ShapeFlow: Learnable Deformations Among 3D Shapes

Chiyu "Max" Jiang*
UC Berkeley
chiyu.jiang@berkeley.edu

Jingwei Huang*
Stanford University
jingweih@stanford.edu

Andrea Tagliasacchi
Google Brain
taglia@google.com

Leonidas Guibas
Stanford University
guibas@stanford.edu

Abstract

We present ShapeFlow, a flow-based model for learning a deformation space for entire classes of 3D shapes with large intra-class variations. ShapeFlow allows learning a multi-template deformation space that is agnostic to shape topology, yet preserves fine geometric details. Different from a generative space where a latent vector is directly decoded into a shape, a deformation space decodes a vector into a continuous flow that can advect a source shape towards a target. Such a space naturally allows the disentanglement of geometric style (coming from the source) and structural pose (conforming to the target). We parametrize the deformation between geometries as a learned continuous flow field via a neural network and show that such deformations can be guaranteed to have desirable properties, such as bijectivity, freedom from self-intersections, or volume preservation. We illustrate the effectiveness of this learned deformation space for various downstream applications, including shape generation via deformation, geometric style transfer, unsupervised learning of a consistent parameterization for entire classes of shapes, and shape interpolation.

1 Introduction

Learning a shared representation space for geometries is a central task in 3D Computer Vision and in Geometric Modeling as it enables a series of important downstream applications, such as retrieval, reconstruction, and editing. For instance, *morphable models* [1] is a commonly used representation for entire classes of shapes with small intra-class variations (i.e., faces), allowing high quality geometry generation. However, morphable models generally assume a *shared* topology and even the same mesh connectivity for all represented shapes, and are thus less extensible to general shape categories with large intra-class variations. Therefore, such approaches have limited applications beyond collections with a shared structure such as humans [1, 2] or animals [3].

In contrast, when trained on large shape collections (e.g., ShapeNet [4]), 3D generative models are not only able to learn a shared latent space for entire classes of shapes (e.g., chairs, tables, airplanes), but also capture large geometric variations between classes. A main area of focus in this field has been developing novel geometry decoders for these latent representations. These generative spaces allow the mapping from a latent code $z \in \mathbb{R}^c$ to some geometric representation of a shape, examples being voxels [5, 6], meshes [7, 8], convexes [9, 10], or implicit functions [11, 12]. Such latent spaces are

*Equal Contribution.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

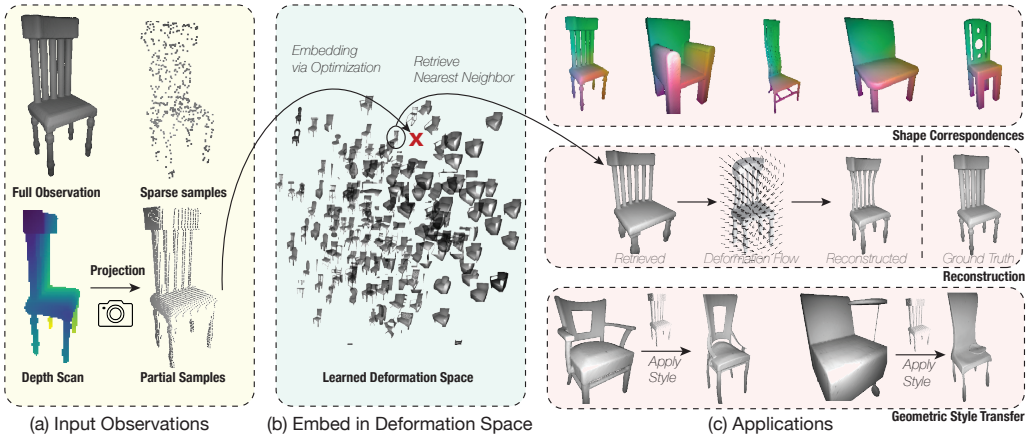

Figure 1: Schematic for learning a deformation space using `ShapeFlow`. (a)Our input is either a sparse point cloud, or a depth map converted into a point cloud. (b) The visualization of the learned latent embedding (2D PCA) of sample shapes in the training set. `ShapeFlow` learns a geometrically meaningful embedding of geometries based on deformation distances in an unsupervised manner. (c) The unsupervised deformation space facilitates various downstream applications, including shape correspondences, reconstruction, and style transfer.

generally smooth and allow interpolation or deformation between arbitrary objects represented in this encoding. However, the shape generation quality is highly dependent on the decoder performance and generally imperfect. While some decoder architectures are able to produce higher quality geometries, auto-encoded shapes *never exactly match* their inputs, leading to a loss of fine geometric details.

In this paper we introduce a different approach to shape generation based on continuous flows between shapes that we term `ShapeFlow`. The approach views the shape generation process from a new perspective – rather than learning a generative space where a learned decoder $\mathcal{F}_\theta$ directly maps a latent code $z_i \in \mathbb{R}^c$ to the shapeX_i as $X_i = \mathcal{F}_\theta(z_i)$, `ShapeFlow` learns a *deformation space* facilitated by a learned deformer $\mathcal{D}_\theta$, where a novel shape $X_{i \leftarrow j}$ is acquired by deforming one of many possible *template shapes* $X_j \in \mathcal{X}$ via this learned deformer: $X_{i \leftarrow j} = \mathcal{D}_\theta(X_j; z_j, z_i)$, where $z_i, z_j \in \mathbb{R}^c$ are the latent codes corresponding of X_i and X_j.

