DiscoNet: Shapes Learning on Disconnected Manifolds for 3D Editing

Éloi Mehr[1], Ariane Jourdan , Nicolas Thome[1], Matthieu Cord[1], and Vincent Guitteny

[1]LIP6, Sorbonne Université

Abstract

Editing 3D models is a very challenging task, as it requires complex interactions with the 3D shape to reach the targeted design, while preserving the global consistency and plausibility of the shape. In this work, we present an intelligent and user-friendly 3D editing tool, where the edited model is constrained to lie onto a learned manifold of realistic shapes. Due to the topological variability of real 3D models, they often lie close to a disconnected manifold, which cannot be learned with a common learning algorithm. Therefore, our tool is based on a new deep learning model, DiscoNet, which extends 3D surface autoencoders in two ways. Firstly, our deep learning model uses several autoencoders to automatically learn each connected component of a disconnected manifold, without any supervision. Secondly, each autoencoder infers the output 3D surface by deforming a pre-learned 3D template specific to each connected component. Both advances translate into improved 3D synthesis, thus enhancing the quality of our 3D editing tool.

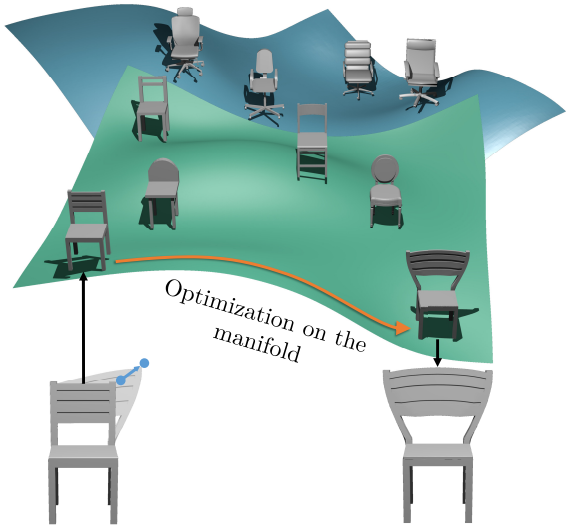


Figure 1: Illustration of a disconnected manifold of 3D chairs, where a chair is optimized on the connected component it lies on to fit a user deformation and generate the expected realistic design.

1. Introduction

While 3D models become increasingly used as virtual and augmented reality experiences expand, the creation of 3D contents still requires advanced skills in 3D modeling, and remains cumbersome even for experts. Free-form deformation algorithms comprise a natural approach to broaden the audience of 3D design softwares. The user starts from a mesh, selected in a 3D database to be close to his intended design. He can then deform this mesh towards his objective using high-level 3D editing tools, such as cage-based deformations [40], positional handles [43], or sketch-based deformations [51]. Even though these tools simplify the 3D modeling process, they still target an experienced user to achieve a realistic result. Indeed, a slight mistake during the editing can make the 3D model completely implausible, as the user is neither guided nor constrained.

The same problematic also arises in photo retouching. Deep learning has proven to be a powerful technique for the development of user-friendly and realistic 2D editing tools [6, 52]. These algorithms leverage real datasets to learn to generate plausible photos, and use this knowledge to retouch a photo in a plausible way from the user's intention. A manifold of real photos is learned, and the original photo is optimized in the latent space of the manifold to meet the user's constraints, therefore reaching the expected design while ensuring the global plausibility of the photo.

Although significant progress has been made in 3D shapes learning, especially with the recent introduction of autoencoders dedicated to 3D [17, 46], two issues remain to devise an efficient learning-guided 3D editing pipeline. Firstly, 3D shapes do not generally lie on a connected manifold, even when they belong to the same category. For example, there is no continuous path which interpolates a desk chair with wheels into a chair with four legs while always generating plausible chairs along the path. The chairs manifold is actually disconnected, each connected component corresponding to a different typology of chairs, as illustrated in Figure 1.

However, a single generator [16] or decoder [21, 26] cannot learn a disconnected manifold without generating implausible models, as proved by our Theorem 1. Secondly, to optimize a 3D model in the latent space and infer a realistic deformation, it is preferable to preserve the correspondences between the vertices of the synthesized shapes.

In this work, we introduce DiscoNet, a new unsupervised deep learning model which solves both previous issues. DiscoNet takes advantage of several 3D autoencoders to learn a disconnected manifold of 3D shapes, where each 3D autoencoder provides a parameterization of one connected component. Moreover, each 3D decoder acts as a residual block [20] by deforming a pre-learned 3D template representing the mean shape of its connected component, rather than generating the whole shape from the ground up. We also design a specific loss scheme to competitively learn the autoencoders and their templates. DiscoNet largely improves the geometric accuracy and the topological consistency of the generated models, as we demonstrate quantitatively and qualitatively on the well-known ShapeNet dataset [10]. We finally exploit DiscoNet to build a learning-guided 3D editing pipeline, based on a simple user interface allowing to easily and realistically reshape a 3D mesh with 3D handles or 2D sketches. Our editing tool leverages the learned manifold to infer a realistic morphing of the original 3D model, which fits the local modifications enforced by the user.

2. Related work

Manifold learning. Autoencoders [21] learn to represent a dataset in a low-dimensional latent space, which is typically possible if the samples lie close to a low-dimensional manifold, *e.g*. if they belong to the same category. [26, 41] extend the autoencoder into a generative model, which allows to sample the latent space such that the decoder follows the data distribution. Another popular generative model is the generative adversarial network (GAN) [16], which does not require an explicit minimization of the reconstruction error or a likelihood maximization, unlike the vanilla or variational autoencoders. An important topic in manifold learning is the disentangling of the latent space, which aims at learning a latent space where each dimension represents an independent factor of variation. Disentangling can be bolstered in variational autoencoders [8] as well as GAN [11]. Nevertheless, such methods are unable to factor dimensions to represent different connected components, as the latent space still remains continuous. Similarly to our approach, [25] solves this issue using several distinct decoders to learn each connected component of a disconnected manifold. They apply a variational method to maximize the mutual information between the output and the decoder which it came from. On the contrary, our method is not probabilistic, and thus allows to pre-learn a 3D template for each connected component. Finally, [22] jointly learns several decoders by minimizing a weighted loss of all the decoders, the weighting terms being also inferred by the network to favor the best decoder for the given input. Our method does not rely on any weighting scheme, but uses a hard thresholding instead, which guarantees the selection of the best autoencoder.

