbf{E}_{mov})$ to articulate at the rest pose; Convex mesh $p_{ref} = (\mathbf{V}_{ref}, \mathbf{E}_{ref})$; The number of simulation steps for this moving part in the current convex $N_s$. Joint information set $\mathcal{J}$ of the moving part $p_{mov}$.
Output: Average penetration depth APD ($\mathcal{L}^{cur}_{phy}$) for this part $p_{mov}$ in the current context. Projection loss $\mathcal{L}^{cur}_{proj}$ for this part $p_{mov}$ in the current context.
1: if $\mathcal{J}$.moving_type == NONE_MOTION then
2: return 0 , 0
3: $\mathbf{N}_{ref} \leftarrow$ Face-Normal$(\mathbf{V}_{ref}, \mathbf{E}_{ref})$
4: $\mathbf{d}^0 \leftarrow$ Vertex-Face-Distance$(\mathbf{V}_{mov}, \mathbf{V}_{ref}, \mathbf{E}_{ref}, \mathbf{N}_{ref})$
5: $\mathbf{S}^0 \leftarrow$ Sign$(\mathbf{d}^0)$
6: $\mathcal{S}_{phy} \leftarrow \emptyset$
7: $\mathcal{S}_{proj} \leftarrow \emptyset$
8: Let $[l, u]$ be the articulation range of $p_{mov}$ which is contained in the joint information $\mathcal{J}$
9: for $t = 1$ to $N_s$ do
10:     $s^t \leftarrow l + (u - l) \cdot \frac{t}{N_s}$
11:     $p^t_{mov} = (\mathbf{V}^t_{mov}, \mathbf{E}_{mov}) \leftarrow$ Articulation-Simulation$(\mathbf{V}_{mov}, s^t, \mathcal{J})$
12:     $\mathbf{D}^t \leftarrow$ Vertex-Face-Distance$(\mathbf{V}^t_{mov}, \mathbf{V}_{ref}, \mathbf{E}_{ref}, \mathbf{N}_{ref})$
13:     $\mathbf{S}^t \leftarrow$ Sign$(\mathbf{D}^t)$
14:     $\mathbf{C}^t \leftarrow$ Vertices-In-Faces$(\mathbf{V}^t_{mov}, \mathbf{V}_{ref}, \mathbf{E}_{ref})$
15:     $\mathbf{C}^t \leftarrow \mathbf{C}^t \wedge (\mathbf{S}^t \neq \mathbf{S}^{t-1})$
16:     PeneD$(p^t_{mov}, p_{ref}) \leftarrow$ Mean$(\mathbf{C}^t \cdot \mathbf{D}^t \cdot \mathbf{S}^t, \text{dim} = 0, 1)$
17:     $\mathcal{S}_{phy} \leftarrow \mathcal{S}_{phy} \cup \{$PeneD$(p^t_{mov}, p_{ref})\}$
18:     $\Delta\mathbf{V}_{mov} \leftarrow \mathbf{V}^t_{mov} - \mathbf{V}^{t-1}_{mov}$
19:     ProjD$(p^t_{mov}, p_{ref}) \leftarrow$ Mean(Sum(Expand$(\Delta\mathbf{V}_{mov}, \text{dim} = 1)$ $\cdot$ Expand$(\mathbf{C}^t \cdot \mathbf{D}^t$, dim=2$)$ $\cdot$ Expand$(\mathbf{N}_{ref}, \text{dim} = 0), \text{dim} = 2), \text{dim} = 0, 1)$
20:     $\mathcal{S}_{proj} \leftarrow \mathcal{S}_{proj} \cup \{$ProjD$(p^t_{mov}, p_{ref})\}$
21:
22: $\mathcal{L}^{cur}_{phy} \leftarrow$ Average$(\mathcal{S}_{phy})$
23: $\mathcal{L}^{cur}_{proj} \leftarrow$ Average$(\mathcal{S}_{proj})$
24: return $\mathcal{L}^{cur}_{phy}$ , $\mathcal{L}^{cur}_{proj}$
```

Algorithm 4 Physics-aware losses.

```
Input: An articulated mesh $a$ with a set of moving parts $\{p_1, ..., p_{k_p}\}$ and their joint information $\{\mathcal{J}_1, ..., \mathcal{J}_{k_p}\}$, where $k_p$ is the number of parts in $a$. Number of simulation steps $N_s$ for a single part articulation simulation process, i.e., moving one part when other parts are put into a specific articulated state. The number of single-part articulation simulation processes $N_{det}$ for each part.
Output: Average penetration depth APD ($\mathcal{L}_{phy}$). Projection loss $\mathcal{L}_{proj}$.
1: $\mathcal{S}_{phy} \leftarrow \emptyset$
2: $\mathcal{S}_{proj} \leftarrow \emptyset$
3: for $p_{mov} = p_1$ to $p_{k_p}$ do
4:     for $i_{step} = 1$ to $N_{det}$ do
5:         Sample an articulation state for each part expect for $p_{mov}$: $\{s_p | p \in \{p_1, ..., p_{k_p}\}, p \neq p_{mov}\}$
6:         Put parts except for $p_{mov}$ into their sampled states; Put $p_{mov}$ into its rest articulation state.