This deformation-centric view of shape generation has various unique properties. First, a deformation space, compared to a generative space, naturally disentangles geometry *style* from *structure*. Style comes from the choice of source shape X_j, which also includes the shape topology and mesh connectivity. Structure includes the general placement of different parts, such as limb positioning in a human figure (i.e., pose), height and width of chair parts etc. Second, unlike template-based mesh generation frameworks such as [13, 14, 7], whose generated shapes are inherently limited by the the template topology, a deformation space allows a multi-template scenario where *each* of the source shapes $X_j \in \mathcal{X}$ can be viewed as a template. Also, unlike volumetric decoders that require a potentially computationally intensive step for (e.g., querying a large number of sample points), `ShapeFlow` directly outputs a mesh (or a point cloud) through deforming the source shape. Finally, by routing the deformations through a common waypoint in this space, we can learn a *shared template* for all geometries of the same class, despite differences in meshing or topology, allowing unsupervised learning of dense correspondences between all shapes within the same class.

The learned deformation function Φ_θ^{ij} deforms the *template* shape X_j into $X_{i \leftarrow j}$ so that it is geometrically close to the target shape X_i. Our deformation function is based on neurally parametrized 3D vector fields or *flows* that locally advect a template shape towards its destination. This novel way of modeling deformations has various innate advantages compared to existing methods. We show that deformation induced by a flow naturally prevents *self-intersections*. Furthermore, we demonstrate that we can parametrize a divergence-free flow field effectively using a neural network, which ensures *volume conservation* during the deformation process. Finally, `ShapeFlow` ensures path invertibility ($\Phi_\theta^{ij} = (\Phi_\theta^{ji})^{-1}$), and therefore also identity preservation ($X_i = \Phi_\theta^{ii}(X_i)$). Compared to traditional deformation parameterizations in computer graphics such as control handles [15, 16]

and control cages [17, 18, 19], ShapeFlow is a flow-model realized by a neural network, allowing a more fine grained deformation without requiring user intervention.

In summary, our main contributions are:

1. We propose a flow-based deformation model via a neural network that allows exact preservation of identity, good preservation of local geometric features, and disentangles geometry style and structure.
2. We show that our deformations by design prevent self-intersections and can preserve volume.
3. We demonstrate that we can learn a common *template* for a class of shapes through which we can derive dense correspondences.
4. We apply our method to interpolate shapes in different poses, producing smooth interpolation between key frames that can be used for animation and content creation.

2 Related work

Traditionally, shape representation in 3D computer vision roughly falls into two categories, *template-based* representation and *template-free* representation. In contrast, ShapeFlow fills a gap in between – it can be viewed as a *multi-template space*, where the source topology can be based on any of the training shapes, and where a very general deformation model is adopted.

Template-based representations. These methods generally assume a fixed topology for all modelled geometries. Morphable models [1] is a commonly used representation for entire classes of shapes with very small intra-class variations, such as faces [1, 20, 21, 22, 23, 24], heads [25, 26], human bodies [27, 28, 2], and even animals [3, 29]. Morphable models generally assume a shared topology and even the same mesh connectivity for all represented shapes, which restricts its use to few shape categories. Recently, neural networks have been employed to generate 3D shapes via morphable models [30, 31, 14, 32, 33, 3, 29]. Some recent work has extended the template-based approach to shapes with larger variations [13, 7, 34, 35], but the generated results are polygonal meshes that often contain self-intersections and are not-watertight.

Template-free representations. These methods generally produce a volumetric implicit representation for the geometries rather than directly representing the surface under certain surface parameterizations, thus allowing the same model to model geometries across different topologies, with potentially large geometric variations. Earlier works in this line utilize voxel representations [36, 37]. Recently, the use of continuous implicit function decoders [38, 39, 11] has been popularized due to its strong representation capacity for more detailed geometry. Similar ideas are extended to represent color, light field, and other scene related properties [40, 41], and coupled with spatial [42, 43] or spatio-temporal [44] latent grid structures to extend to larger scenes and domains. Still, these approaches lack the fine structures of real geometric models.

Shape deformation. Parametrizing the space of admissible deformations in a set of shapes with diverse topologies is a challenging problem. Directly predicting offsets for each mesh vertex with insufficient regularization will lead to non-physical deformations such as self-intersections. In computer graphics, geometry deformation is usually parameterized using a set of deformation handles [15] or deformation cages [17, 18, 19]. Surface-based energies are usually optimized in the deformation process [45, 46, 16, 47] to maintain rigidity, isometry, or other desired geometric properties. Similar to our work, earlier work by [48] proposed deforming objects using time-dependent divergence-free vector fields, though the deformations are not learnable. More recently, learned deformation models have been proposed, directly predicting vertex offsets [49], control point offsets [50], or control cage deformations [51]. Different from our end-to-end deformation setting, the graphics approaches are typically aimed at interactive and incremental shape editing applications.

Flow models. Flow models have traditionally been used in machine learning for learning generative models for a given data distribution. Some examples of flow models include RealNVP [52] and Masked Auto-Regressive Flows [53]; these generally involve a discrete number of learned transformations. Continuous normalizing flow models have also been recently proposed [54, 55], and our method is mainly inspired by these works. They create bijective mappings via a learned advection process, and are trained using a differential Ordinary Differential Equation (ODE) solver. PointFlow [56] and OccFlow [57] are similar to our approach in using such learned flow dynamics for modeling geometry. However, PointFlow [56] maps point clouds corresponding to geometries

to a learned prior distribution while ShapeFlow directly learns the deformations function between geometries, bypassing a prior distribution and better preserves geometric details. OccFlow [57] only models the temporal deformation sequence for one object, while ShapeFlow learns a deformation space for entire classes of geometries.