Shapes learning. As reviewed in [45], morphable models [2, 5, 9] are a foundational 3D shapes learning technique. They define a low-dimensional parameterization of a base shape, usually learned with a principal component analysis. Nevertheless, a major drawback is the need of shapes correspondences, or at a least a partial shapes matching. [15, 50] solve this issue by embedding the shapes in a coarse 3D occupancy voxels grid, in addition to gaining all the CNN efficiency with an autoencoder or a GAN. [19, 44] extend the 3D CNN to high resolutions using octrees. Unfortunately, although voxels-based 3D representations remove the need of 3D correspondences for the learning, they remain a poor surrogate to handle 3D shapes, which are inherently surfacic rather than volumic data. [38] circumvents the issue by learning the continuous truncated signed distance function, which is theoretically infinitely accurate and can easily handle different shape topologies. Nonetheless, computing a signed distance function requires a watertight 3D mesh, which is a very strong prerequisite, and an onerous isosurface extraction is also needed to convert it back to a 3D model. [39] introduces an encoder of 3D point clouds, invariant to the ordering of the points, whereas [13] presents a point cloud decoder, learned with a 3D distance invariant to the ordering of the predicted point cloud. Both ideas can be jointly used to provide a point cloud autoencoder or GAN as demonstrated in [1]. A further step consists in learning an autoencoder of 3D meshes, *i.e*. point clouds with topology, using a decoder which acts as a parameterization of a 2-manifold, as shown in [17, 46]. A close idea is also exploited in [27, 31], but they mostly focus on extending graph neural networks ([34, 35]) to take directly into account the local topology of the mesh. Our work builds on the 3D surface autoencoder [17, 46], which is particularly suitable in an editing system as its output can be used straightforwardly.

Intelligent editing. A lot of works already exist to provide 3D editing algorithms which are not data-driven, but analyze solely the geometry to edit. For example, [14] computes intelligent wires that the user can edit to deform the shape, and [18] proposes an automatic selection algorithm that eases the subsequent editing. However, it is natural to leverage a 3D database, if available, in order to guide the editing process. Thus, [23] shows how to segment a 3D dataset to recombine parts and synthesize new 3D models, whereas [30] achieves a similar goal with a deep recursive autoencoder which smoothly handles interpolations even between different assembly structures. The editing can also act on the geometry itself, rather than the shape structure, as demonstrated in [48, 49]. They exploit a labeled 3D dataset

to provide an editing tool driven by high-level semantic attributes. As done in our paper, manifold learning can be used to leverage a latent space representing a plausible manifold of the input space. [32] uses such an approach to realistically deform voxelized shapes, by steadily encoding and decoding the edited model via the latent space. Of course, the final decoding inevitably belongs to the learned manifold, which cannot represent all realistic shapes. [6, 52] solve this issue to retouch photos with high-level tools. Instead of giving the optimal synthetic photo to the user, they retarget the differences between the reconstructed original photo and the synthesized photo to the original photo, using either optical flow or masking techniques. Unlike [32], we also apply a retargeting technique in our approach to preserve the topology and the fine details of the original 3D model. Moreover, we work on 3D meshes instead of coarse 3D voxels grids.

3. Learning 3D shapes with DiscoNet

3.1. Preliminaries and assumptions

[17, 46] introduce the 3D autoencoder that serves as our ground architecture. The encoder f can be based on PointNet [39] or FoldingNet [46], to make it invariant to the ordering of the input point cloud. The decoder g is based on AtlasNet [17], using a sphere as the parameterization of the surface to generate. g is a function of the latent vector given by f concatenated with a 3D coordinate on the unit sphere, with output in $\mathbb{R}^3$. It can alternatively be seen as a function which maps a latent vector to a function from $\mathbb{S}^2$ to $\mathbb{R}^3$. The output mesh is generated by sampling the sphere, and computing its corresponding transformation through the decoder. The topology of the sphere can finally be carried onto the output point cloud. This means that all synthesized shapes will be homeomorphic to a sphere. As we will see in subsection 4.3, the output shape is only used as an intermediate to retarget a realistic morphing to the original shape. To compute this morphing, it is useful to work on a spherical topology, on which we can always find 3D correspondences.

Let S be a sampling of the unit sphere. To learn a single 3D autoencoder, [17, 46] minimize the following Chamfer loss, invariant to the ordering of the output point cloud:

$$d_{\mathrm{CH}}(x, g \circ f(x)) = \sum_{p \in x} \min_{q \in g(f(x),S)} \|p-q\|_2^2 + \sum_{q \in g(f(x),S)} \min_{p \in x} \|p-q\|_2^2 \ , \quad (1)$$

where $g(f(x), S) = \{g(f(x), s) \mid s \in S\}$ is the reconstruction of the input point cloud x by the autoencoder $g \circ f$.

We assume that our dataset contains samples from a disconnected manifold of plausible shapes, which can be partitioned into (at least) k separated components. Our goal is to learn these k components by extending the 3D autoencoder described above, since such an autoencoder cannot learn a disconnected manifold without also generating implausible models. Indeed, the image by a (continuous) decoder of the latent space (which is connected) must be connected. We could think that such an autoencoder would yet learn to compact the inevitable implausible part of its latent space. However, Theorem 1 proves that this is not possible, as any interpolating path between two separated components intersects in the latent space a ball of implausible models, which cannot be reduced by changing the decoder's weights.