7:         $p_{ref} \leftarrow$ Merge-Meshes$(\{p_1, ..., p_{k_p}\} \setminus \{p_{mov}\})$
8:         $\mathcal{L}^{cur}_{phy}, \mathcal{L}^{cur}_{proj} \leftarrow$ Single-Simulation$(p_{mov}, p_{ref}, N_s, \mathcal{J}_{p_{mov}})$
9:         $\mathcal{S}_{phy} \leftarrow \mathcal{S}_{phy} \cup \{\mathcal{L}^{cur}_{phy}\}$
10:        $\mathcal{S}_{proj} \leftarrow \mathcal{S}_{proj} \cup \{\mathcal{L}^{cur}_{proj}\}$
11:
12: $\mathcal{L}_{phy} \leftarrow$ Average$(\mathcal{S}_{phy})$
13: $\mathcal{L}_{proj} \leftarrow$ Average$(\mathcal{S}_{proj})$
14: return $\mathcal{L}_{phy}$ , $\mathcal{L}_{proj}$
```

How to use $\mathcal{L}_{proj}$ to optimize the articulated mesh a? Since the calculation process of $\mathcal{L}_{proj}$ is differentiable, we can update shape a by back-propagating $\mathcal{L}_{proj}$ to update the global shape deformation parameter $\mathbf{z}$. Specifically, we calculate the gradient of $\mathcal{L}_{proj}$ over $\mathbf{z}$, which can be easily realized by the support of PyTorch's "autograd" package, and then update $\mathbf{z}$ via the gradient, *i.e.,* $\mathbf{z} \leftarrow \mathbf{z} - \epsilon_{proj} \frac{\partial \mathcal{L}_{proj}}{\partial \mathbf{z}}$. ϵ_{proj} serves as the "learning rate" for the deformation coefficient $\mathbf{z}$ here and is set to 10^{-4} at the training time and 10^{-5} at the inference/sampling time in our implementation.

A.4. Additional Explanations

Mesh smooth layer. Directly composing deformed convexes together for object-level meshes would usually lead to unwanted artifacts near convex edges, as shown in each middle one of every three shapes in Figure 1. To tackle this issue and to produce smooth object-level meshes, we add a smooth layer following [3], resulting in the rightmost shape in every three shapes drawn in Figure 1.

B. Additional Experiments

B.1. Few-Shot Mesh Generation

We further discuss our few-shot mesh generation performance from the following aspects to validate the merits of our hierarchical mesh deformation-based generative scheme:

- Few-shot generation performance w.r.t. the number of observed reference examples (#Shots);
- Few-shot generation performance on rigid categories;
- Additional results on human bodies.

Few-shot generation performance w.r.t. #Shots. For Scissors, Eyeglasses, and TrashCan containing relatively rich objects, we try to vary the value of the number of observed examples (#Shots) and compare the performance achieved by different methods on each few-shot setting. We consider three additional settings, namely 2-shots, 4-shots, and 8-shots. To make results comparable across different few-shot settings, we use the **same test set** for all of those settings. From Table 1, 2, 3, we can observe that our method can consistently outperform baseline strategies. It successfully guides the model to generate diverse samples even under the 2-shots setting, bypassing baselines by a large margin, *e.g.,* 189% relatively higher coverage ratio than DeepMetaHandles on the Eyeglasses category. Notice that it is not at a cost of sacrificing the mesh quality, *i.e.,* we can achieve the lowest MMD scores at the same time on all of those three categories.

Further, as a general empirical rule observed from the results, more observed examples would always lead to better few-shot generation performance, guiding the model to generate visually plausible samples with higher diversity. It aligns well with our intuitions and also conclusions made in few-shot image generation works [5, 4].

Few-shot generation performance on rigid mesh categories. To further prove the effectiveness of our hierarchical mesh deformation-based generative strategy as a general few-shot generation method not restricted to articulated objects, we test it on four rigid categories from the ShapeNet dataset [1] (*i.e.*, Table, Chair, Lamp, and Airplane) and compare it to the baselines. Categories for pretraining convex-level deformation models of Table, Chair, Lamp, and Airplane are Chair, Table, Airplane, and Lamp respectively. From Table 4, our method can still achieve better performance on all of those four categories. It further demonstrates the superiority of our hierarchical mesh deformation strategy as a general few-shot generation method over previous methods. We also report the results achieved by our method on the other two few-shot learning settings. We can make similar observations by comparing across different few-shot learning settings. As a qualitative evaluation, we draw samples from our model for the above four categories in Figure 2 (under the 4-shot setting).

Deformation results on human bodies Ours approach can indeed generalize to complex shape such as human bodies as exampled in Figure 3. We mainly leverage this example to show the method is not restricted to relatively simple piece-wise rigid objects demonstrated in main experiments. However, we do not conduct abundant experiments on it since human bodies are deformable and have rich data sources, which diverge from our focus on piece-wise rigid objects with limited examples.