3 Method

Consider a set of N shapes $\mathcal{X} = \{X_1, X_2, \cdots, X_N\}$. Each shape is represented by a polygonal mesh $X_i = \{\mathcal{V}_i, \mathcal{E}_i\}$, where $\mathcal{V}_i = \{v_1, v_2, \cdots, v_{n_i}\}$ is an ordered set of n_i points that represent the vertices of the polygonal mesh. For each point $v \in \mathcal{V}_i$, we have $v \in \mathbb{R}^d$. $\mathcal{E} = \{e_1, e_2, \cdots, e_{m_i}\}$ is a set of m_i polygonal elements, where each element $e \in \mathcal{E}_i$ indexes into a set of vertices $v \in \mathcal{V}_i$. For one-way deformations, we seek a *mapping* $\Phi_\theta^{ij} : \mathbb{R}^d \mapsto \mathbb{R}^d$ that minimizes the geometric distance between the deformed source shape $\Phi_\theta^{ij}(X_i)$ and the target shape X_j:

$$\arg\min_\theta \quad \mathcal{C}(\Phi_\theta^{ij}(X_i), X_j)\,, \tag{1}$$

where $\mathcal{C}(X_i, X_j)$ is the *symmetric* Chamfer distance between two shapes X_i, X_j. Note the mapping operates on the vertices $\mathcal{V}_i$, while retaining the mesh connectivity expressed by $\mathcal{E}_i$. As in previous work [58, 38], since mesh-to-mesh Chamfer distance computation is expensive, we proxy it using the point set to point set Chamfer distance between uniform point samples on the meshes. Furthermore, in order to learn a symmetric deformation space, we optimize for maps that minimize the *symmetric* deformation distance:

$$\min_\theta \quad \mathcal{C}(\Phi_\theta^{ij}(X_i), X_j) + \mathcal{C}(X_i, \Phi_\theta^{ji}(X_j))\,. \tag{2}$$

We define such maps as an advection process via a *flow* function $\boldsymbol{f}_\theta(x(t), t)$, where we associate intermediate deformations with an interpolation parameter $t \in [0, 1]$. For any pair of shapes i, j:

$$\Phi_\theta^{ij}(\boldsymbol{x}_i \in X_i) = \boldsymbol{x}_i(1), \quad \boldsymbol{x}_i(T) = \boldsymbol{x}_i + \int_0^T \boldsymbol{f}_\theta^{ij}(\boldsymbol{x}_i(t), t)\, dt\,. \tag{3}$$

Intersection-free deformations. Not introducing self-intersections is a key property in shape deformation, since self-intersecting deformations are not physically plausible. In Proposition 1 (**supplementary material**), we prove that this property is *algebraically* satisfied in our formulation. Note that this property holds under the assumption of perfect integration. Errors in numerical integration will lead to its violation. However, we will empirically show in Sec. C.2 (**supplementary material**) that this can be controlled by bounding the numerical integration error.

Invertible deformations. For any pair of shapes, it would be ideal if performing a deformation of X_i into X_j, and then back to X_i, would recover X_i exactly. We want the deformation to be *lossless* for identity transformations, or, more formally, $\Phi^{ij}(\Phi^{ji}(\mathbf{x})) = \mathbf{x}$. In Proposition 3 (**supplementary material**), we derive a condition on $\boldsymbol{f}_\theta$ that is *sufficient* to ensure bijectivity $\forall t \in (0, 1]$:

$$\boldsymbol{f}_\theta^{ji}(\boldsymbol{x}, t) = -\boldsymbol{f}_\theta^{ij}(\boldsymbol{x}, 1 - t), \quad t \in (0, 1]\,. \tag{4}$$

3.1 Deformation flow field

At the core of the learned deformations (3) is a learnable flow field $\boldsymbol{f}(\boldsymbol{x}, t)$. We start by assigning latent codes $\boldsymbol{z}_i, \boldsymbol{z}_j \in \mathbb{R}^c$ to the shapes i, j, and then define the flow as:

$$\boldsymbol{f}_\theta^{ij}(\boldsymbol{x}, t) = \underbrace{\boldsymbol{h}_\eta(\boldsymbol{x}, \boldsymbol{z}_i + t(\boldsymbol{z}_j - \boldsymbol{z}_i))}_{\textbf{flow function}} \cdot \underbrace{\boldsymbol{s}_\sigma((\boldsymbol{z}_j - \boldsymbol{z}_i)/||\boldsymbol{z}_j - \boldsymbol{z}_i||_2)}_{\textbf{sign function}} \cdot \underbrace{||\boldsymbol{z}_j - \boldsymbol{z}_i||_2}_{\textbf{flow magnitude}}\,, \tag{5}$$

where $\theta = \{\eta, \sigma\}$ are trainable parameters of a *neural network*. Note the same deformation function can be *shared* for all pairs of shapes (i, j), and that this flow satisfies the invertibility condition (4).

Flow function. The function $\boldsymbol{h}_\eta(\boldsymbol{x}, \boldsymbol{z})$, receives in input the spatial coordinates $\boldsymbol{x}$ and a latent code $\boldsymbol{z}$. When deforming from shape i to shape j, the latent code $\boldsymbol{z}$ linearly interpolates between the two endpoints. $\boldsymbol{h}_\eta(\cdot) : \mathbb{R}^{d+c} \mapsto \mathbb{R}^d$ is a fully-connected neural network with weights η.

Sign function. The sign function $\boldsymbol{s}_\sigma(\boldsymbol{z})$, receives the normalized direction for the vector from $\boldsymbol{z}_i$ to $\boldsymbol{z}_j$. The sign function has the additional requirement that it be symmetric, which can be satisfied either by *fully-connected neural networks* with learnable parameters σ, with zero bias and symmetric activation function (e.g., `tanh`), or by construction via the hub-and-spokes model of Section 3.2.