Theorem 1. *Let g_θ be a decoder, and $(\mathcal{M}_1, \mathcal{M}_2)$ a partition of the subset of plausible models of the input space, such that $d(\mathcal{M}_1, \mathcal{M}_2) > 0$. For any $r > 0$, there exists $\mathrm{C}_r > 0$, such that whatever the weights θ with $\|\theta\|_\infty \leq r$, on any continuous path between $g_\theta^{-1}(\mathcal{M}_1)$ and $g_\theta^{-1}(\mathcal{M}_2)$ there exists a latent vector h, such that any model in $g_\theta(B(h, \mathrm{C}_r))$ is implausible,* i.e. *$g_\theta(B(h, \mathrm{C}_r)) \subset (\mathcal{M}_1 \cup \mathcal{M}_2)^c$.*

Proof. See the supplementary material. ■

3.2. Disconnected manifold learning

To tackle the learning of k components, the Theorem 1 encourages to introduce k 3D autoencoders that we denote $\{g_i \circ f_i \mid i \in [\![1, k]\!]\}$. To jointly learn our k autoencoders, we minimize the following loss for a given input x:

$$\mathcal{L}(\theta_1, \ldots, \theta_k) = \min_{i \in [\![1,k]\!]} d_{\mathrm{CH}}(x, g_i \circ f_i(x)) \,, \quad (2)$$

where θ_i denotes the weights of the i-th autoencoder.

Thus, for each input, the gradient of the loss is back-propagated only through the autoencoder which achieves the minimum of the Chamfer losses among all the autoencoders, namely only the best autoencoder is optimized for each input. If one of the autoencoders becomes slightly better than the others for a specific component of the input manifold, *i.e.* a specific kind of shapes, it achieves the lowest Chamfer loss on these inputs. Hence, it is updated to become better, and therefore achieves an even lower loss over the next epoch. This mechanism engages a virtuous circle, which drives each autoencoder to focus on a different subset of low variance. Moreover, as only one autoencoder is updated for each input, the more one autoencoder specializes itself on a specific component, the more the others are hindered to learn this component, and therefore are also forced to learn another component of the input manifold, reinforcing even more this virtuous circle. During the inference, we naturally choose for each input the autoencoder achieving the lowest loss $\mathcal{L}$.

To help each autoencoder to specialize, we also use a reassignment algorithm during the mini-batch training. Indeed, due to the random initialization of the k autoencoders, it could happen that at the beginning of the training all shapes reach their lowest Chamfer loss with the same autoencoder, impeding the other autoencoders from being optimized. In that case, one of the autoencoders would learn the whole

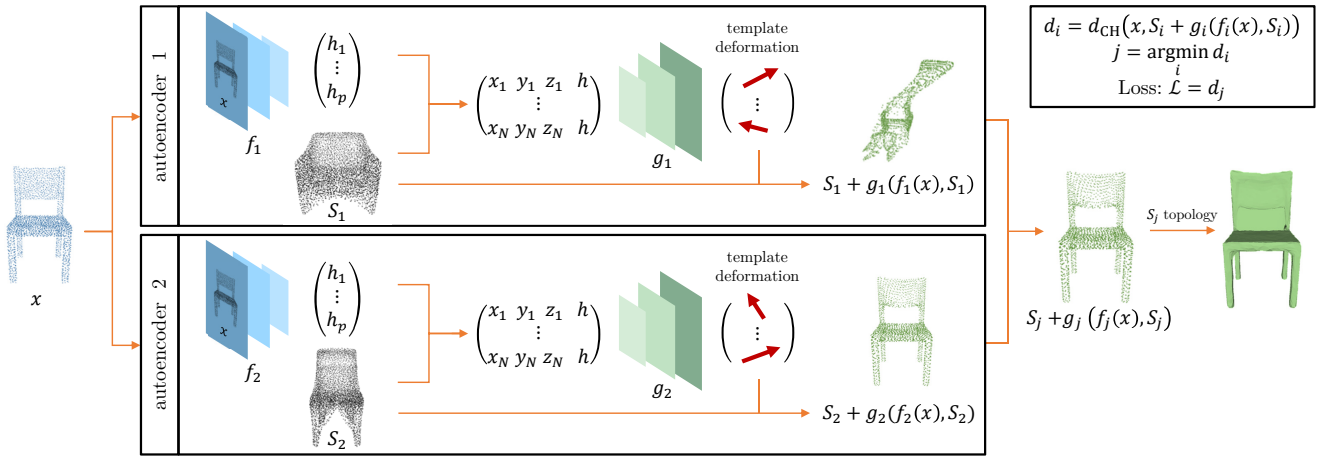


Figure 2: Illustration of the DiscoNet pipeline with $k = 2$ autoencoders. The input x is reconstructed through both autoencoders, the best reconstruction being used as the output. Each decoder synthesizes the shape by warping its own pre-learned 3D template S_i.

dataset, while the others would remain at their initial random state. To preclude that effect, our reassignment algorithm aims at dynamically changing the $\arg\min$ in Equation (2), such that each autoencoder is assigned to at least n shapes for each mini-batch, where n is a small ratio η of the mini-batch size. It proceeds by reassigning to each autoencoder the n shapes whose Chamfer loss d_{CH} is the closest to the minimal loss $\mathcal{L}$, in order to force the backpropagation for those shapes to go through this autoencoder. Hence, each autoencoder is guaranteed to be optimized at each mini-batch. The expected number of shapes per component is not needed, as long as the ratio η is chosen to be lower than the proportion of shapes in the smallest component, since η is only a lower-bound preventing any autoencoder from being left aside. A pseudo code is given in the supplementary material.

