3DN: 3D Deformation Network

Weiyue Wang[1] Duygu Ceylan[2] Radomir Mech[2] Ulrich Neumann[1]

[1]University of Southern California
Los Angeles, California
{weiyuewa,uneumann}@usc.edu

[2]Adobe
San Jose, California
{ceylan,rmech}@adobe.com

Abstract

Applications in virtual and augmented reality create a demand for rapid creation and easy access to large sets of 3D models. An effective way to address this demand is to edit or deform existing 3D models based on a reference, e.g., a 2D image which is very easy to acquire. Given such a source 3D model and a target which can be a 2D image, 3D model, or a point cloud acquired as a depth scan, we introduce 3DN*, an end-to-end network that deforms the source model to resemble the target. Our method infers per-vertex offset displacements while keeping the mesh connectivity of the source model fixed. We present a training strategy which uses a novel differentiable operation,* mesh sampling operator*, to generalize our method across source and target models with varying mesh densities.* Mesh sampling operator *can be seamlessly integrated into the network to handle meshes with different topologies. Qualitative and quantitative results show that our method generates higher quality results compared to the state-of-the art learning-based methods for 3D shape generation.*

1. Introduction

Applications in virtual and augmented reality and robotics require rapid creation and access to a large number of 3D models. Even with the increasing availability of large 3D model databases [1], the size and growth of such databases pale when compared to the vast size of 2D image databases. As a result, the idea of editing or deforming existing 3D models based on a reference image or another source of input such as an RGBD scan is pursued by the research community.

Traditional approaches for editing 3D models to match a reference target rely on optimization-based pipelines which either require user interaction [32] or rely on the existence of a database of segmented 3D model components [9]. The development of 3D deep learning methods [17, 2, 31, 28, 10] inspire more efficient alternative ways to handle 3D data. In fact, a multitude of approaches have been presented over the past few years for 3D shape generation using deep learning. Many of these, however, utilize voxel [33, 5, 37, 29, 24, 30, 34, 27] or point based representations [3] since the representation of meshes and mesh connectivity in a neural network is still an open problem. The few recent methods which do use mesh representations make assumptions about fixed topology [7, 25] which limits the flexibility of their approach.

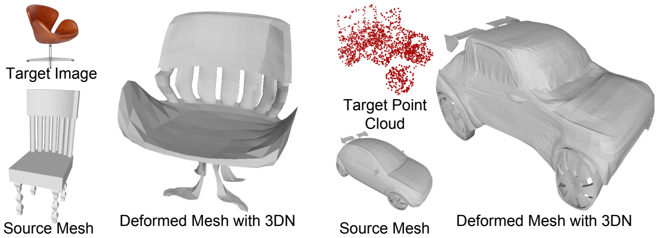


Figure 1: 3DN deforms a given a source mesh to a new mesh based on a reference target. The target can be a 2D image or a 3D point cloud.

This paper describes *3DN*, a 3D deformation network that deforms a source 3D mesh based on a target 2D image, 3D mesh, or a 3D point cloud (e.g., acquired with a depth sensor). Unlike previous work which assume a fixed topology mesh for all examples, we utilize the mesh structure of the source model. This means we can use any existing high-quality mesh model to generate new models. Specifically, given any source mesh and a target, our network estimates vertex displacement vectors (3D offsets) to deform the source model while maintaining its mesh connectivity. In addition, the global geometric constraints exhibited by many man-made objects are explicitly preserved during deformation to enhance the plausibility of the output model.

Our network first extracts global features from both the source and target inputs. These are input to an *offset decoder* to estimate per-vertex offsets. Since acquiring ground truth correspondences between the source and target is very challenging, we use unsupervised loss functions (e.g., Chamfer and Earth Mover's distances) to compute the similarity of the deformed source model and the target. A difficulty in measuring similarity between meshes is the varying mesh densities across different models. Imagine a planar surface represented by just 4 vertices and 2 triangles as opposed to a dense set of planar triangles. Even though these meshes represent the same shape, vertex-based similarity computation may yield large errors. To overcome this problem, we adopt a point cloud intermediate representation. Specifically, we sample a set of points on both the deformed source mesh and the target model and measure the loss between the resulting point sets. This measure introduces a differentiable mesh sampling operator which propagates features, e.g., offsets, from vertices to points in a differentiable manner.

We evaluate our approach for various targets including 3D shape datasets as well as real images and partial points scans. Qualitative and quantitative comparisons demonstrate that our network learns to perform higher quality mesh deformation compared to previous learning based methods. We also show several applications, such as shape interpolation. In conclusion, our contributions are as follows:

- We propose an end-to-end network to predict 3D deformation. By keeping the mesh topology of the source fixed and preserving properties such as symmetries, we are able to generate plausible deformed meshes.
- We propose a differentiable mesh sampling operator in order to make our network architecture resilient to varying mesh densities in the source and target models.

2. Related Work

2.1. 3D Mesh Deformation

3D mesh editing and deformation has received a lot of attention from the graphics community where a multitude of interactive editing systems based on preserving local Laplacian properties [20] or more global features [4] have been presented. With easy access to growing 2D image repositories and RGBD scans, editing approaches that utilize a reference target have been introduced. Given source and target pairs, such methods use interactive [32] or heavy processing pipelines [9] to establish correspondences to drive the deformation. The recent success of deep learning has inspired alternative methods for handling 3D data. Yumer and Mitra[36] propose a volumetric CNN that generates a deformation field based on a high level editing intent. This method relies on the existence of model editing results based on semantic controllers. Kurenkov et al. present DeformNet [14] which employs a free-form deformation (FFD) module as a differentiable layer in their network. This network, however, outputs a set of points rather than a deformed mesh.Furthermore, the deformation space lacks smoothness and points move randomly. Groueix et al. [6] present an approach to compute correspondences across deformable models such as humans. However, they use an intermediate common template representation which is hard to acquire for man-made objects. Pontes et al. [16] and Jack et al. [11] introduce methods to learn FFD. Yang et al. propose Foldingnet [35] which deforms a 2D grid into a 3D point cloud while preserving locality information. Compared to these existing methods, our approach is able to generate higher quality deformed meshes by handling source meshes with different topology and preserving details in the original mesh.