Flow magnitude. With this regularization, we ensure that the distance within the latent space is directly proportional to the amount of required deformation between two shapes, and obtain several properties:

- *Consistency* of the latent space, which ensures deforming half way from i to j is equivalent to deforming all-way from i to the latent code half-way between i and j:

$$\int_0^\alpha \boldsymbol{f}_\theta^{ij}(\boldsymbol{x}(t),t)dt = \int_0^1 \boldsymbol{f}_\theta^{ik}(\boldsymbol{x}(t),t)dt, \text{ where } \boldsymbol{z}_k = \boldsymbol{z}_i + \alpha(\boldsymbol{z}_j - \boldsymbol{z}_i)\,.$$

- *Identity* preservation $\Phi_\theta^{ii}(\boldsymbol{x}) = \boldsymbol{x}$:

$$\Phi_\theta^{ii}(\boldsymbol{x}) = \boldsymbol{x} + \int_0^1 \cancelto{=0}{\boldsymbol{f}_\theta^{ii}(\boldsymbol{x},t)dt} = \boldsymbol{x}\,.$$

Implicit regularization: volume conservation. By learning a divergence-free flow field for the deformation process, we show that the volume of any enclosed mesh can be conserved through the deformation sequence; see Proposition 4 (**supplementary material**). While we could penalize for divergence change via a loss, resulting in approximate volume conservation, we show how this hard-constraint can be implicitly and exactly satisfied without resorting to auxiliary loss functions. Based on Gauss's theorem, the volume integral of the flow divergence is equal to the surface integral of flux, which amounts to zero for solenoidal flows. Additional, any divergence-free vector field can be represented as the curl of a vector potential. This allows us to parameterize a *strictly* divergence-free flow field by first parameterizing a C^2 vector field as the vector potential. In particular, we parameterize the flow as $\boldsymbol{h}_\eta(\cdot) = \nabla \times \boldsymbol{g}_\eta(\cdot)$, with $\boldsymbol{g}_\eta$ using a fully-connected network: $\mathbb{R}^{3+c} \mapsto \mathbb{R}^3$. Since the curl operator $\nabla\times$ is a series of first-order spatial derivatives, it can be efficiently calculated via a sum of the first-order derivatives with respect to the input layer for (x, y, z), computed through a single backpropagation step; refer to the architecture in Sec. B.1 (**supplementary material**).

Implicit regularization: symmetries. Given that many geometric objects have a natural plane/axis/point of symmetry, being able to enforce implicit symmetry is a desired quality for the deformation network. We can parameterize the flow function $\boldsymbol{h}_\eta(\cdot)$ by first parameterizing a $\boldsymbol{g}_\eta : \mathbb{R}^{d+1} \mapsto \mathbb{R}^d$. Without loss of generality, assume $d = 3$, and let yz be the plane of symmetry:

$$\begin{cases} \boldsymbol{h}_\eta^{(x)}(x,y,z,t) = (\boldsymbol{g}_\eta^{(x)}(x,y,z,t) - \boldsymbol{g}_\eta^{(x)}(-x,y,z,t))/2 & \text{[anti-symmetric part]}\,, \\ \boldsymbol{h}_\eta^{(y,z)}(x,y,z,t) = (\boldsymbol{g}_\eta^{(y,z)}(x,y,z,t) + \boldsymbol{g}_\eta^{(y,z)}(-x,y,z,t))/2 & \text{[symmetric part]}\,, \end{cases} \tag{6}$$

where the superscript denotes the x, y, z components of the vector output.

Explicit regularization: surface metrics. Additionally, surface metrics such as rigidity and isometry can be explicitly enforced via an auxiliary loss term to the overall loss function. A simple isometry constraint can be enforced by penalizing the change in edge lengths of the original mesh through the transformations, similar to the stretch regularization in [59, 60].

Implementation. We use a modified version of IM-NET [11] as the backbone flow model where we adjust the model with different number of hidden and output nodes. We defer discussions about the model architecture and training details to Sec. B.1 (**supplementary material**)

3.2 Hub-and-spoke deformation

Given a set of N training shapes: $\mathcal{X} = \{X_1, X_2, \cdots, X_N\}$, we train the deformer by picking random pairs of shapes from the set. There are two strategies for learning the deformation, either by directly deforming between each pair of shapes, or deforming each pair of shapes via a canonical latent shape corresponding to a "hub" latent code. Additionally, we use an encoder-less approach (i.e., an auto-decoder [39]) where we initialize N random latent codes $\mathcal{Z} = \{\boldsymbol{z}_1, \cdots, \boldsymbol{z}_N\}$ from $\mathcal{N}(0, 0.1)$,

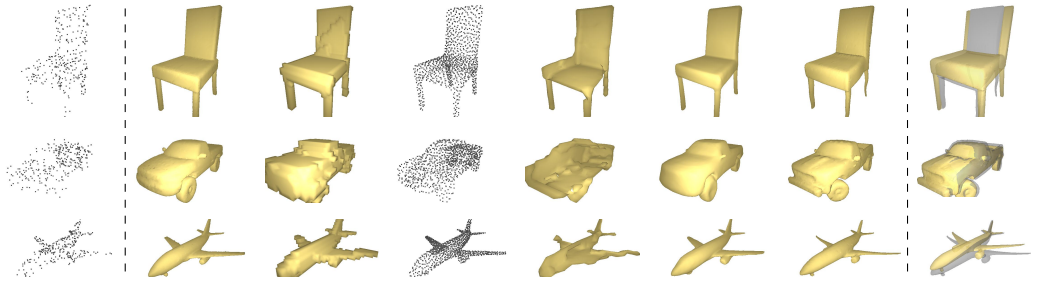

Figure 2: Qualitative comparison of mesh reconstruction from sparse point clouds as inputs. The shapes generated by ShapeFlow are able to preserve CAD-like geometric features (i.e. style) while faithfully aligning with the input observations (i.e. structure). The retrieved model is overlaid with the deformed model.