Contrary to [22], our reassignment algorithm prevents the collapse of the autoencoders without using any weighting inference scheme. Moreover, we guarantee during the learning to use the best autoencoder for each input (except for the few reassigned inputs). The backpropagation is also faster as it is only done through one autoencoder for each input, unlike [22]. Finally, we do not need to learn any prior on the decoders. If k is lower than the "true" number of components, each decoder will simply model the components closest to its cluster. That is why $k = 2$ works well in practice.

3.3. Template-based autoencoders

The spherical parameterization of each decoder is only used for its topology, but it does not give any hint on the geometry to generate. Our second contribution to learn a disconnected manifold of 3D shapes consists in replacing the sphere in each decoder with a dedicated 3D template, closer to the expected shape. The templates are learned before their corresponding autoencoders, in such a way that each template becomes representative of the mean geometry of the component that is going to be learned by its autoencoder.

This template is used as a base shape that the decoder will locally deform towards the expected shape. More formally, let S_i be the pre-learned template of the i-th autoencoder. The reconstruction of x by $g_i \circ f_i$ is now given by

$$S_i + g_i(f_i(x), S_i)\,, \tag{3}$$

like a ResNet [20] architecture, contrary to the AtlasNet architecture. The decoder g_i only predicts a small transformation of each vertex of the template S_i. Thus, it becomes easier to learn such a decoder, as we start from a template which is already close to the kind of shapes that this autoencoder is going to learn. It also helps each decoder keep a coherent topology, and a consistent correspondence between the vertices of the generated shapes. Lastly, it leads to an improved reconstruction, as each autoencoder only needs to slightly deform its template to match an input. Of course, once the templates are learned, it is necessary to initialize the decoders g_i so that they predict very small deformations for any input in order to leverage their templates. If a decoder starts with too large deformations, it distorts its template into an implausible shape, making the template useless. The whole DiscoNet inference pipeline is illustrated in Figure 2. Notice that we can also pre-learn a template with a single autoencoder, as it also improves its accuracy.

We want the templates to represent the "centroids" of each component of the input manifold. However, they cannot be simply learned using k-means or mean-shift algorithms [3, 12] since the training shapes have no correspondences to match their vertices. We could embed the training shapes in a 3D voxels grid to compute a k-means clustering, but

the Euclidean distance is known not to be the most relevant metric regarding voxelized 3D models as explained in [42]. To learn these k templates, we start with k discretized unit spheres $S_1, \dots, S_k$, and treat their vertices as optimization variables. Similarly to subsection 3.2, we jointly learn the templates by minimizing a 3D distance between each template S_i and its closest training shapes. We do not use the Chamfer distance, as it is also known to lead to poor mean shapes as illustrated in [1]. We use the Earth Mover distance instead, which is an optimal transport distance derived from the Wasserstein distance in the discrete setting:

$$d_{\mathrm{EM}}(x, S) = \min_{\substack{\varphi : S \to x \\ \varphi \text{ bijective}}} \sum_{p \in S} \|p - \varphi(p)\|_2^2 \,. \tag{4}$$

Note that the Earth Mover distance requires the templates to have the same number of points than the training shapes, but this is a mild constraint as the training dataset is resampled anyway as a pre-processing step. To compute this distance, we use the relaxation algorithm described in [4]. To jointly learn the templates, we minimize by mini-batch gradient descent the following loss over all training shapes x:

$$\mathcal{L}_t(S_1, \dots, S_k) = \min_{i \in [\![1,k]\!]} d_{\mathrm{EM}}(x, S_i) \,. \tag{5}$$

The loss $\mathcal{L}_t$ acts similarly to the loss $\mathcal{L}$. Therefore, each template is going to converge towards a specific typology of shapes. For identical reasons as those explained in subsection 3.2, we also use the same reassignment algorithm to help the templates approximate different kinds of shapes. Once the templates are learned, they remain fixed and each autoencoder is associated to a different template.

During the optimization of the templates, nothing prevents their vertices from mixing up, breaking the initial spherical topology as illustrated in Figure 3. To get a consistent spherical topology of the pre-learned templates, we run a Poisson surface reconstruction [24] on their point cloud, which requires accurate oriented normals. We extract these normals from a coarse mesh computed from the template point cloud with the marching cubes algorithm [33].

3.4. Experiments

	chairs	*cars*	*planes*	*all*
AtlasNet [17]	2.31	0.91	0.94	1.78
FN/AN [46]+[17]	2.22	0.87	0.89	1.46
DiscoNet	**1.87**	**0.84**	**0.81**	**1.25**

Table 1: Mean distance error over the test set ($\times 100$), with $k = 2$ for chairs, cars, and planes, and $k = 3$ for all three datasets together.

We evaluate our model on the ShapeNetCore v2 dataset [10], on three categories suited to 3D editing: chairs (6778 meshes), cars (3533), and planes (4045). We split each

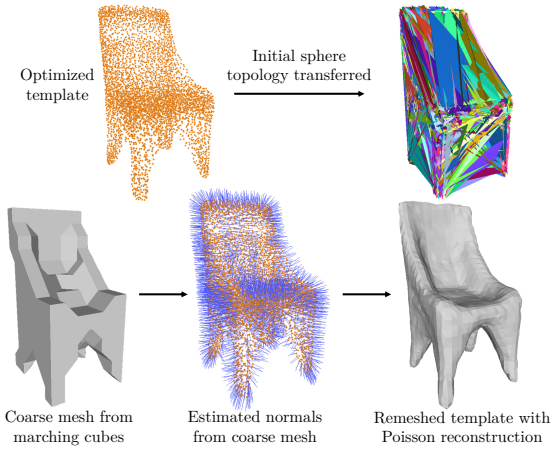


Figure 3: The first row illustrates that the template optimization breaks the initial topology. The second row shows our pipeline to remesh the template point cloud.

dataset into a training set (80% of the models), a validation set (10 %) and a test set (10 %). We implement two baselines: the AtlasNet [17] baseline, consisting in a PointNet-like encoder, and a decoder parameterized with a sphere; and the "FN/AN" baseline, using the same sphere-based AtlasNet decoder, but a FoldingNet encoder [46], which takes as input the local covariances in addition to the point cloud coordinates, and also adds graph pooling layers which are local max pooling over the neighborhood of each point. To run fair comparisons between these two baselines and DiscoNet, we base all models on the same deep architecture, except covariances and graph pooling absent in AtlasNet. Implementation details are given in the supplementary material.