2.2. Single View 3D Reconstruction

Our work is also related to single-view 3D reconstruction methods which have received a lot of attention from the deep learning community recently. These approaches have used various 3D representations including voxels [33, 2, 5, 37, 29, 24, 30, 34], point clouds [3], octrees [23, 8, 26], and primitives [38, 15]. Sun et al. [21] present a dataset for 3D modeling from single-images. However, pose ambiguity and artifacts widely occur in this dataset. More recently, Sinha et al. [19] propose a method to generate the surface of an object using a representation based on geometry images. In a similar approach, Groueix et al. [7] present a method to generate surfaces of 3D shapes using a set of parametric surface elements. The more recent method of Hiroharo et al. [13] and Kanazawa et al. [12] also uses differentiable renderer and per-vertex displacements as a deformation method to generate meshes from image sets. Wang et al. [25] introduce a graph-based network to reconstruct 3D manifold shapes from input images. These recent methods, however, are limited to generating manifolds and require 3D output to be topology invariant for all examples.

3. Method

Given a source 3D mesh and a target model (represented as a 2D image or a 3D model), our goal is to deform the source mesh such that it resembles the target model as close as possible. Our deformation model keeps the triangle topology of the source mesh fixed and only updates the vertex positions. We introduce an end-to-end *3D deformation network (3DN)* to predict such per-vertex displacements of the source mesh.

We represent the source mesh as $S = (V, E)$, where $V \in \mathbb{R}^{N_V \times 3}$ is the (x, y, z) positions of vertices and $E \in \mathbb{Z}^{N_E \times 3}$ is the set of triangles and encodes each triangle with the indices of vertices. N_V and N_E denote the number of vertices and triangles respectively. The target model T is either a $H \times W \times 3$ image or a 3D model. In case T is a 3D model, we represent it as a set of 3D points $T \in \mathbb{R}^{N_T \times 3}$, where N_T denotes the number of points in T.

As shown in Figure 2, 3DN takes S and T as input and outputs per-vertex displacements, i.e., offsets, $O \in \mathbb{R}^{N_V \times 3}$.

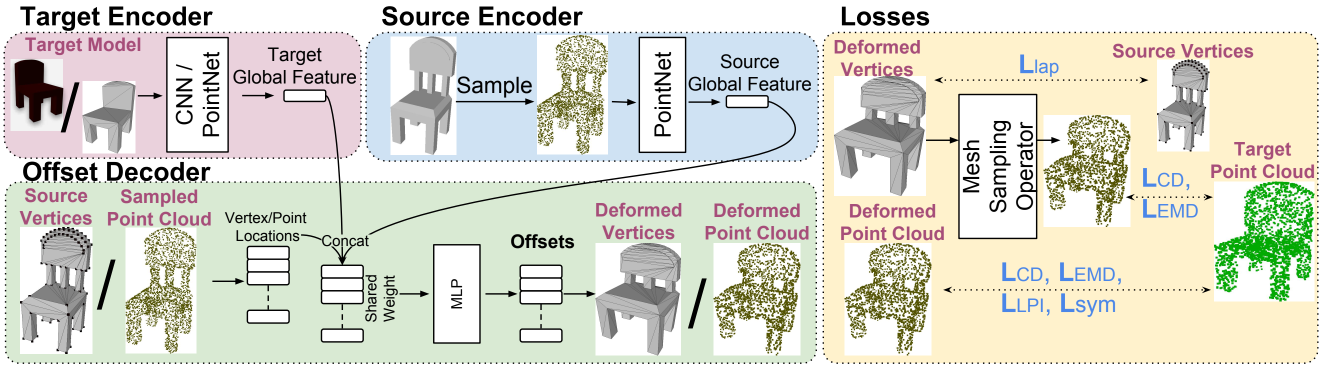


Figure 2: 3DN extracts global features from both the source and target. 'MLP' denotes the '1×1' conv as in PointNet [17]. These features are then input to an offset decoder which predicts per-vertex offsets to deform the source. We utilize loss functions to preserve geometric details in the source ($L_{Lap}, L_{LPI}, L_{Sym}$) and to ensure deformation output is similar to the target (L_{CD}, L_{EMD}).

The final deformed mesh is $S' = (V', E)$, where $V' = V + O$. Moreover, 3DN can be extended to produce per-point displacements when we replace the input source vertices with a sampled point cloud on the source. 3DN is composed of a target and a source encoder which extract global features from the source and target models respectively, and an offset decoder which utilizes such features to estimate the shape deformation. We next describe each of these components in detail.

3.1. Shape Deformation Network (3DN)

Source and Target Encoders. Given the source model S, we first uniformly sample a set of points on S and use the PointNet [17] architecture to encode S into a *source global feature vector*. Similar to the source encoder, the target encoder extracts a *target global feature vector* from the target model. In case the target model is a 2D image, we use VGG [18] to extract features. If the target is a 3D model, we sample points on T and use PointNet. We concatenate the source and target global feature vectors into a single *global shape feature vector* and feed into the offset decoder.

Offset Decoder. Given the global shape feature vector extracted by the source and target encoders, the offset decoder learns a function $F(\cdot)$ which predicts per-vertex displacements, for S. In other words, given a vertex $\mathbf{v} = (x_v, y_v, z_v)$ in S, the offset decoder predicts $F(\mathbf{v}) = \mathbf{o_v} = (x_{o_v}, y_{o_v}, z_{o_v})$ updating the deformed vertex in S' to be $\mathbf{v}' = \mathbf{v} + \mathbf{o_v}$.