corresponding to each training shape. $\forall \boldsymbol{z} \in \mathcal{Z}, \boldsymbol{z} \in \mathbb{R}^c$. The latent codes are jointly optimized, along with the network parameters θ. Additionally, we define a "hub" latent vector as $\boldsymbol{z}_0 = [0, \cdots, 0] \in \mathbb{R}^d$. Under the hub-and-spokes deformation model, the training process amounts to finding:

$$\arg\min_{\theta, \mathcal{Z}} \sum_{(i,j) \in \mathcal{X} \times \mathcal{X}} \mathcal{C}(\Phi_\theta^{0j}(\Phi_\theta^{i0}(X_i)), X_j) + \mathcal{C}(X_i, \Phi_\theta^{0i}(\Phi_\theta^{j0}(X_j))) \,. \tag{7}$$

A visualization for the learned latent space via the hub-and-spokes model is shown in Fig. 1(b). With hub-and-spokes training, we can define the sign function $\boldsymbol{s}(\cdot)$ (Sec. 3.1) simply to produce $+1$ for the path towards the zero hub and -1 for the path from the hub, without the need of parameters.

3.3 Encoder-free embedding

We adopt an encoder-free scheme for learning the deformation space, as well as embedding new observations into the deformation space. After we acquire a learned deformation space by training with the hub-and-spokes approach, we are able to embed *new* observations of point clouds into the learned latent space by optimizing for the latent code that minimizes the deformation error of random shapes in the original deformation space to the new observation. Again, this "embedding via optimization" approach is similar to the auto-decoder approach in [39, 61]. The embedding z_n of a new point cloud X_n amounts to seeking:

$$\arg\min_{z_n} \sum_{i \in \mathcal{X}} \mathcal{C}(\Phi_\theta^{0i}(\Phi_\theta^{n0}(X_n)), X_i) + \mathcal{C}(X_n, \Phi_\theta^{0n}(\Phi_\theta^{i0}(X_i))) \,. \tag{8}$$

4 Experiments

4.1 ShapeNet deformation space

As a first experiment, we learn the deformation space for entire classes of shapes from ShapeNet [4], and illustrate two downstream applications for such a deformation space: shape generation by deformation, and shape canonicalization. Specifically, we experiment on three representative shape categories in ShapeNet: chair, airplane and car. For each category, we follow the official train/test/validation split for the data. We preprocess the geometries into watertight manifolds using the preprocessing pipeline in [38], and further simplify the meshes to $1/10$th of the original number of vertices using [63]. The deformation space is learned by deforming random pairs of objects using a hub-and-spokes deformation approach (as described in Section 3.2). More training details for learning the deformation space can be found in Section B.2 (**supplementary material**).

4.1.1 Surface reconstruction by template deformation

The learned deformation space can be used for reconstructing objects based on input observations. A schematic for this process is provided in Fig. 1: a new observation X_n, in the form of a point cloud, can be embedded into a latent code z_n the latent deformation space according to Eqn. 8. The top-k

category	Chamfer-L_1 (↓)					IoU (↑)					Normal Consistency (↑)				
	DMC	OccFlow	PSGN	R2N2	ShapeFlow	DMC	OccFlow	PSGN	R2N2	ShapeFlow	DMC	OccFlow	PSGN	R2N2	ShapeFlow
airplane	0.0969	0.0711	0.0976	0.1525	0.0858	0.5762	0.7158	-	0.4453	0.6156	0.8134	0.8857	-	0.6546	0.8387
car	0.1729	0.1218	0.1294	0.1949	0.1388	0.7182	0.8029	-	0.6728	0.6644	0.8222	0.8647	-	0.6979	0.7690
chair	0.1284	0.1302	0.1756	0.1851	0.1888	0.6250	0.6513	-	0.5166	0.4390	0.8348	0.8593	-	0.6599	0.7647
mean	0.1328	0.1077	0.1342	0.1775	0.1378	0.6398	0.7233	-	0.5449	0.5730	0.8235	0.8699	-	0.6708	0.7908

Table 1: Quantitative evaluation of shape reconstruction performance for ShapeNet [4] models.

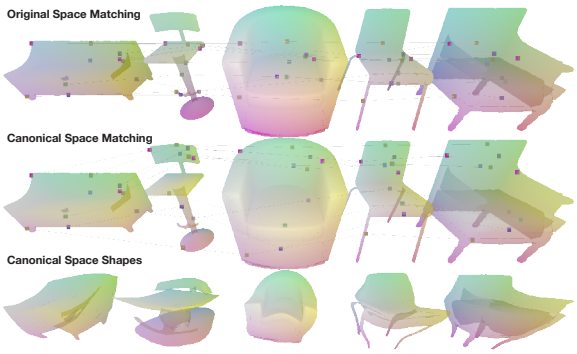

Figure 3: Unsupervised correspondences. Shapes RGB colors correspond to the x, y, z coordinates of each point in the original / canonical space.

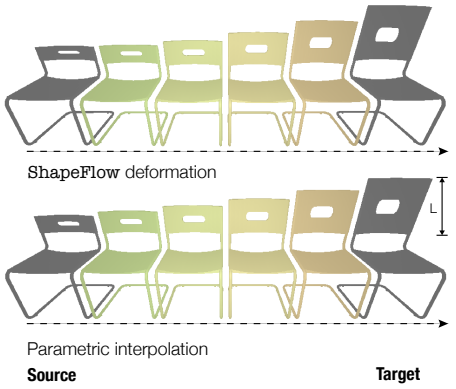

Figure 4: Deformation via parametric controllers [64] compared to the `ShapeFlow` interpolation.

nearest training shapes in the latent space are retrieved, and deformed to z_n. During this step we further fine tune the network parameters θ to perform a better fitting to the observed point cloud.

Task definition. We seek to reconstruct a complete object given a (potentially incomplete) sparse input point cloud. Following [38], we subsample 300 points from mesh surfaces and add a Gaussian noise of 0.05 to the point samples. As a measure of the reconstruction quality, we measure the volumetric Intersection-over-Union (IoU), Chamfer-L_1, as well as normal consistency metrics.