We claimed that our learning model is able to represent more accurately the shapes' topology. As pointed out in [38], AtlasNet only uses the vertices of the output surface to compute its evaluation metric, and therefore does not penalize the unrealistic triangles that cover empty spaces in the ground-truth shape. That is why, similarly to [38], we use the mean distance from a uniform sampling over all the reconstructed surface y to the ground-truth mesh x to evaluate our model against the baselines. This evaluation distance purposefully penalizes the triangles in y that should not exist (*i.e.* whose samples are far from the input x), and thus is representative of the quality of the reconstructed shape. As shown by the Table 1, DiscoNet greatly outperforms the state-of-the-art, as it achieves the lowest error on all the datasets.

Qualitatively, DiscoNet also produces more accurate geometry and topology, as illustrated in Figure 4. In particular, we observe that FN/AN (the best baseline) generates a lot of self-intersections contrary to our model. Additional qualitative results are provided in the supplementary material. Figure 10 in subsection 4.4 will also demonstrate that this improved quality over the state-of-the-art strongly enhances

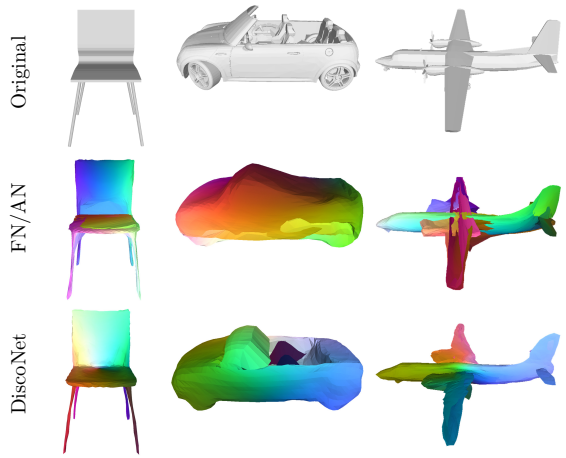


Figure 4: Reconstruction results, with colors transferred from the unit sphere for FN/AN and from our pre-learned templates for DiscoNet to highlight the topology. $k = 2$ for DiscoNet.

the results of our 3D editing system.

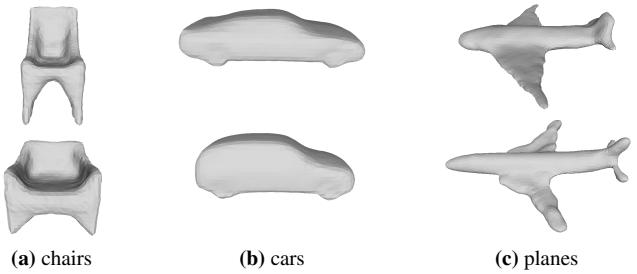
Besides, we experiment the specific ability of DiscoNet to learn a disconnected manifold. Figure 5 shows that the pre-learned templates automatically specialize on two different kinds of shapes, as expected. Furthermore, Figure 6 illustrates some reconstruction results obtained by each autoencoder on the chairs dataset. We see that DiscoNet naturally clusters the dataset such that each autoencoder is dedicated to a different cluster, one with lounge chairs and office chairs, and one with standard chairs.

Finally, the ablation study provided in the Table 2 demonstrates the complementarity of all our contributions. In this table, "multiAE" is the DiscoNet model using a sphere parameterization of the decoder instead of any pre-learned template, and "template" is the DiscoNet model with $k = 1$, *i.e.* using a single autoencoder with a pre-learned template instead of a sphere parameterization of the decoder. The Table 2 shows that both learning several autoencoders (subsection 3.2) and pre-learning 3D templates (subsection 3.3) are essential to the performance of DiscoNet. Notice that the poor performance of the "template" model on the mixed dataset of all three categories together is completely expected, as a single template cannot well represent three such different classes, while "multiAE" performs well on this dataset as it can naturally be clustered into three classes.

4. Learning-guided 3D editing

4.1. Editing interface

We are now able to leverage our learned manifold of 3D shapes to create a realistic 3D editing system, summarized in Figure 7. The user selects a 3D mesh x belonging to a component of shapes that an autoencoder $g \circ f$ has learned using DiscoNet. We propose an innovative interface to allow



(a) chairs **(b)** cars **(c)** planes

Figure 5: DiscoNet pre-learned templates with $k = 2$, on each of the three datasets: chairs, cars, and planes.

Figure 6: Illustration of the clustering ability of DiscoNet to specialize on 2 different kinds of chairs using $k = 2$ autoencoders.

	chairs	*cars*	*planes*	*all*
multiAE	2.42	0.89	0.95	1.45
template	1.93	0.86	0.81	1.61
DiscoNet	**1.87**	**0.84**	**0.81**	**1.25**

Table 2: Mean distance error over the test set ($\times 100$), with $k = 2$ for chairs, cars, and planes, and $k = 3$ for all three datasets together.

the user to easily edit this shape by locally retouching the 3D mesh without any constraint. Our system continuously updates the edited model to make it plausible, while complying with the user's editing. Our interface offers two different user-friendly editing tools.