Offset decoder is easily extended to perform point cloud deformations. When we replace the input vertex locations to point locations, say given a point $\mathbf{p} = (x_p, y_p, z_p)$ in the point cloud sampled form S, the offset decoder predicts a displacement $F(\mathbf{p}) = \mathbf{o_p}$, and similarly, the deformed point is $\mathbf{p}' = \mathbf{p} + \mathbf{o_p}$.

The offset decoder has an architecture similar to the PointNet segmentation network [17]. However, unlike the original PointNet architecture which concatenates the global shape feature vector with per-point features, we concatenate the original point positions to the global shape feature. We find this enables to better capture the vertex and point locations distribution in the source, and results in effective deformation results. We study the importance of this architecture in Section 4.3. Finally we note that, our network is flexible to handle source and target models with varying number of vertices.

3.2. Learning Shape Deformations

Given a deformed mesh S' produced by 3DN and the 3D mesh corresponding to the target model $T = (V_T, E_T)$, where $V_T \in \mathbb{R}^{N_{V_T} \times 3} (N_{V_T} \neq N_V)$ and $E_T \neq E$, the remaining task is to design a loss function that measures the similarity between S' and T. Since it is not trivial to establish ground truth correspondences between S' and T, our method instead utilizes the Chamfer and Earth Mover's losses introduced by Fan et al. [3]. In order to make these losses robust to different meshing densities across source and target models, we operate on set of points uniformly sampled on S' and T by introducing the *differentiable mesh sampling operator (DMSO)*. DMSO is seamlessly integrated in 3DN and bridges the gap between handling meshes and loss computation with point sets.

Differentiable Mesh Sampling Operator. As is illustrated in Figure 3, DMSO is used to sample a uniform set of points from a 3D mesh. Suppose a point $\mathbf{p}$ is sampled on the face $\mathbf{e} = (\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3})$ enclosed by the vertices $\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3}$. The position of $\mathbf{p}$ is then

$$\mathbf{p} = w_1\mathbf{v_1} + w_2\mathbf{v_2} + w_3\mathbf{v_3},$$

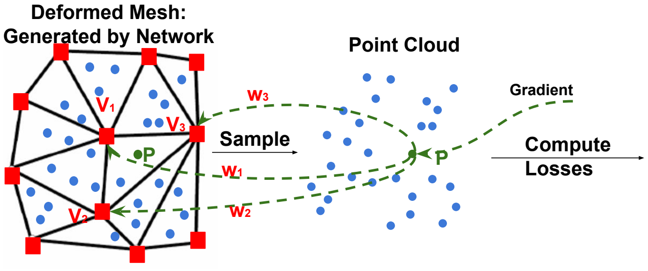


Figure 3: Differentiable mesh sampling operator (best viewed in color). Given a face $\mathbf{e} = (\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3})$, p is sampled on $\mathbf{e}$ in the network forward pass using barycentric coordinates w_1, w_2, w_3. Sampled points are used during loss computation. When performing back propagation, gradient of p is passed back to $(\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3})$ with the stored weights w_1, w_2, w_3. This process is differentiable.

where $w_1 + w_2 + w_3 = 1$ are the barycentric coordinates of $\mathbf{p}$. Given any typical feature for the original vertices, the per-vertex offsets in our case, $\mathbf{o_{v_1}}, \mathbf{o_{v_2}}, \mathbf{o_{v_3}}$, the offset of $\mathbf{p}$ is

$$\mathbf{o_p} = w_1\mathbf{o_{v_1}} + w_2\mathbf{o_{v_2}} + w_3\mathbf{o_{v_3}}.$$

To perform back-propogation, the gradient for each original per-vertex offsets $\mathbf{o_{v_i}}$ is calculated simply by $g_{\mathbf{o_{v_i}}} = w_i g_{\mathbf{o_{v_p}}}$, where g denotes the gradient.

We train 3DN using a combination of different losses as we discuss next.

Shape Loss. Given a target model, T, inspired by [3], we use Chamfer and Earth Mover's distances to measure the similarity between the deformed source and the target. Specifically, given the point cloud PC sampled on the deformed output and PC_T sampled on the target model, Chamfer loss is defined as

$$\begin{aligned} L_{\texttt{CD}}^{\texttt{Mesh}}(PC, PC_T) = &\sum_{p_1 \in PC} \min_{p_2 \in PC_T} \|p_1 - p_2\|_2^2 \\ &+ \sum_{p_2 \in PC_T} \min_{p_1 \in PC} \|p_1 - p_2\|_2^2, \end{aligned} \quad (1)$$

and Earth Mover's loss is defined as

$$L_{\texttt{EMD}}^{\texttt{Mesh}}(PC, PC_T) = \min_{\phi: PC \rightarrow PC_T} \sum_{p \in PC} \|p - \phi(p)\|_2, \quad (2)$$

where $\phi : PC \rightarrow PC_T$ is a bijection.

We compute these distances between point sets sampled both on the source (using the DMSO) and target models. Moreover, computing the above losses on point sets sampled on source and target models further helps for robustness to different mesh densities. In practice, for each (S, T) source-target model pair, we also pass a point cloud sampled on S together with T through the decoder offset in a second pass to help the network cope with sparse meshes. Specifically, given a point set sampled on S, we predict per-point offsets and compute the above Chamfer and Earth Mover's losses between the resulting deformed point cloud and T. We denote these two losses as $L_{\texttt{CD}}^{\texttt{Points}}$ and $L_{\texttt{EMD}}^{\texttt{Points}}$. During testing, this second pass is not necessary and we only predict per-vertex offsets for S.

We note that we train our model with synthetic data where we always have access to 3D models. Thus, even if the target is a 2D image, we use the corresponding 3D model to compute the point cloud shape loss. During testing, however, we do not need access to any 3D target models, since the global shape features required for offset prediction are extracted from the 2D image only.