B.2. Transfer Learning and Fine-tuning for Convex Deformations

We examine the role of transfer learning and fine-tuning in our method and find that 1) **Transfer learning**'s power can be boosted by increasing the amount of source data and is affected by the affinity between source and target categories; 2) **Fine-tuning** can help learn category-specific deformation patterns, benefiting quality and diversity.

Transfer learning w.r.t. amount of data for transferring. Table 6 compares the model's performance when a) using all source data for transferring, b) using half amount of total source data, and c) use no transfer learning. The transfer learning's effectiveness can be boosted increasing the amount of source data for transferring.

Transfer learning w.r.t. source categories. To test the influence of source categories for deformation pattern transferring on the few-shot generation performance, we try to vary the source categories by using each category individually as the source and comparing them. We conduct experiments on three relatively rich articulated mesh categories,

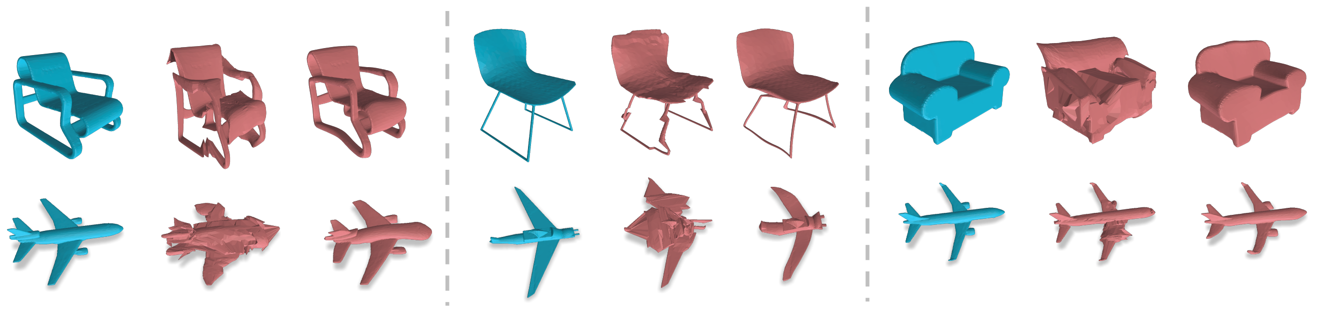

Figure 1. **The effectiveness of the mesh smooth layer.** For every three shapes, the **leftmost** one is the reference shape for deformation, the **middle** one is the generated mesh without smoothing, and the **right** one is the shape after smoothing.

Table 1. **Experimental comparisons on Eyeglasses category.** MMD is multiplied by 10^3. **Bold** numbers for best values.

#Shots	Method	MMD (↓)	COV (%, ↑)	1-NNA (%, ↓)	JSD (↓)
2	PolyGen [8]	9.558	3.18	99.70	0.1543
	DeepMetaHandles [6]	8.684	6.67	99.21	0.1585
	Ours	**7.279**	**19.30**	**98.41**	**0.0903**
4	PolyGen [8]	8.669	8.28	98.42	0.1425
	DeepMetaHandles [6]	6.685	12.57	**98.50**	0.1036
	Ours	**6.303**	**27.54**	98.80	**0.0840**
8	PolyGen [8]	7.663	15.56	97.24	0.1054
	DeepMetaHandles [6]	6.222	17.33	97.99	0.0864
	Ours	**6.102**	**34.56**	**97.25**	**0.0799**

namely Eyeglasses, Scissors, and TrashCan. From Table 5, we can make the following observations: 1) For Scissors, Lamp is a friendly category to reduce the minimum matching distance for samples of higher quality. 2) For TrashCan, Lamp as the source category can help with enhancing the diversity of generated samples, perhaps due to diverse deformation patterns transferred to the TrashCan's body parts.

Fine-tuning. The effectiveness of the fine-tuning process can be examined by comparing the ablated version and the full model demonstrated in Table 6.

Intermediates for transferring deformation patterns. We choose to use convexes as intermediates to learn and transfer shared deformation patterns in this work. It comes from the assumption and the observation that the convex distribution is more similar across different categories than the whole shape distribution (as shown in Figure 4). Therefore, from our intuition, convexes are more promising to serve as intermediates for transferring mesh deformation patterns across different categories. We further validate this intuition by trying to learn and transfer deformation patterns at the object level. As shown in Table 7, our model using convexes as intermediates can perform much better than the trail on transferring at the object level. This validates one of our crucial assumptions in this work.

Besides, we cannot even observe trivially transferring deformation patterns at the object level as a better strategy than the method without any deformation transferring. Usually, transferring deformation rules for the whole object requires some special designs [10]. This confirms the value of our hierarchical deformation design for mesh deformation transferring.