Results. We benchmark against various state-of-the-art shape generation models that outputs voxel grids (3D-R2N2 [5]), upsampled point sets (PSGN [58]), mesh surfaces (DMC [62]) and implicit surfaces (OccFlow [38]); see quantitative results in Table 1. Qualitative comparisons between the generated geometries are illustrated in Figure 2. Note our shape deformations are more constrained (i.e., less expressive) than traditional auto-encoding/decoding, resulting in slightly lower metrics (Table 1). However, `ShapeFlow` is able to produce visually appealing results (Figure 2), as the retrieved shapes are of CAD quality – and *fine geometric details are preserved* by the deformation.

4.1.2 Canonicalization of shapes

An additional property of the deformation space learned through the hub-and-spoke formulation is that it naturally learns an aligned canonical deformation of all shapes. The canonical deformation corresponds to the zero latent code that corresponds to the hub, for shape $i : \{\boldsymbol{z}_i, X_i\}$ it is simply the deformation of X_i from latent code $\boldsymbol{z}_i$ to the hub latent code $\boldsymbol{z}_0 : \Phi_\theta^{i0}(X_i)$. Dense correspondences between shapes can be acquired by searching for the nearest point on the opposing shape in the canonical space. For a point $\boldsymbol{x} \in X_i$, the corresponding point on X_j is found as:

$$\psi^{i \to j}(\boldsymbol{x} \in X_i) = \underset{\boldsymbol{y} \in X_j}{\arg\min} ||\Phi_\theta^{j0}(\boldsymbol{x}) - \Phi_\theta^{j0}(\boldsymbol{y})||_2^2 . \tag{9}$$

Evaluation metric. To quantitatively evaluate the quality of the such surface correspondences learned in an unsupervised manner, we propose the *Semantic Matching Score* (SMS) as a metric for evaluating such correspondences. While semantic correspondences between shapes do not exist, semantic part labels are provided in various shape datasets, including ShapeNet. Denote $L(\boldsymbol{x})$ as an evaluation of the semantic label for the point $\boldsymbol{x}$, $\langle \cdot, \cdot \rangle$ is a label comparison operator that evaluates to

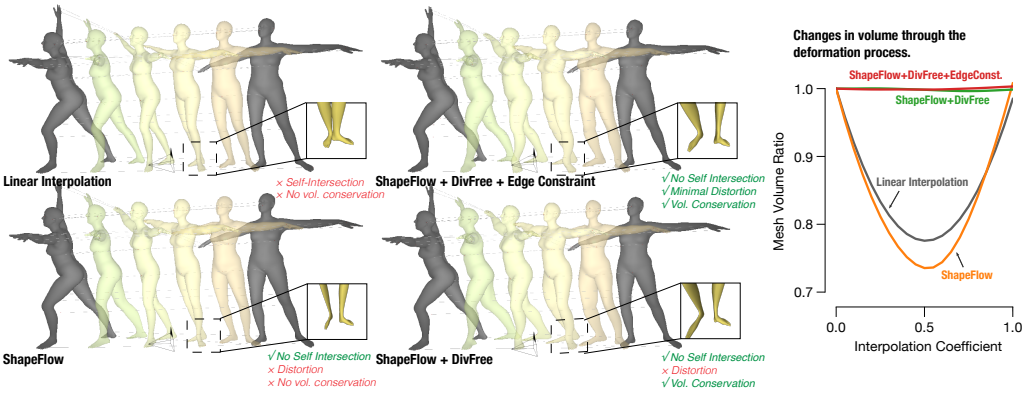

Figure 5: Animation of human figures via `ShapeFlow` deformation. (left) Intermediate poses interpolated between frames. (right) Volume change of the intermediate meshes, showing that our divergence-free flows conserve volume throughout the deformation.

one if the categorical labels are the same and zero otherwise. We define SMS between (X_i, X_j) as:

$$\mathcal{S}(X_i, X_j) = \frac{1}{2}\Big(\frac{1}{|X_i|}\sum_{\boldsymbol{x}\in X_i}\langle L(\boldsymbol{x}), L(\psi^{i\to j}(\boldsymbol{x}))\rangle + \frac{1}{|X_j|}\sum_{\boldsymbol{x}\in X_j}\langle L(\boldsymbol{x}), L(\psi^{j\to i}(\boldsymbol{x}))\rangle\Big) \tag{10}$$

We choose 10,000 random pairs of shapes in the chair category to compute semantic matching scores.

Results. We first visualize the canonicalization and surface correspondences of shapes in the deformation space in Fig. 3. We compare the semantic matching score for our learned dense correspondence function with the naive baseline of nearest neighbor matching in the original (ShapeNet) shape space. The results are presented in the inset table. The shapes align better in the canonical pose, and the matches found by canonical space matching are more semantically correct, especially between originally poorly aligned space due to different aspect ratios (e.g., couch and bar stool). This is reflected in the improved SMS matching score, as reported in the inset table.

Domain	SMS↑
ShapeNet	0.779
`ShapeFlow`	**0.816**

4.2 Human deformation animation

`ShapeFlow` can be used to producing smooth animated deformations between pairs of 3D geometries. These animations are subject to the implicit and explicit constraints for volume and isometry conservation; see Section 3.1. To test the quality of such animated deformations, we choose two relatively distinct SMLP poses [2], and produce continuous deformations for in-between frames. Given that dense correspondences between shapes are given, we change the distance metric $\mathcal{C}$ in Eqn. 2 to be the pairwise L_2 norm between all vertices. We supervise the deformation with 5 intermediate frames produced via linear interpolation. Denoting the geometries at the two end-points as $i = 0$ and $j = 1$, the deformation at intermediate step α is:

$$X_{\alpha\in[0,1]} = \frac{1}{2}\Big(\Phi_\theta^{0\alpha}(X_0) + \Phi_\theta^{1\alpha}(X_1)\Big),\ z_\alpha = (1-\alpha)z_0 + \alpha z_1 \tag{11}$$

Results. We present the results of this deformation in Figure 5. We compare several cases, including direct linear interpolation, deformation using an unconstrained flow, volume constrained flow, as well as volume and edge length constrained flow model. The volume change curve in Figure 5 empirically validates our theoretical result in Section 3.1, that (1) a divergence-free flow conserves the *volume* of a mesh through the deformation process, and (2) prevents self-intersections of the mesh, as in the example in Figure 5. Furthermore, we find that explicit constraints, such as the edge length constraint, reduces surface distortions.