Symmetry Loss. Many man-made models exhibit global reflection symmetry and our goal is to preserve this during deformation. However, the mesh topology itself does not always guarantee to be symmetric, i.e., a symmetric chair does not always have symmetric vertices. Therefore, we propose to preserve shape symmetry by sampling a point cloud, $M(PC)$, on the mirrored deformed output and measure the point cloud shape loss with this mirrored point cloud as

$$\begin{aligned} L_{\texttt{sym}}(PC, PC_T) = &L_{CD}(M(PC), PC_T) \\ &+ L_{EMD}(M(PC), PC_T). \end{aligned} \quad (3)$$

We note that we assume the reflection symmetry plane of a source model to be known. In our experiments, we use 3D models from ShapeNet [1] which are already aligned such that the reflection plane coincides with the $xz-$ plane.

Mesh Laplacian Loss. To preserve the local geometric details in the source mesh and enforce smooth deformation across the mesh surface, we desire the Laplacian coordinates of the deformed mesh to be the same as the original source mesh. We define this loss as

$$L_{\texttt{lap}} = \sum_i ||Lap(S) - Lap(S')||_2. \quad (4)$$

where Lap is the mesh Laplacian operator, S and S' are the original and deformed meshes respectively.

Local Permutation Invariant Loss. Most traditional deformation methods (such as FFD) are prone to suffer from possible self-intersections that can occur during deformation (see Figure 4). To prevent such self-intersections, we present a novel *local permutation invariant loss*. Specifically, given a point p and a neighboring point at a distance δ to p, we would like to preserve the distance between these two neighboring points after deformation as well. Thus, we define

$$L_{\texttt{LPI}} = -\min\left(F(V + \delta) - F(V), \mathbf{0}\right). \quad (5)$$

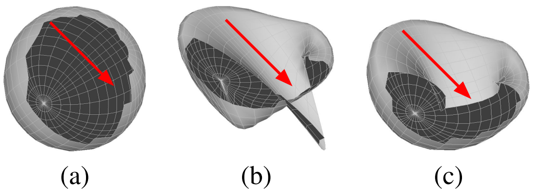


Figure 4: Self intersection. The red arrow is the deformation handle. (a) Original Mesh. (b) Deformation with self-intersection. (c) Plausible deformation.

where δ is a vector with a small magnitude and $\mathbf{0} = (0, 0, 0)$. In our experiments we define $\delta \in \{(\epsilon, 0, 0), (0, \epsilon, 0), (0, 0, \epsilon)\}$ where $\epsilon = 0.05$. The intuition behind of this is to preserve the local ordering of points in the source. We observe that the local permutation invariant loss helps to achieve smooth deformation across 3D space. Given all the losses defined above, we train 3DN with a combined loss of

$$L = \omega_{L_1} L^{\mathtt{Mesh}}_{\mathtt{CD}} + \omega_{L_2} L^{\mathtt{Mesh}}_{\mathtt{EMD}} + \omega_{L_3} L^{\mathtt{Points}}_{\mathtt{CD}} + \omega_{L_4} L^{\mathtt{Points}}_{\mathtt{EMD}} + \omega_{L_5} L_{\mathtt{sym}} + \omega_{L_6} L_{\mathtt{lap}} + \omega_{L_7} L_{\mathtt{LPI}}, \quad (6)$$

where $\omega_{L_1}, \omega_{L_2}, \omega_{L_3}, \omega_{L_4}, \omega_{L_5}, \omega_{L_6}, \omega_{L_7}$ denote the relative weighting of the losses.

4. Experiments

In this section, we perform qualitative and quantitative comparisons on shape reconstruction from 3D target models (Section 4.1) as well as single-view reconstruction (Section 4.2). We also conduct ablation studies of our method to demonstrate the effectiveness of the offset decoder architecture and the different loss functions employed. Finally, we provide several applications to demonstrate the flexibility of our method. More qualitative results and implementation details can be found in supplementary material.

Dataset. In our experiments, we use the ShapeNet Core dataset [1] which includes 13 shape categories and an official traning/testing split. We use the same template set of models as in [11] for potential source meshes. There are 30 shapes for each category in this template set. When training the 2D image-based target model, we use the rendered views provided by Choy et al. [2]. We note that we train a single network across all categories.

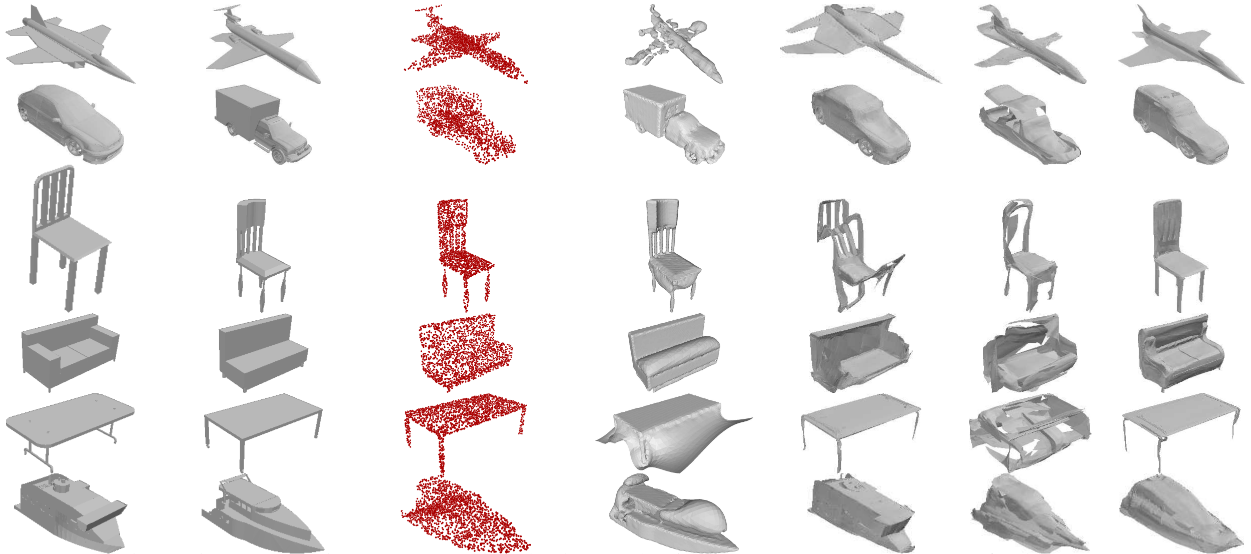
Template Selection. In order to sample source and target model pairs for 3DN, we train a PointNet based autoencoder to learn an embedding of the 3D shapes. Specifically, we represent each 3D shape as a uniformly sampled set of points. The encoder encodes the points as a feature vector and the decoder predicts the point positions from this feature vector (please refer to the supplementary material for details). Given the embedding composed of the features extracted by the encoder, for each target model candidate, we choose the nearest neighbor in this embedding as the source model. Source models are chosen from the aforementioned template set. No class label information is required during this procedure, however, the nearest neighbors are queried within the same category. When given a target 2D image for testing, if no desired source model is given, we use the point set generation network, PSGN [3], to generate an initial point cloud, and use its nearest neighbor in our embedding as the source model.