Shape retrieval experiments for comparing quality and diversity. As a further intuitive demonstration on the effectiveness of our techniques (*i.e.,* transfer learning, fine-tuning, and convex as intermediates) on improving the results' diversity and quality transfer learning, we conduct a shape retrieval experiment and compare our model with different ablated versions. Given each target shape, we select the its closest shape from generated assets as the retrieval results. The results are presented in Figure 5. Only ours can give results that are plausible, closet to the target, while also different from each other.

B.3. Generation via Deformation

There are a variety of generation techniques that have been developed recently in the tide of AIGC, such as score-based generative models and diffusion models. Along with them, many works try to explore the possibility of leveraging such techniques for 3D content generation. They mostly aim at generating shapes as a whole without considering their functionalities.

In this work, we wish to generate physically-realistic ar-

Table 2. **Experimental comparisons on Scissors category.** MMD is multiplied by 10^3. **Bold** numbers for best values.

#Shots	Method	MMD (↓)	COV (%, ↑)	1-NNA (%, ↓)	JSD (↓)
2	PolyGen [8]	7.311	4.55	99.65	0.5192
	DeepMetaHandles [6]	6.154	12.19	98.39	0.3315
	Ours	**2.503**	**26.19**	**98.31**	**0.1412**
4	PolyGen [8]	4.015	9.52	98.96	0.3459
	DeepMetaHandles [6]	1.875	24.20	98.76	0.2173
	Ours	**1.534**	**50.45**	**97.81**	**0.1299**
8	PolyGen [8]	3.108	13.16	97.91	0.2067
	DeepMetaHandles [6]	1.747	32.19	96.76	0.2017
	Ours	**1.184**	**63.47**	**96.12**	**0.1256**

Table 3. **Experimental comparisons on TrashCan category.** MMD is multiplied by 10^3. **Bold** numbers for best values.

#Shots	Method	MMD (↓)	COV (%, ↑)	1-NNA (%, ↓)	JSD (↓)
2	PolyGen [8]	16.448	4.29	97.44	0.3224
	DeepMetaHandles [6]	14.589	6.33	90.24	0.2170
	Ours	**12.746**	**9.00**	**84.81**	**0.1436**
4	PolyGen [8]	10.218	13.22	93.09	0.2331
	DeepMetaHandles [6]	9.402	**17.14**	86.49	0.1913
	Ours	**8.499**	**17.14**	**73.11**	**0.1003**
8	PolyGen [8]	9.001	16.67	90.79	0.1938
	DeepMetaHandles [6]	8.970	22.83	84.44	0.1379
	Ours	**8.134**	**24.47**	**71.02**	**0.0935**

ticulated meshes and resort to a relatively traditional generation technique, *i.e.,* generation via deformation. The reasons of our choice are mainly as follows: **1)** Deformation-based generation could let us parameterize shape variations into a low-dimensional space, by leveraging cages to control deformations and by using deformation bases to record common deformation patterns. **2)** Deformation-based generation could easily let us find suitable intermediates such that shared deformation patterns can be transferred across different categories easily. **3)** It also enables us to devise an effective physics-aware correction scheme to guide the model to generate physically-realistic deformations.

To explore more possibilities, we try to design another method that leverages point cloud diffusion, surface reconstruction, and a physics-aware correction designed for the reverse diffusion process, supported by the differentiable surface reconstruction algorithm. However, sometimes we observe that it tends to produce meshes of poor quality. We suppose that it comes from the difficulty to impose physics-related constraints on the generated meshes from point cloud diffusion.

We would elaborate on details of this method and its results in the following text.

Method: Hierarchical generation via point cloud diffusion. It proceeds as follows: 1) Represent the shape via an object-convex hierarchy via convex decomposition. 2) Train a convex-level conditional point cloud diffusion model on source categories. 3) Transfer the pre-trained convex-level point cloud diffusion model to the target category via fine-tuning. 4) Compose convex point clouds for object point clouds. 5) Reconstruct the mesh surface for part-level point clouds via a pre-trained differential poison solver model (an SAP model) [9] which is further fine-tuned on the target category. 6) Compose parts together for the final articulated mesh with a physics-aware correction scheme.

Existing problems. Despite the flexible generation ability and high diversity of the sampled shapes, this strategy suffers from the poor articulated mesh quality when we further add a physics-aware correction scheme on top of reconstructed part surfaces, as shown in Figure 6.

Discussions. Generating physically-realistic articulated meshes are usually challenged by the few-shot difficulty, mesh quality, and the physically-realistic expectations. We explore a mesh deformation-based physics-aware generation strategy in this work by transferring deformation patterns from large categories and further with a physics-aware correction scheme that can improve the physical validity of generated samples while at the same time preserving transferred knowledge.