4.3 Comparison with parametric deformations

As a final experiment, we compare the unsupervised deformation acquired using `ShapeFlow` with interpolations of parametric CAD models. We use an exemplar parametric CAD model from [64];

see Figure 4. ShapeFlow produces novel intermediate shapes of CAD level geometric quality that are consistent with those produced by interpolating a parametric model.

5 Conclusions and future work

ShapeFlow is a flow-based model capable to build high-quality shape-spaces by using deformation flows. We analytically show that ShapeFlow prevents self-intersections, and provide ways to regularize volume, isometry, and symmetry. ShapeFlow can be applied to reconstruct new shapes via the deformation of existing templates. A main limitation for the current framework is that it does not incorporate semantic supervision for matching shapes. Future directions include analyzing part structures of geometries by grouping similar vector fields [65], and exploring semantics-aware deformations. Furthermore, ShapeFlow may be used for the inverse problem of inferring a solenoidal flow field given tracer observations [66], an important problem in engineering physics.

Acknowledgements

We thank Or Litany, Tolga Birdal, Yueqi Duan, Kaichun Mo for helpful discussions regarding the project. Andrea Tagliasacchi is funded by NSERC Discovery grant RGPIN-2016-05786, NSERC Collaborative Research and Development grant CRDPJ 537560-18, and NSERC Research Tool Instruments RTI-16-2018. The authors acknowledge the support of a Vannevar Bush Faculty fellowship, a grant form the Samsung GRO program, and gifts from Amazon AWS, Autodesk and Snap.

Broader impact

The work has broad potential impact within the computer vision and graphics community, as it describes a novel methodology that enables a range of new applications, from animation to novel content creation. We have discussed the potential future directions the work could take in Sec. 5.

On the broader societal level, this work remains largely academic in nature, and does not pose foreseeable risks regarding defense, security, and other sensitive fields.

References

[1] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 187–194, 1999.

[2] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, October 2015.

[3] Silvia Zuffi, Angjoo Kanazawa, David W Jacobs, and Michael J Black. 3d menagerie: Modeling the 3d shape and pose of animals. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6365–6373, 2017.

[4] 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.

[5] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pages 628–644. Springer, 2016.

[6] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2088–2096, 2017.

[7] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 216–224, 2018.

[8] Charlie Nash, Yaroslav Ganin, SM Eslami, and Peter W Battaglia. Polygen: An autoregressive generative model of 3d meshes. *arXiv preprint arXiv:2002.10880*, 2020.

[9] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnet: Learnable convex decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[10] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. Bsp-net: Generating compact meshes via binary space partitioning. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[11] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019.

[12] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Deep structured implicit functions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[13] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–67, 2018.

[14] Or Litany, Alex Bronstein, Michael Bronstein, and Ameesh Makadia. Deformable shape completion with graph convolutional autoencoders. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1886–1895, 2018.

[15] Scott Schaefer, Travis McPhail, and Joe Warren. Image deformation using moving least squares. In *ACM SIGGRAPH 2006 Papers*, pages 533–540, 2006.

[16] Alec Jacobson, Ilya Baran, Jovan Popovic, and Olga Sorkine. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph.*, 30(4):78, 2011.

[17] Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. Harmonic coordinates for character articulation. *ACM Transactions on Graphics (TOG)*, 26(3):71–es, 2007.

[18] Yaron Lipman, David Levin, and Daniel Cohen-Or. Green coordinates. *ACM Transactions on Graphics (TOG)*, 27(3):1–10, 2008.

[19] Ofir Weber, Mirela Ben-Chen, Craig Gotsman, et al. Complex barycentric coordinates with applications to planar shape deformation. In *Computer Graphics Forum*, volume 28, page 587, 2009.

[20] James Booth, Anastasios Roussos, Stefanos Zafeiriou, Allan Ponniah, and David Dunaway. A 3d morphable model learnt from 10,000 faces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5543–5552, 2016.

[21] Patrik Huber, Guosheng Hu, Rafael Tena, Pouria Mortazavian, P Koppen, William J Christmas, Matthias Ratsch, and Josef Kittler. A multiresolution 3d morphable face model and fitting framework. In *Proceedings of the 11th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2016.

[22] Xiangyu Zhu, Junjie Yan, Dong Yi, Zhen Lei, and Stan Z Li. Discriminative 3d morphable model fitting. In *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, volume 1, pages 1–8. IEEE, 2015.

[23] Xiangyu Zhu, Zhen Lei, Xiaoming Liu, Hailin Shi, and Stan Z Li. Face alignment across large poses: A 3d solution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 146–155, 2016.

[24] Xiangyu Zhu, Zhen Lei, Junjie Yan, Dong Yi, and Stan Z Li. High-fidelity pose and expression normalization for face recognition in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 787–796, 2015.

[25] Hang Dai, Nick Pears, William AP Smith, and Christian Duncan. A 3d morphable model of craniofacial shape and texture variation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3085–3093, 2017.