Evaluation Metrics. Given a source and target model pair (S, T), we utilize three metrics in our quantitative evaluations to compare the deformation output S' and the target T: 1) Chamfer Distance (CD) between the point clouds sampled on S' and T, 2) Earth Mover's Distance (EMD) between the point clouds sampled on S' and T, 3) Intersection over Union (IoU) between the solid voxelizations of S' and T. We normalize the outputs of our method and previous work into a unit cube before computing these metrics. We also evaluate the visual plausibility of our results by providing a large set of qualitative examples.

Comparison We compare our approach with state-of-the-art reconstruction methods. Specifically, we compare to three categories of methods: 1) learning-based surface generation, 2) learning-based deformation prediction, and 3) traditional surface reconstruction methods. We would like to note that we are solving a fundamentally different problem than surface generation methods. Even though, having a source mesh to start with might seem advantageous, our problem at hand is not easier since our goal is not only to generate a mesh similar to the target but also preserve certain properties of the source. Furthermore, our source meshes are obtained from a fixed set of templates which contain only 30 models per category.

4.1. Shape Reconstruction from Point Cloud

For this experiment, we define each 3D model in the testing split as target and identify a source model in the testing split based on the autoencoder embedding described above. 3DN computes per-vertex displacements to deform the source and keeps the source mesh topology fixed. We evaluate the quality of this mesh with alternative meshing techniques. Specifically, given a set of points sampled on the desired target model, we reconstruct a 3D mesh using Poisson surface reconstruction. As shown in Figure 5, this comparison demonstrates that even with a ground truth set of points, generating a mesh that preserves sharp features is not trivial. Instead, our method utilizes the source mesh connectivity to output a plausible mesh. Furthermore, we apply the learning-based surface generation technique of AtlasNet [7] on the uniformly sampled points on the target model. Thus, we expect AtlasNet only to perform surface generation without any deformation. We also compare



(a) Source Template (b) Target Mesh (c) Target Point Cloud (d)Poisson (e)FFD (f)AtlasNet (g) Ours

Figure 5: Given a source (a) and a target (b) model from the ShapeNet dataset, we show the deformed meshes obtained by our method (g). We also show Poisson surface reconstruction (d) from a set of points sampled on the target (c). We also show comparisons to previous methods of Jack et al. (e) and AtlasNet (f).

		plane	bench	box	car	chair	display	lamp	speaker	rifle	sofa	table	phone	boat	Mean
EMD	AtlasNet	3.46	3.18	4.20	2.84	3.47	3.97	3.79	3.83	2.44	3.19	3.76	3.87	2.99	3.46
	FFD	1.88	2.02	2.50	2.11	2.13	2.69	2.42	3.06	1.55	2.44	2.44	1.88	2.00	**2.24**
	Ours	0.79	1.98	3.57	1.24	1.12	3.08	3.44	3.40	1.79	2.06	1.34	3.27	2.27	2.26
CD	AtlasNet	2.16	2.91	6.62	3.97	3.65	3.65	4.48	6.29	0.98	4.34	6.01	2.44	2.73	3.86
	FFD	3.22	4.53	6.94	4.45	4.99	5.98	8.72	11.97	1.97	6.29	6.89	3.61	4.41	5.69
	Ours	0.38	2.40	5.26	0.90	0.82	5.59	8.74	9.27	1.52	2.55	0.97	2.66	2.77	**3.37**
IoU	AtlasNet	56.9	53.3	31.3	44.0	47.9	48.0	41.6	33.2	63.4	44.7	43.8	58.7	50.9	46.7
	FFD	29.0	42.3	28.4	21.1	42.2	27.9	38.9	52.5	31.9	34.7	43.3	22.9	47.7	35.6
	Ours	71.0	40.7	43.6	75.8	66.3	40.4	25.1	49.2	40.0	60.6	57.9	50.1	42.6	**51.1**

Table 1: Point cloud reconstruction results on ShapeNet core dataset. Metrics are mean Chamfer distance ($\times 0.001$, CD) on points, Earth Mover's distance ($\times 100$, EMD) on points and Intersection over Union (%, IoU) on solid voxelized grids. For both CD and EMD, the lower the better. For IoU, the higher the better.

to the method of Jack et al. [11] (FFD) which introduces a learning based method to apply free form deformation to a given template model to match an input image. This network consists of a module which predicts FFD parameters based on the features extracted from the input image. We retrain this module such that it uses the features extracted from the points sampled on the 3D target model. As shown in Figure 5, the deformed meshes generated by our method are higher quality than the previous methods. We also report quantitative numbers in Table 1. While AtlastNet achieves lower error based on Chamfer Distance, we observe certain artifacts such as holes and disconnected surfaces in their results. We also observe that our deformation results are smoother than FFD.