As for two key designs in our work, transferring cross-

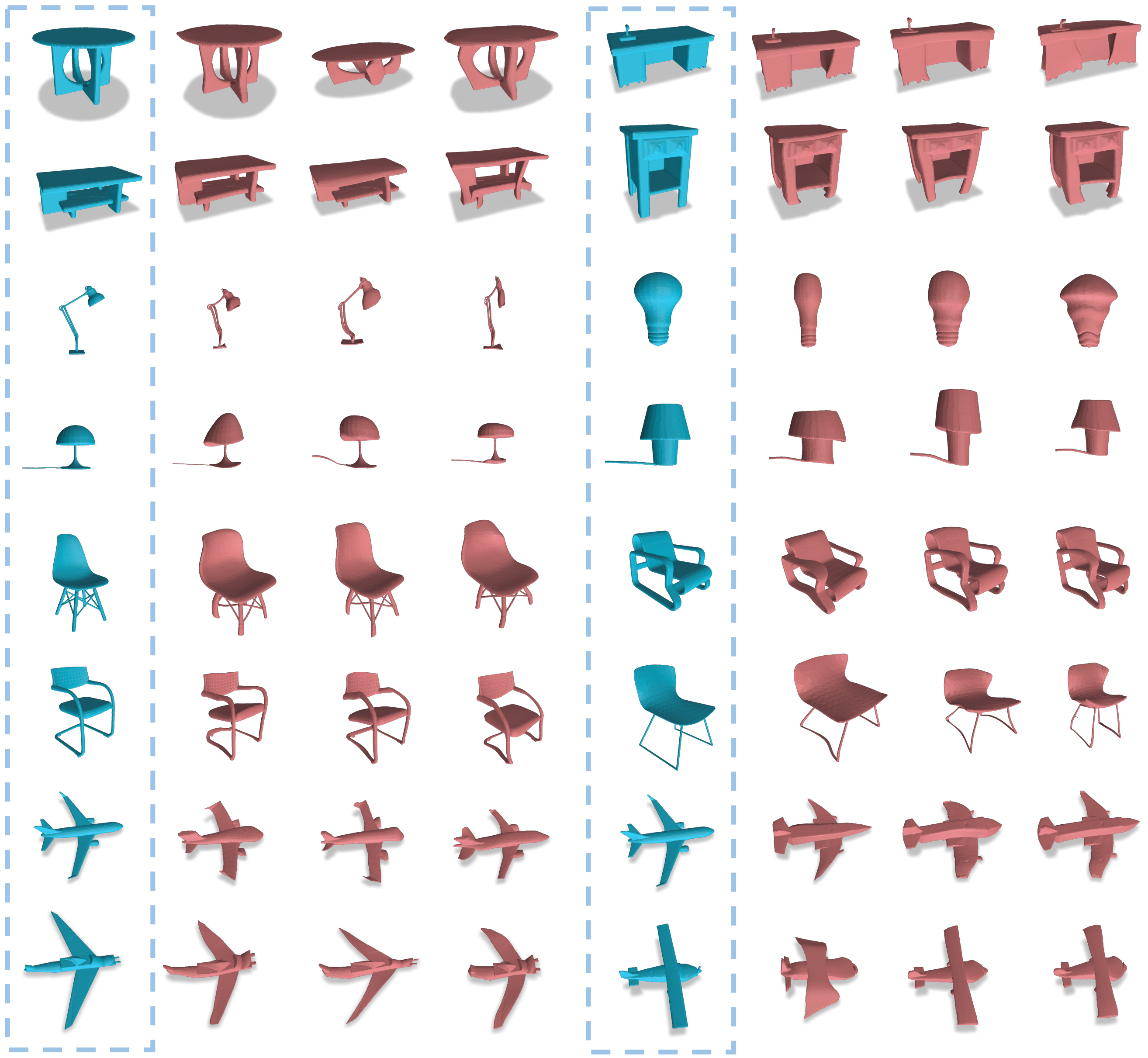

Figure 2. **Qualitative evaluation on few-shot rigid mesh generation.** For every four shapes, the leftmost one (highlighted by blue rectangles) is the reference shape from the training set, while the remaining three are conditionally generated samples. Object categories from top to down are Table, Lamp, Chair, and Airplane.

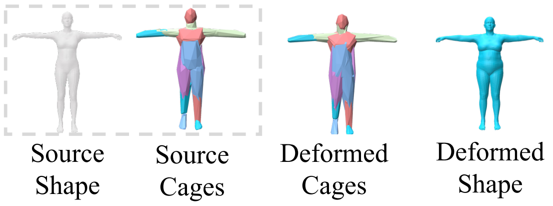

Figure 3. **Additional deformation results on human bodies.**

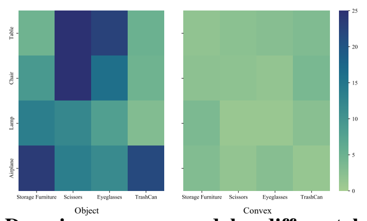

Figure 4. **Domain gap measured by different levels of shapes.** Heatmaps of minimum matching distances between pre-training and target datasets.

category shared shape patterns at the convex level is a relatively general idea that can also be adapted to other gen-

Table 4. **Experimental comparisons on rigid mesh categories.** MMD is multiplied by 10^3. **Bold** numbers for best values.