The user can attach 3D control points $(a_1, \ldots, a_m)$ to the mesh, which act as handles to deform it. The user drags some of these handles, for example $(a_1, \ldots, a_p)$, the input mesh being locally deformed to match the new positions of these control points. Let $(a'_1, \ldots, a'_m)$ be the control points after the user has pulled some of them. We compute a 3D deformation field δ with interpolates the 3D translations $a'_i - a_i$ using RBF interpolation [7]. This deformation field applied to the reconstructed original model defines our optimization target z: $z = g(f(x)) + \delta(g(f(x))$.

Inspired by [28], the user can also resketch directly over any 2D view of the 3D model. In this case, the optimization target is directly the pixels coordinates of the 2D sketch S.

4.2. Optimization

In order to find a plausible shape which fulfills the user's constraints, we define an optimization energy over the latent vector h to match the optimization target defined previously.

With the 3D control points interface, we minimize

$$E_c^1(h) = \frac{1}{|J|} \sum_{j \in J} \|g(h)_j - z_j\|_2^2 \ , \tag{6}$$

where J is the set of vertices of z close to the deformed handles: $J = \left\{ j \mid \exists i \in [\![1, p]\!] , \|z_j - a'_i\|_2 \leq \rho \right\}$, with ρ a positive threshold. E_c^1 compels the synthesized mesh $g(h)$ to match the target z around the handles pulled by the user. Indeed, we do not aim at matching the whole shape z, which is implausible, since the user's deformation is only local.

With the sketching interface, we minimize

$$E_c^2(h) = \frac{1}{|S|} \sum_{s \in S} \min_{j \in V} \|P[g(h)_j] - s\|_2^2 \tag{7}$$

where P is the projection associated to the view chosen by the user to draw his sketch S, and V is the subset of vertices of $g(f(x))$ which are on the contour of $P[g(f(x))]$. E_c^2 enforces the contour of the synthesized mesh $g(h)$ in the view defined by P to match the 2D sketch S.

We point out that Equations (6) and (7) defining E_c^1 and E_c^2 implicitly assume a persistent correspondence between the vertices of the shapes generated by the decoder g, since the sets J and V remain fixed during the optimization.

To constrain the synthetic shape $g(h)$ to lie close to the original shape, we also add a similarity term which penalizes in the latent space the distance to the original shape:

$$E_s(h) = \lambda_s \|h - f(x)\|_2^2 \ . \tag{8}$$

Finally, to reinforce the plausibility of the synthetic shape $g(h)$, we fit a Gaussian mixture model on the latent space, and minimize the negative log-likelihood

$$E_p(h) = -\lambda_p \log(p(h)) \, , \tag{9}$$

where $p(h)$ is the density of the Gaussian mixture.

We minimize the final energy $E_c(h) + E_s(h) + E_p(h)$ using the BFGS method [37], starting from $h = f(x)$.

4.3. Retargeting

The outcome $\widehat{h}$ of the previous optimization allows us to synthesize a plausible mesh $g(\widehat{h})$, close to the edited model. Unfortunately, this mesh does not necessarily respect the topology and the details of the original model x. To tackle this issue, we retarget the deformations between $g(f(x))$ and $g(\widehat{h})$ to x, in order to get a realistic morphing of x, keeping all its details, while matching the user's editing.

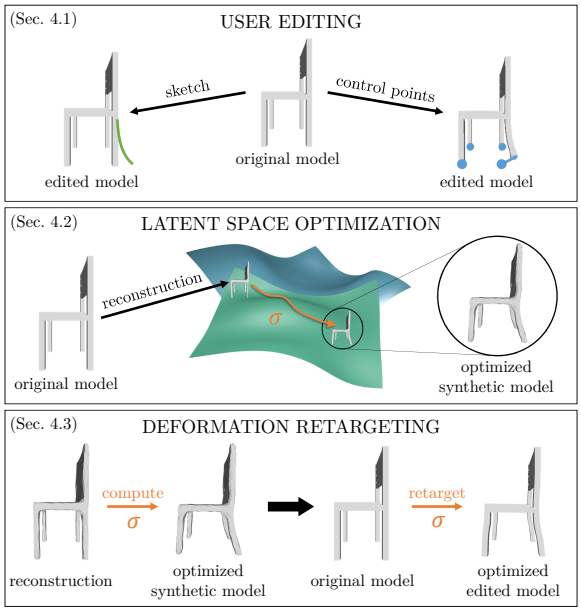


Figure 7: Illustration of our 3D editing pipeline: optimization on the learned shapes manifold to locally fit a user edit, and retargeting of the induced morphing onto the original model.

First we subsample $g(f(x))$, starting from a random vertex, and iteratively adding the furthest one to the current subsampled set, to get a uniform covering of keypoints C. Then, we compute a 3D deformation field σ between $g(f(x))$ and $g(\widehat{h})$ using kernel regression [36] over the previously subsampled set of keypoints C. The final realistic edited model provided to the user is given by the retargeting $x + \sigma(x)$. Other approaches could also be used to compute such a morphing σ, for example with non rigid registration [29, 47]. However, as DiscoNet preserves the correspondences in the decoder's outputs thanks to the pre-learned templates, we can adopt the simpler kernel regression, which also smooths the deformation field and produces better results.

In order to get an accurate retargeting, the deformation σ should predict a realistic morphing for any point of x. This implies that σ should be computed from a set of keypoints that fully covers the space spanned by x, whatever its topology. This is finally the reason why we restrain the templates to the spherical topology in DiscoNet, in order to ensure that the set of keypoints C does not miss any part.

4.4. Results

We now demonstrate the capabilities of our 3D editing pipeline. To remove any bias, the experiments are made with shapes coming from the test set. Our experiments are based on DiscoNet using $k = 2$, *i.e.* the manifold is learned with 2 autoencoders, each one based on its own pre-learned 3D template. The hyperparameters of the optimization and retargeting are specified in the supplementary material.