4.2. Single-view Reconstruction

We also compare our method to recent state-of-the-art single view image based reconstruction methods including Pixel2Mesh [25], AtlasNet [7] and FFD [11]. Specifically, we choose a target rendered image from the testing split and input to the previous methods. For our method, in addition to this target image, we also provide a source model selected from the template set. We note that the scope of our work is not single-view reconstruction, thus the comparison with Pixel2Mesh and AtlasNet is not entirely fair. However, both quantitative (see Table 2) and qualitative (Figure 6) results still provide useful insights. Though the rendered output of AtlasNet and Pixel2Mesh in Figure 6 are visually plausible, self-intersections and disconnected surfaces often exist in their results. Figure 7 illustrates this by rendering the output meshes in wireframe mode. Furthermore, as shown in Figure 7, while surface generation methods struggle to capture shape details such as chair handles and car wheels, our method preserves these details that reside in the source mesh.

		plane	bench	box	car	chair	display	lamp	speaker	rifle	sofa	table	phone	boat	Mean
EMD	AtlasNet	3.39	3.22	3.36	3.72	3.86	3.12	5.29	3.75	3.35	3.14	3.98	3.19	4.39	3.67
	Pxel2mesh	2.98	2.58	3.44	3.43	3.52	2.92	5.15	3.56	3.04	2.70	3.52	2.66	3.94	**3.34**
	FFD	2.63	3.96	4.87	2.98	3.38	4.88	7.19	5.04	3.58	3.70	3.56	4.11	3.86	4.13
	Ours	3.30	2.98	3.21	3.28	4.45	3.91	3.99	4.47	2.78	3.31	3.94	2.70	3.92	3.56
CD	AtlasNet	5.98	6.98	13.76	17.04	13.21	7.18	38.21	15.96	4.59	8.29	18.08	6.35	15.85	13.19
	Pixel2mesh	6.10	6.20	12.11	13.45	11.13	6.39	31.41	14.52	4.51	6.54	15.61	6.04	12.66	11.28
	FFD	3.41	13.73	29.23	5.35	7.75	24.03	45.86	27.57	6.45	11.89	13.74	16.93	11.31	16.71
	Ours	6.75	7.96	8.34	7.09	17.53	8.35	12.79	17.28	3.26	8.27	14.05	5.18	10.20	**9.77**
IoU	AtlasNet	39.2	34.2	20.7	22.0	25.7	36.4	21.3	23.2	45.3	27.9	23.3	42.5	28.1	30.0
	Pixel2mesh	51.5	40.7	43.4	50.1	40.2	55.9	29.1	52.3	50.9	60.0	31.2	69.4	40.1	47.3
	FFD	30.3	44.8	30.1	22.1	38.7	31.6	35.0	52.5	29.9	34.7	45.3	22.0	50.8	36.7
	Ours	54.3	39.8	49.4	59.4	34.4	47.2	35.5	45.3	57.62	60.7	31.3	71.5	46.5	**48.7**

Table 2: Quantitative comparison on ShapeNet rendered images. Metrics are CD ($\times 0.001$), EMD ($\times 100$) and IoU (%).

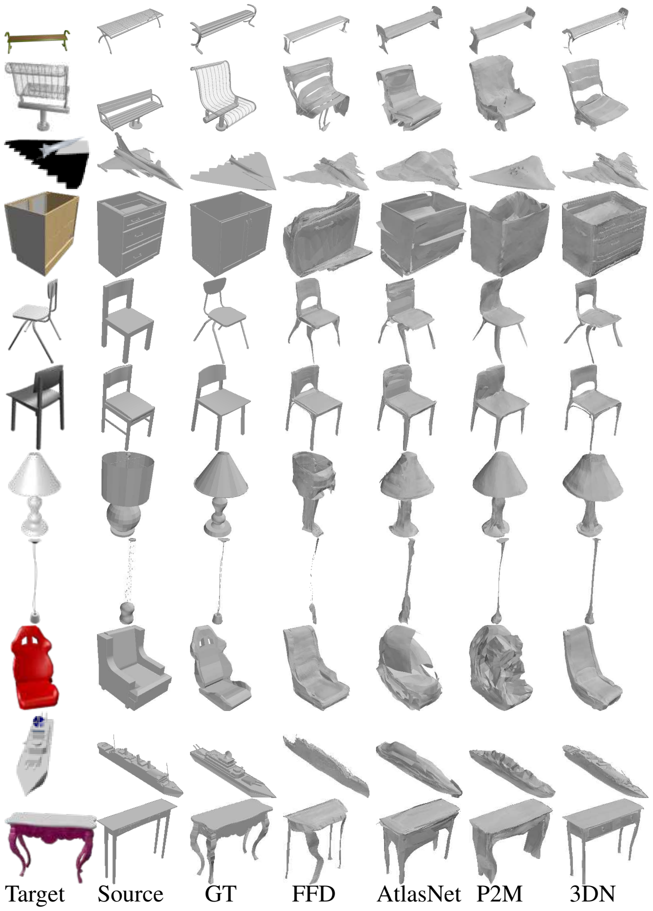


Figure 6: Given a target image and a source, we show deformation results of FFD, AtlasNet, Pixel2Mesh (P2M), and 3DN. We also show the ground truth target model (GT).

Evaluation on real images. We further evaluate our method on real product images that can be found online. For each input image, we select a source model as described before and provide the deformation result. Even though our method has been trained only on synthetic images, we observe that it generalizes to real images as seen in Figure 8. AtlasNet and Pixel2Mesh fail in most cases, while our method is able to generate plausible results by taking advantages of source meshes.