Category	#Shots	Method	MMD (↓)	COV (%, ↑)	1-NNA (%, ↓)	JSD (↓)
Table	2	Ours	6.162	8.20	98.87	0.1118
	4	Ours	4.206	13.63	97.80	0.0763
	8	DeepMetaHandles [6]	6.640	7.45	99.34	0.1046
		Ours	**2.356**	**25.23**	**96.33**	**0.0479**
Chair	2	Ours	4.148	15.20	99.42	0.1080
	4	Ours	3.363	19.36	98.63	0.0512
	8	DeepMetaHandles [6]	5.205	12.70	99.43	0.0738
		Ours	**2.690**	**28.43**	**97.09**	**0.0294**
Lamp	2	Ours	4.029	16.80	97.17	0.0930
	4	Ours	3.556	20.70	96.13	0.0749
	8	DeepMetaHandles [6]	9.730	11.40	99.75	0.1966
		Ours	**2.822**	**30.20**	**93.67**	**0.0671**
Airplane	2	Ours	1.029	15.93	97.47	0.0906
	4	Ours	0.927	21.15	97.33	0.0401
	8	DeepMetaHandles [6]	2.295	9.37	99.31	0.1654
		Ours	**0.869**	**25.71**	**97.08**	**0.0334**

Table 5. **Few-shot generation performance w.r.t. source categories for transfer learning** on Eyeglasses, Scissors, and TrashCan categories. "All" denotes using all of those four categories as the source category.

Target Category	Source Category	MMD (↓)	COV (%, ↑)	1-NNA (%, ↓)	JSD (↓)
Eyeglasses	Table	6.07	25.00	99.13	0.0875
	Chair	8.23	20.53	99.57	0.1188
	Lamp	6.47	26.67	98.46	0.0973
	Airplane	8.28	21.91	99.35	0.1042
	All	**6.06**	**29.82**	**98.26**	**0.0681**
Scissors	Table	1.63	51.07	97.55	0.1579
	Chair	1.57	56.76	97.10	0.1752
	Lamp	**1.34**	54.39	97.79	0.1328
	Airplane	1.67	52.19	97.36	0.1724
	All	1.50	**57.89**	**97.02**	**0.1274**
TrashCan	Table	8.43	17.07	72.87	**0.0933**
	Chair	**8.39**	17.07	74.29	0.0939
	Lamp	9.49	**19.29**	73.58	0.1138
	Airplane	13.20	10.32	74.29	0.1429
	All	8.43	17.14	**72.09**	0.0994

eration techniques for shape space enrichment. However, further imposing physical validity on top of the generated samples is not a trivial thing. Our physics-aware correction works well for the deformation-based generation. But what is the most ideal strategy that can be combined with other generation techniques naturally worth further exploring.

C. Experimental Settings

C.1. Datasets

Rigid mesh datasets for pre-training. For pre-training data from rigid object categories, we select 9947 instances from ShapeNet [1] dataset, including Table, Chair, Lamp, and Airplane. We list the number of instances in each category in Table 9.

Articulated mesh datasets for evaluation. For articulated object datasets, we select six articulated object cat-

Table 6. Ablations w.r.t. the effectiveness of pre-training and fine-tuning.

Method	MMD (↓)	COV (%, ↑)	1-NNA (%, ↓)	JSD (↓)	APD (↓)
Ours w/o Transfer	5.424	46.64	93.01	0.1159	1.3822
Ours w/ Transfer (Half Data)	5.201	49.43	92.81	0.1130	1.3365
Ours w/o Fine-tuning	6.538	43.20	94.70	0.1437	1.4530
Ours	**5.198**	**50.45**	**92.36**	**0.1118**	**1.3192**

Table 7. **Comparison between methods using the transfer learning strategy at the object level and the convex level.** For metrics of each version, we report their average value over all categories. MMD is multiplied by 10^3 and APD is multiplied by 10^2. **Bold** numbers for best values.

Method	MMD (↓)	COV (%, ↑)	1-NNA (%, ↓)	JSD (↓)
Ours (Transfer Obj.)	7.339	32.70	96.11	0.1557
Ours w/o Hier.	7.170	36.41	95.43	0.1492
Ours	**5.198**	**50.45**	**92.36**	**0.1118**

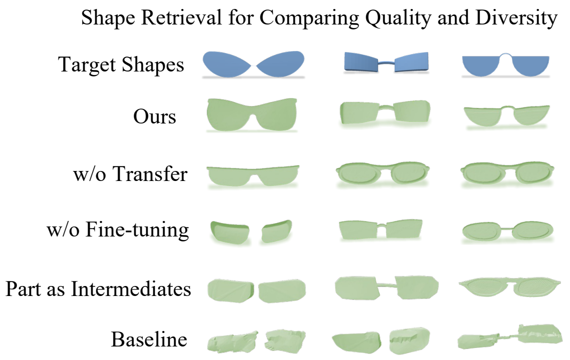

Figure 5. **Shape retrieval for comparing quality and diversity.**

egories from [11] and four rigid object categories from ShapeNet [1]. We list the number of its instances in Table 8.