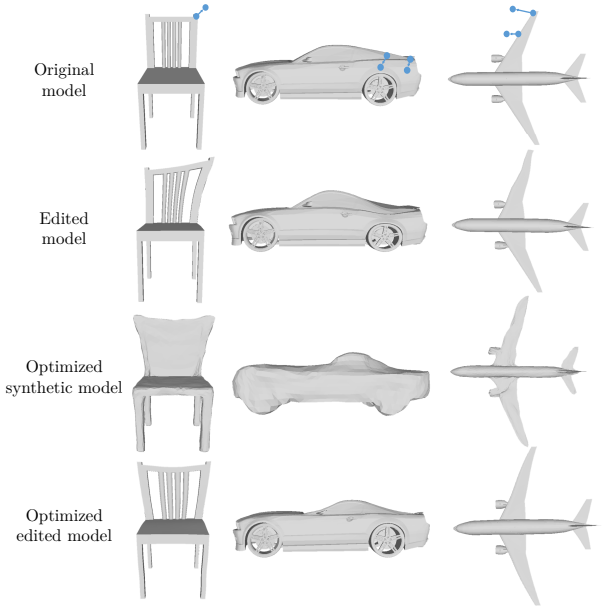


Figure 8: Results of handles-based editing on three examples.

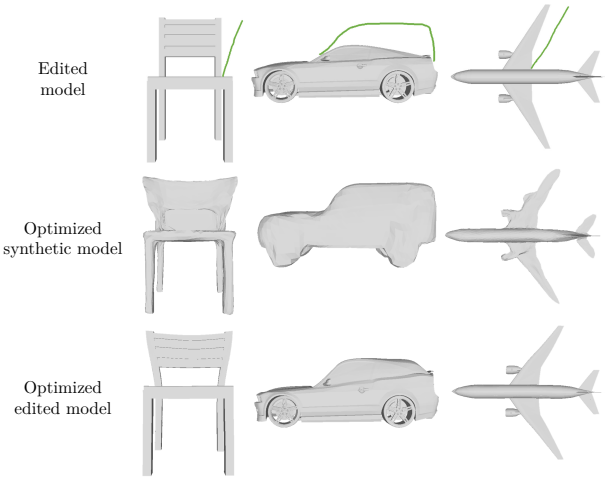


Figure 9: Results of sketch-based editing on three examples.

Figures 8 and 9 show the performance of our handles-based and sketch-based editing tool. Both interfaces produce realistic shapes, compliant with the user's editing. More editing examples can be found in the supplementary material.

Importantly, Figure 10 compares two editing results, one using the FN/AN learning baseline, the other based on DiscoNet. It justifies the benefit of DiscoNet over the 3D learning state-of-the-art, since it leads to a superior edited model, due to a better reconstruction, but also a more consistent topology, needed for a meaningful optimization as evoked in subsection 4.2.

Finally, we also compare the impact of our plausibility and similarity terms on the optimization result. Figure 11

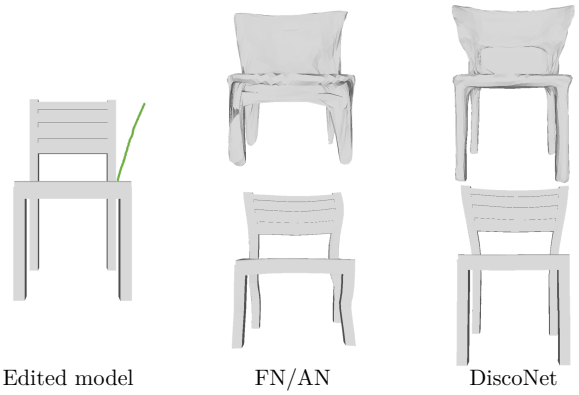


Figure 10: Highlighting of the benefit of DiscoNet versus the state-of-the-art learning model for 3D editing purpose.

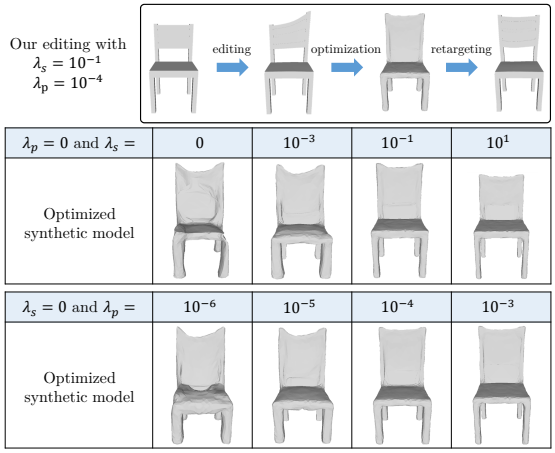


Figure 11: Illustration of the relevance of the similarity (first row) and plausibility (second row) energies for the optimization.

shows that both terms are useful to enforce a plausible shape, close to the initial edited model.

5. Conclusion

In this paper we introduce DiscoNet, a new unsupervised shapes learning model able to learn a disconnected manifold. DiscoNet jointly learns several autoencoders, each one being based on its own pre-learned 3D template. We show that DiscoNet outperforms the state-of-the-art, as it generates finer shapes with better topology. We finally design a novel 3D editing pipeline, which allows an inexperienced user to easily edit a 3D shape, either by deforming 3D handles or by drawing over the model, while preserving the realism effortlessly. At the core of our 3D editing system, DiscoNet is leveraged to turn an edited shape into its realistic counterpart. We also demonstrate that DiscoNet is essential to the performance of our 3D editing tool.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *International Conference on Machine Learning (ICML)*, 2018.

[2] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. Scape: Shape completion and animation of people. In *SIGGRAPH*, 2005.

[3] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *Symposium on Discrete Algorithms (SODA)*, volume 18 of *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035, 2007.

[4] Dimitri P. Bertsekas. The auction algorithm: A distributed relaxation method for the assignment problem. *Annals of Operations Research*, 14(1-4):105–123, 1988.