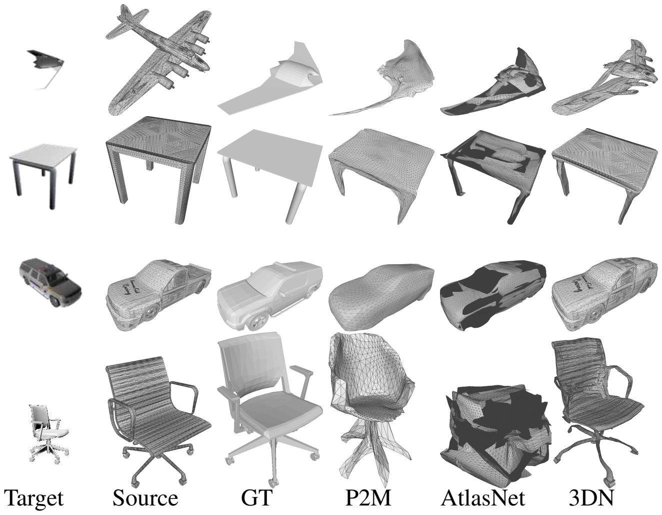


Figure 7: For a given target image and source model, we show ground truth model and results of Pixel2Mesh (P2M), AtlasNet, and our method (3DN) rendered in wire-frame mode to better judge the quality of the meshes. Please zoom into the PDF for details.

4.3. Ablation Study

We study the importance of different losses and the offset decoder architecture on ShapeNet chair category. We compare our final model to variants including 1) 3DN without the symmetry loss, 2) 3DN without the mesh Laplacian loss, 3) 3DN without the local permutation invariance loss, and 4) fusing global features with midlayer features instead of the original point positions (see the supplemental material for details).

We provide quantitative results in Table 3. Symmetry loss helps the deformation to produce plausible symmetric shapes. Local permutation and Laplacian losses help to obtain smoothness in the deformation field across 3D

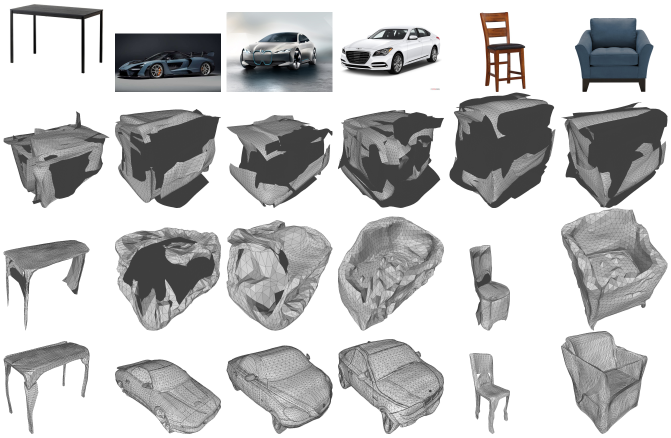


Figure 8: Qualitative results on online product images. The first row shows the images scrapped online. Second and third row are results of AtlasNet and Pixel2Mesh respectively. Last row is our results.

	CD	EMD	IoU
3DN	**4.50**	**2.06**	**41.0**
-Symmetry	4.78	2.73	36.7
-Mesh Laplacian	4.55	2.08	39.8
-Local Permutation	5.31	2.96	35.4
Midlayer Fusion	6.63	3.03	30.9

Table 3: Quantitative comparison on ShapeNet rendered images. '-x' denotes without x loss. Metrics are CD ($\times 1000$), EMD ($\times 0.01$) and IoU (%).

space and along the mesh surface. However, midlayer fusion makes the network hard to converge to a valid deformation space.

4.4. Applications

Random Pair Deformation. In Figure 9 we show deformation results for randomly selected source and target model pairs. While the first column of each row is the source mesh, the first row of each column is the target. Each grid cell shows deformation results for the corresponding source-target pair.

Shape Interpolation. Figure 10 shows shape interpolation results. Each row shows interpolated shapes generated from the two targets and the source mesh. Each intermediate shape is generated using a weighted sum of the global feature representations of the target shapes. Notice how the interpolated shapes gradually deform from the first to the second target.

Shape Inpainting. We test our model trained in Section 4.1 on targets in the form of partial scans produced by RGBD data [22]. We provide results in Figure 11 with different selection of source models. We note that AtlastNet fails on such partial scan input.

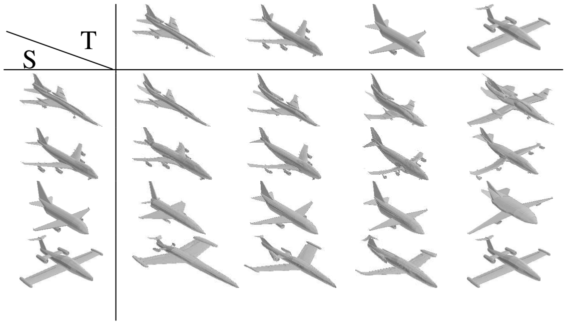


Figure 9: Deformation with different source-target pairs. 'S' and 'T' denote source meshes and target meshes respectively.

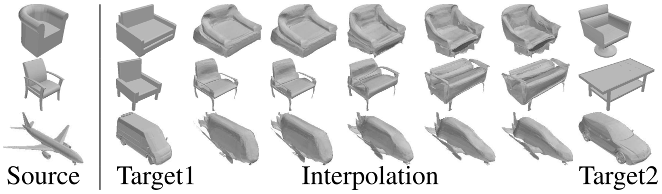


Figure 10: Shape interpolation.

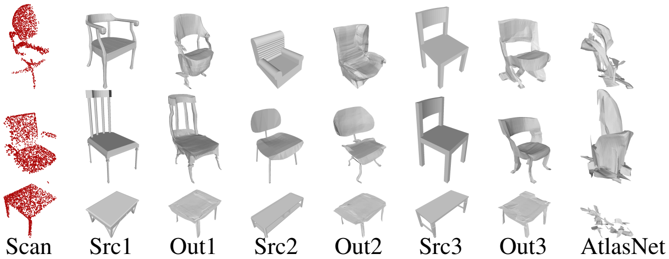


Figure 11: Shape inpainting with real point cloud scan as input. Src means source mesh and 'out' is the corresponding deformed mesh.