C.2. Baselines.

We compare our method with two typical mesh generative strategies, namely PolyGen [8] that falls into the genre of direct surface generation strategies, and DeepMetaHandles [6] that leverages mesh deformation for generation. To further adapt them to our articulated mesh generation scenario, we make further modifications to their original algorithms. We briefly summarize our implementations of such two methods as follows.

For PolyGen [8], we download the official TensorFlow implementation. We then design a part-by-part generation strategy to leverage it for articulated mesh generation. We define a part order $(\mathcal{P}_1, ..., \mathcal{P}_k)$, where k is the number of parts, and generate joints of each part via generating two joint points (if any, otherwise generating no points, equivalent to generating an empty joint point set), *i.e.,* $\mathcal{J}_i (1 \leq i \leq k)$. Therefore, we generate mesh vertices, mesh surfaces, and joint points via the following order: $[(\mathcal{V}_1, \mathcal{F}_1, \mathcal{J}_1), ..., (\mathcal{V}_k, \mathcal{F}_k, \mathcal{J}_k)]$.

For DeepMetaHandles [6], we use the official implementation. To leverage it for articulated mesh generation, we train a deformation-based generative model for each part individually. Then, an object is generated by generating its part. Directly using the above strategy yields the default version of our compared DeepMetaHandles ("DeepMetaHandles"). We could further add a physics-aware deformation correction scheme on top of it, leading to the improved version (denoted as "DeepMetaHandles w/ Phy.").

C.3. Metrics.

For generative model-related metrics, we follow the computing processes adopted in [7] and evaluate corresponding values on 4096 points sampled from mesh surfaces. Average Penetration Depth evaluates the average per-vertex penetration depth further averaged over all articulation states. Its computing process is the same as that of $\mathcal{L}_{phy}$ (see Algorithms 3 4 for details).

C.4. Additional Implementation Details

Convex decomposition. We use BSP-Net [2] to provide intra-category consistent shape co-segmentation for each category. It's worth further mentioning that the number of convexes that we set for BSP-Net performs only as the upper bound of the number it uses for decomposition. For rigid categories, we set the number of convexes to 256. For articulated objects, we list their settings in Table 10. For each part, we first try the 128 convexes setting and double the number of convexes by 2 if BSP-Net's training loss cannot converge.

For a visual understanding of the convex decomposition, we draw some examples of decomposed convexes of instances from the Table category in Figure 7.

Convex-level generative model. For the convex-level generative model, we use a sphere mesh containing 42 vertices

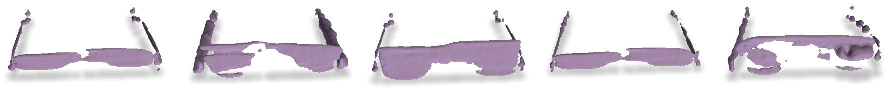
Figure 6. **Examples of articulated meshes generated by the point cloud diffusion-based strategy** (Category: Eyeglasses).

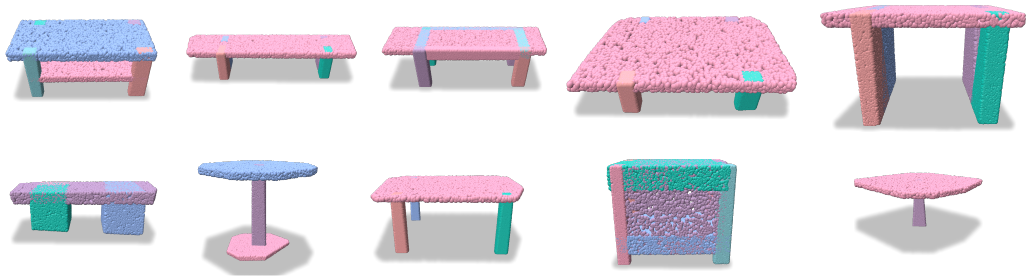
Figure 7. **Examples for decomposed convexes** (converted to point clouds) of instances for the Table category. Different colors represent different convexes.

and 80 faces [12] as the template of cages. The number of deformation bases is set to 16.

We use a neural network $\psi_\theta(\cdot)$ to parameterize deformation bases. It takes a convex c as input and predicts its deformation bases $B_c = \psi_\theta(c)$. It first extracts the convex feature for c via a PointNet encoder (applied on 4096 points sampled from c's surface). Then we feed the convex feature and the cage t_c of the convex c to a MultiFold network, same as the network used in [12] for per-point features with the bottleneck size set to 512 and the number of folds set to 3. We then use an MLP for basis prediction with weight size in each layer set to (128, 128), (128, 128), and (128, 48) with LeakyReLU layer using the default α value between every two fully-connected layers.