[5] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *SIGGRAPH*, 1999.

[6] Andrew Brock, Theodore Lim, James Millar Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[7] Martin D. Buhmann. *Radial Basis Functions*. Cambridge University Press, 2003.

[8] Christopher P. Burgess, Irina Higgins, Arka Pal, Loïc Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in β-vae. In *NeurIPS Workshop on Learning Disentangled Features*, 2017.

[9] Thomas J. Cashman and Andrew W. Fitzgibbon. What shape are dolphins? Building 3D morphable models from 2D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):232–244, 2013.

[10] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing 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. *arXiv preprint arXiv:1512.03012*, 2015.

[11] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2016.

[12] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, 1995.

[13] Haoqiang Fan, Hao Su, and Leonidas Guibas. A point set generation network for 3d object reconstruction from a single image. In *Conference on Computer Vision and Pattern Regognition (CVPR)*, 2017.

[14] Ran Gal, Olga Sorkine, Niloy J. Mitra, and Daniel Cohen-Or. iwires: An analyze-and-edit approach to shape manipulation. In *SIGGRAPH*, 2009.

[15] Rohit Girdhar, David F. Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *European Conference on Computer Vision (ECCV)*, 2016.

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2014.

[17] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *Conference on Computer Vision and Pattern Regognition (CVPR)*, 2018.

[18] Emilie Guy, Jean-Marc Thiery, and Tamy Boubekeur. Click&draw selection. In *SIGGRAPH*, 2013.

[19] Christian Häne, Shubham Tulsiani, and Jitendra Malik. Hierarchical surface prediction for 3d object reconstruction. In *International Conference on 3D Vision (3DV)*, 2017.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[21] Geoffrey Hinton and Ruslan Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504 – 507, 2006.

[22] Dominic Jack, Jhony K. Pontes, Sridha Sridharan, Clinton Fookes, Sareh Shirazi, Frédéric Maire, and Anders Eriksson. Learning free-form deformations for 3d object reconstruction. In *Asian Conference on Computer Vision (ACCV)*, 2018.

[23] Arjun Jain, Thorsten Thormählen, Tobias Ritschel, and Hans-Peter Seidel. Exploring shape variations by 3d-model decomposition and part-based recombination. In *Eurographics*, 2012.

[24] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Symposium on Geometry Processing (SGP)*, 2006.

[25] Mahyar Khayatkhoei, Maneesh K. Singh, and Ahmed Elgammal. Disconnected manifold learning for generative adversarial networks. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

[26] Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

[27] Ilya Kostrikov, Zhongshi Jiang, Daniele Panozzo, Denis Zorin, and Joan Bruna. Surface networks. In *Conference on Computer Vision and Pattern Regognition (CVPR)*, 2018.

[28] Vladislav Kraevoy, Alla Sheffer, and Michiel van de Panne. Modeling from contour drawings. In *Symposium on Sketch-Based Interfaces and Modeling (SBIM)*, 2009.

[29] Hao Li, Robert W. Sumner, and Mark Pauly. Global correspondence optimization for non-rigid registration of depth scans. In *Symposium on Geometry Processing (SGP)*, 2008.

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

[31] Or Litany, Alex Bronstein, Michael Bronstein, and Ameesh Makadia. Deformable shape completion with graph convolutional autoencoders. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[32] Jerry Liu, Fisher Yu, and Thomas Funkhouser. Interactive 3d modeling with a generative adversarial network. In *International Conference on 3D Vision (3DV)*, 2017.

[33] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH*, 1987.

[34] Jonathan Masci, Davide Boscaini, Michael M. Bronstein, and Pierre Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *ICCV Workshop on 3D Representation and Recognition*, 2015.

[35] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[36] Èlizbar A. Nadaraya. On estimating regression. *Theory of Probability and its Applications*, 9:141–142, 1964.

[37] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 2006.

[38] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Conference on Computer Vision and Pattern Regognition (CVPR)*, 2019.

[39] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Conference on Computer Vision and Pattern Regognition (CVPR)*, 2017.

[40] Jess R. Nieto and Toni Susin. Cage based deformations: A survey. *Lecture Notes in Computational Vision and Biomechanics*, 7:75–99, 2013.

[41] Danilo J. Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning (ICML)*, 2014.

[42] Justin Solomon, Fernando de Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. In *SIGGRAPH*, 2015.

[43] Robert W. Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. In *SIGGRAPH*, 2007.

[44] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *International Conference on Computer Vision (ICCV)*, 2017.

[45] Kai Xu, Vladimir G. Kim, Qixing Huang, Niloy Mitra, and Evangelos Kalogerakis. Data-driven shape analysis and processing. In *SIGGRAPH Asia Courses*, 2016.

[46] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Conference on Computer Vision and Pattern Regognition (CVPR)*, 2018.

[47] I-Cheng Yeh, Chao-Hung Lin, Olga Sorkine, and Tong-Yee Lee. Template-based 3d model fitting using dual-domain relaxation. *IEEE Transactions on Visualization and Computer Graphics*, 99, 2010.

[48] Ersin Yumer, Siddhartha Chaudhuri, Jessica K. Hodgins, and Levent Burak Kara. Semantic shape editing using deformation handles. In *SIGGRAPH*, 2015.

[49] Ersin Yumer and Niloy J. Mitra. Learning semantic deformation flows with 3d convolutional networks. In *European Conference on Computer Vision (ECCV)*, 2016.

[50] Wei Zhang, Suyog Gupta, Xiangru Lian, and Ji Liu. Staleness-aware async-sgd for distributed deep learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2016.

[51] Kun Zhou, Jin Huang, John Snyder, Xinguo Liu, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Large mesh deformation using the volumetric graph laplacian. In *SIGGRAPH*, 2005.

[52] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision (ECCV)*, 2016.