5. Conclusion

We have presented 3DN, an end-to-end network architecture for mesh deformation. Given a source mesh and a target which can be in the form of a 2D image, 3D mesh, or 3D point clouds, 3DN deforms the source by inferring pervertex displacements while keeping the source mesh connectivity fixed. We compare our method with recent learning based surface generation and deformation networks and show superior results. Our method is not without limitations, however. Certain deformations indeed require to change the source mesh topology, e.g., when deforming a chair without handles to a chair with handles. If large holes exist either in the source or target models, Chamfer and Earth Mover's distances are challenging to compute since it is possible to generate many wrong point correspondences.

In addition to addressing the above limitations, our future work include extending our method to predict mesh texture by taking advantages of differentiable renderer [13].

References

[1] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. Shapenet: An information-rich 3d model repository. *arxiv*, 2015.

[2] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *ECCV*, 2016.

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

[4] R. Gal, O. Sorkine, N. J. Mitra, and D. Cohen-Or. iwires: An analyze-and-edit approach to shape manipulation. *ACM Trans. on Graph.*, 28(3), 2009.

[5] R. Girdhar, D. Fouhey, M. Rodriguez, and A. Gupta. Learning a predictable and generative vector representation for objects. In *ECCV*, 2016.

[6] T. Groueix, M. Fisher, V. G. Kim, B. Russell, and M. Aubry. 3d-coded : 3d correspondences by deep deformation. In *ECCV*, 2018.

[7] T. Groueix, M. Fisher, V. G. Kim, B. Russell, and M. Aubry. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *CVPR*, 2018.

[8] C. Häne, S. Tulsiani, and J. Malik. Hierarchical surface prediction for 3d object reconstruction. In *3DV*, 2017.

[9] Q. Huang, H. Wang, and V. Koltun. Single-view reconstruction via joint analysis of image and shape collections. *ACM Trans. Graph.*, 2015.

[10] Q. Huang, W. Wang, and U. Neumann. Recurrent slice networks for 3d segmentation on point clouds. *arXiv preprint arXiv:1802.04402*, 2018.

[11] D. Jack, J. K. Pontes, S. Sridharan, C. Fookes, S. Shirazi, F. Maire, and A. Eriksson. Learning free-form deformations for 3d object reconstruction. In *ACCV*, 2018.

[12] A. Kanazawa, S. Kovalsky, R. Basri, and D. W. Jacobs. Learning 3d deformation of animals from 2d images. In *Eurographics*, 2016.

[13] H. Kato, Y. Ushiku, and T. Harada. Neural 3d mesh renderer. In *CVPR*, 2018.

[14] A. Kurenkov, J. Ji, A. Garg, V. Mehta, J. Gwak, C. Choy, and S. Savarese. Deformnet: Free-form deformation network for 3d shape reconstruction from a single image. *arXiv preprint arXiv:1708.04672*, 2017.

[15] C. Niu, J. Li, and K. Xu. Im2struct: Recovering 3d shape structure from a single rgb image. In *CVPR*, 2018.

[16] J. K. Pontes, C. Kong, S. Sridharan, S. Lucey, A. Eriksson, and C. Fookes. Image2mesh: A learning framework for single image 3d reconstruction. In *ACCV*, 2017.

[17] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017.

[18] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[19] A. Sinha, A. Unmesh, Q. Huang, and K. Ramani. Surfnet: Generating 3d shape surfaces using deep residual networks. In *CVPR*, 2018.

[20] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Eurographics*, 2004.

[21] X. Sun, J. Wu, X. Zhang, Z. Zhang, C. Zhang, T. Xue, J. B. Tenenbaum, and W. T. Freeman. Pix3d: Dataset and methods for single-image 3d shape modeling. In *CVPR*, 2018.

[22] M. Sung, V. G. Kim, R. Angst, and L. Guibas. Data-driven structural priors for shape completion. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 2015.

[23] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *ICCV*, 2017.

[24] S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *CVPR*, 2017.

[25] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. *arXiv preprint arXiv:1804.01654*, 2018.

[26] P.-S. Wang, C.-Y. Sun, Y. Liu, and X. Tong. Adaptive o-cnn: A patch-based deep representation of 3d shapes. *arXiv preprint arXiv:1809.07917*, 2018.

[27] W. Wang, Q. Huang, S. You, C. Yang, and U. Neumann. Shape inpainting using 3d generative adversarial network and recurrent convolutional networks. In *ICCV*, 2017.

[28] W. Wang, R. Yu, Q. Huang, and U. Neumann. Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In *CVPR*, 2018.

[29] J. Wu, Y. Wang, T. Xue, X. Sun, W. T. Freeman, and J. B. Tenenbaum. MarrNet: 3D Shape Reconstruction via 2.5D Sketches. In *NIPS*, 2017.

[30] J. Wu, C. Zhang, X. Zhang, Z. Zhang, W. T. Freeman, and J. B. Tenenbaum. Learning shape priors for single-view 3d completion and reconstruction. In *NIPS*, 2018.

[31] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, pages 1912–1920, 2015.

[32] K. Xu, H. Zheng, H. Zhang, D. Cohen-Or, L. Liu, and Y. Xiong. Photo-inspired model-driven 3d object modeling. *ACM Trans. Graph.*, 30(4):80:1–80:10, 2011.

[33] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In *NIPS*, 2016.

[34] G. Yang, Y. Cui, S. Belongie, and B. Hariharan. Learning single-view 3d reconstruction with limited pose supervision. In *ECCV*, 2018.

[35] Y. Yang, C. Feng, Y. Shen, and D. Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *CVPR*, 2018.

[36] M. E. Yumer and N. J. Mitra. Learning semantic deformation flows with 3d convolutional networks. In *ECCV*, 2016.

[37] R. Zhu, H. Kiani Galoogahi, C. Wang, and S. Lucey. Rethinking reprojection: Closing the loop for pose-aware shape reconstruction from a single image. In *ICCV*, 2017.

[38] C. Zou, E. Yumer, J. Yang, D. Ceylan, and D. Hoiem. 3d-prnn: Generating shape primitives with recurrent neural networks. In *ICCV*, 2017.