[26] Stylianos Ploumpis, Evangelos Ververas, Eimear O'Sullivan, Stylianos Moschoglou, Haoyang Wang, Nick Pears, William Smith, Baris Gecer, and Stefanos P Zafeiriou. Towards a complete 3d morphable model of the human head. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[27] Nils Hasler, Carsten Stoll, Martin Sunkel, Bodo Rosenhahn, and H-P Seidel. A statistical model of human pose and body shape. In *Computer graphics forum*, volume 28, pages 337–346. Wiley Online Library, 2009.

[28] Brett Allen, Brian Curless, and Zoran Popović. The space of human body shapes: reconstruction and parameterization from range scans. *ACM transactions on graphics (TOG)*, 22(3):587–594, 2003.

[29] Silvia Zuffi, Angjoo Kanazawa, and Michael J Black. Lions and tigers and bears: Capturing non-rigid, 3d, articulated shape from images. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 3955–3963, 2018.

[30] Kyle Genova, Forrester Cole, Aaron Maschinot, Aaron Sarna, Daniel Vlasic, and William T Freeman. Unsupervised training for 3d morphable model regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8377–8386, 2018.

[31] Soubhik Sanyal, Timo Bolkart, Haiwen Feng, and Michael J Black. Learning to regress 3d face shape and expression from an image without 3d supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7763–7772, 2019.

[32] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J Black. Generating 3d faces using convolutional mesh autoencoders. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 704–720, 2018.

[33] Nikos Kolotouros, Georgios Pavlakos, and Kostas Daniilidis. Convolutional mesh regression for single-image human shape reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4501–4510, 2019.

[34] Theo Deprelle, Thibault Groueix, Matthew Fisher, Vladimir Kim, Bryan Russell, and Mathieu Aubry. Learning elementary structures for 3d shape generation and matching. In *Advances in Neural Information Processing Systems*, pages 7433–7443, 2019.

[35] Vignesh Ganapathi-Subramanian, Olga Diamanti, Soeren Pirk, Chengcheng Tang, Matthias Niessner, and Leonidas Guibas. Parsing geometry using structure-aware shape templates. In *2018 International Conference on 3D Vision (3DV)*, pages 672–681. IEEE, 2018.

[36] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[37] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in neural information processing systems*, pages 82–90, 2016.

[38] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[39] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[40] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, pages 1119–1130, 2019.

[41] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *arXiv preprint arXiv:2003.08934*, 2020.

[42] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, and Thomas Funkhouser. Local implicit grid representations for 3d scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[43] Rohan Chabra, Jan Eric Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. *arXiv preprint arXiv:2003.10983*, 2020.

[44] Chiyu Max Jiang, Soheil Esmaeilzadeh, Kamyar Azizzadenesheli, Karthik Kashinath, Mustafa Mustafa, Hamdi A Tchelepi, Philip Marcus, Anima Anandkumar, et al. Meshfreeflownet: A physics-constrained deep continuous space-time super-resolution framework. *arXiv preprint arXiv:2005.01463*, 2020.

[45] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, pages 109–116, 2007.

[46] Isaac Chao, Ulrich Pinkall, Patrick Sanan, and Peter Schröder. A simple geometric model for elastic deformations. *ACM transactions on graphics (TOG)*, 29(4):1–6, 2010.

[47] Mikaela Angelina Uy, Jingwei Huang, Minhyuk Sung, Tolga Birdal, and Leonidas Guibas. Deformation-aware 3d model embedding and retrieval. *arXiv preprint arXiv:2004.01228*, 2020.

[48] Wolfram Von Funck, Holger Theisel, and Hans-Peter Seidel. Vector field based shape deformations. *ACM Transactions on Graphics (TOG)*, 25(3):1118–1125, 2006.

[49] Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. 3dn: 3d deformation network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1038–1046, 2019.

[50] Andrey Kurenkov, Jingwei Ji, Animesh Garg, Viraj Mehta, JunYoung Gwak, Christopher Choy, and Silvio Savarese. Deformnet: Free-form deformation network for 3d shape reconstruction from a single image. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 858–866. IEEE, 2018.

[51] Wang Yifan, Noam Aigerman, Vladimir Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. Neural cages for detail-preserving 3d deformations. *arXiv preprint arXiv:1912.06395*, 2019.

[52] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[53] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017.

[54] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pages 6571–6583, 2018.

[55] Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

[56] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4541–4550, 2019.

[57] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5379–5389, 2019.

[58] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017.

[59] Matheus Gadelha, Rui Wang, and Subhransu Maji. Deep manifold prior. *arXiv preprint arXiv:2004.04242*, 2020.

[60] Jan Bednarik, Shaifali Parashar, Erhan Gundogdu, Mathieu Salzmann, and Pascal Fua. Shape reconstruction by learning differentiable surface representations. *arXiv preprint arXiv:1911.11227*, 2019.

[61] Piotr Bojanowski, Armand Joulin, David Lopez-Paz, and Arthur Szlam. Optimizing the latent space of generative networks. *arXiv preprint arXiv:1707.05776*, 2017.

[62] Yiyi Liao, Simon Donne, and Andreas Geiger. Deep marching cubes: Learning explicit surface representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2916–2925, 2018.

[63] Chris Rorden. Fast Quadratic Mesh Simplification, 2020. URL `https://github.com/sp4cerat/Fast-Quadric-Mesh-Simplification`.

[64] Adriana Schulz, Ariel Shamir, Ilya Baran, David I. W. Levin, Pitchaya Sitthi-Amorn, and Wojciech Matusik. Retrieval on parametric shape collections. *ACM Transactions on Graphics*, 36(1), January 2017.

[65] Liefei Xu, H Quynh Dinh, Philippos Mordohai, and Thomas Ramsay. Detecting patterns in vector fields. In *49th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, page 1136, 2011.

[66] Christian E Willert and Morteza Gharib. Digital particle image velocimetry. *Experiments in fluids*, 10(4):181–193, 1991.