Training protocols. In the pre-training stage, total losses for optimizing the convex-level deformation-based generative model is composed of the convex deformation loss $\mathcal{L}_C$ and two penalty terms, *i.e.,* $\mathcal{L}_{sp}$ and $\mathcal{L}_{orth}$. Specifically, the total loss is $\mathcal{L} = \mathcal{L}_C + \lambda_{sp} \cdot \mathcal{L}_{sp} + \lambda_{orth} \cdot \mathcal{L}_{orth}$, where λ_{sp} and λ_{orth} are both set to 10^{-4} in our implementation.

In the fine-tuning stage, the object-level physical penalty $\mathcal{L}_{phy}$ (calculated on the final shape optimized via $\mathcal{L}_{proj}$) is further added to encourage the network to produce physically-realistic deformations. Therefore, the total loss becomes $\mathcal{L} = \mathcal{L}_C + \lambda_{phy} \cdot \mathcal{L}_{phy} + \lambda_{sp} \cdot \mathcal{L}_{sp} + \lambda_{orth} \cdot \mathcal{L}_{orth}$. We empirically set λ_{phy} to 1.0.

We use the Adam optimizer with the momentum set to (0.9, 0.999) for optimization in both the pre-training and the fine-tuning stages. The initial learning rate is set to 10^{-4}. And it decayed by 0.5 after every 100 epochs.

Evaluation protocols. We adopt the few-shot generation evaluation strategy. The default number of shots is set to 5. We use the same test set to compare different methods. Specifically, for each articulated mesh category, we randomly select 5 instances for training while using the remaining instances for test. At the test time, such 5 instances serve as reference examples to generate new samples. For each reference example, we generate 40 samples from it. It is realized by passing the example mesh into the hierarchical deformation generative model and randomly sampling 40 global deformation coefficients $\mathbf{z}$ from its deformation coefficient distribution model. Each global deformation coefficient $\mathbf{z}$ together with the synchronized deformation bases $\{S_c B_c\}$ are used to deform the example to a new shape.

As for the 2-shot, 4-shot, and 8-shot settings present in the supplementary material, we use the same strategy to split instances in the articulated mesh category into a few-shot training set and a test set.

Table 8. **Number of instances in each articulated object category.**

Method	Storage Furniture	Scissors	Eyeglasses	Oven	Lamp	TrashCan
#Instances	31	46	65	10	13	43

Table 9. **The number of instances in each rigid object category.**

Method	Table	Chair	Lamp	Airplane
#Instances	3000	2447	1500	3000

Table 10. **The number of convexes used for convex decomposition** for each kind of part of each articulated object category.

Method	Storage Furniture	Scissors	Eyeglasses	Oven	Lamp	TrashCan
Link 0	128	128	128	128	128	128
Link 1	512	128	128	512	128	512
Link 2	512	-	128	-	128	-
Link 3	-	-	-	-	128	-

References

[1] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 4, 8, 9

[2] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. Bsp-net: Generating compact meshes via binary space partitioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 45–54, 2020. 9

[3] Francisco González García, Teresa Paradinas, Narcis Coll, and Gustavo Patow. * cages: a multilevel, multi-cage-based system for mesh deformation. *ACM Transactions on Graphics (TOG)*, 32(3):1–13, 2013. 4

[4] Zheng Gu, Wenbin Li, Jing Huo, Lei Wang, and Yang Gao. Lofgan: Fusing local representations for few-shot image generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8463–8471, 2021. 4

[5] Yan Hong, Li Niu, Jianfu Zhang, and Liqing Zhang. Deltagan: Towards diverse few-shot image generation with sample-specific delta. In *European Conference on Computer Vision*, pages 259–276. Springer, 2022. 4

[6] Minghua Liu, Minhyuk Sung, Radomir Mech, and Hao Su. Deepmetahandles: Learning deformation meta-handles of 3d meshes with biharmonic coordinates. *arXiv preprint arXiv:2102.09105*, 2021. 2, 5, 6, 8, 9

[7] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2837–2845, 2021. 9

[8] Charlie Nash, Yaroslav Ganin, SM Ali Eslami, and Peter Battaglia. Polygen: An autoregressive generative model of 3d meshes. In *International conference on machine learning*, pages 7220–7229. PMLR, 2020. 5, 6, 9

[9] Songyou Peng, Chiyu Jiang, Yiyi Liao, Michael Niemeyer, Marc Pollefeys, and Andreas Geiger. Shape as points: A differentiable poisson solver. *Advances in Neural Information Processing Systems*, 34:13032–13044, 2021. 6

[10] Minhyuk Sung, Zhenyu Jiang, Panos Achlioptas, Niloy J Mitra, and Leonidas J Guibas. Deformsyncnet: Deformation transfer via synchronized shape deformation spaces. *arXiv preprint arXiv:2009.01456*, 2020. 5

[11] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11097–11107, 2020. 9

[12] Wang Yifan, Noam Aigerman, Vladimir G Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. Neural cages for detail-preserving 3d deformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 75–83, 2020. 10