ARAPReg: An As-Rigid-As Possible Regularization Loss for Learning Deformable Shape Generators

Qixing Huang*
UT Austin
huangqx@cs.utexas.edu

Xiangru Huang*
UT Austin & MIT
xiangruhuang816@gmail.com

Bo Sun*
UT Austin
bosun@cs.utexas.edu

Zaiwei Zhang
UT Austin
zaiweizhang@utexas.edu

Junfeng Jiang
Hohai University
jiangjf.hhu@gmail.com

Chandrajit Bajaj
UT Austin
bajaj@cs.utexas.edu

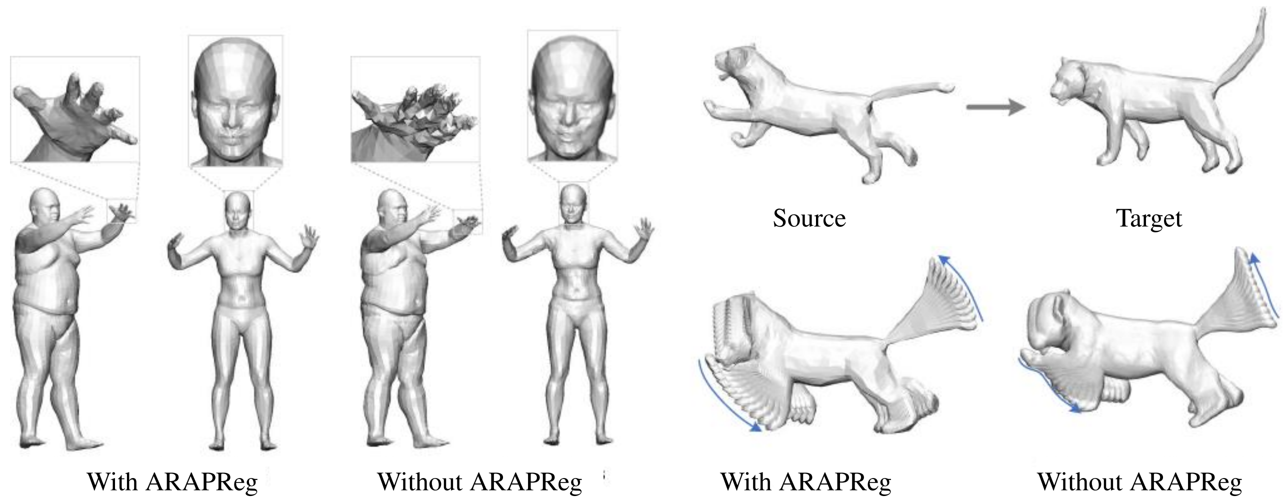

Figure 1: Our approach learns a shape generator from a collection of deformable shapes. The shape generator is trained with a novel as-rigid-as-possible regularization (ARAPReg) loss that promotes the preservation of multi-scale shape features.

Abstract

This paper introduces an unsupervised loss for training parametric deformation shape generators. The key idea is to enforce the preservation of local rigidity among the generated shapes. Our approach builds on an approximation of the as-rigid-as possible (or ARAP) deformation energy. We show how to develop the unsupervised loss via a spectral decomposition of the Hessian of the ARAP energy. Our loss nicely decouples pose and shape variations through a robust norm. The loss admits simple closed-form expressions. It is easy to train and can be plugged into any standard generation models, e.g., variational auto-encoder (VAE) and auto-decoder (AD). Experimental results show that our approach outperforms existing shape generation approaches considerably on public benchmark datasets of various shape categories such as human, animal and bone. Our code and data are available at https://github.com/GitBoSun/ARAPReg.

1. Introduction

This paper considers learning a parametric mesh generator from a deformable shape collection with shapes that exhibit the same topology but undergo large geometric variations (see examples below of a deforming human, animal, and bone). This problem arises in numerous visual computing and relevant fields such as recovery of neural morphogenesis, data-driven shape reconstruction, and image-based reconstruction, to name just a few (c.f. [54]).

Deformable shapes differ from many other visual objects (e.g., images and videos) because there are natural constraints underlying the shape space. One such example is the local rigidity constraint; namely, corresponding surface patches among neighboring shapes in the shape space undergo approximately rigid transformations. This constraint manifests the preservation of geometric features (e.g., facial features of humans and toes of animals) among local neighborhoods of the underlying shape space. An interesting problem thus is the use of this constraint to train shape generators from a collection of training shapes, where the local rigidity constraint accurately and efficiently propagates features of the training shapes to new synthetic shapes produced by the generator.

In this paper, we study how to model the local rigidity constraint as an unsupervised loss functional for generative modeling. The proposed loss can be combined with standard mesh generators such as variational auto-encoders (VAEs) [44, 28, 39, 7] and auto-decoders (ADs) [59, 61]. A key property of our loss functional is that it is consistent with other training losses. This property offers multiple advantages. For example, the learned generator is insensitive to the tradeoff parameters among the loss terms. As another example, the training procedure converges faster than the setting where loss terms may compete against each other.

Our approach, called ARAPReg, builds on the established as-rigid-as-possible (or ARAP) deformation model [43, 49, 55] that measures the non-rigid deformation between two shapes. Its key ingredients include use oof the Hessian of the ARAP deformation model to derive an explicit regularizer for the Jacobian of the shape generator and a robust norm on the Hessian to model pose and shape variations of deformable shapes. The outcome is a simple closed-form formulation for training mesh generators. ARAPReg differs from prior works that enforce ARAP losses between synthetic shapes and a base shape [17, 27, 63], that may introduce competing losses when the underlying shape space has large deformations.

We have evaluated ARAPReg across a variety of public benchmark datasets such as DFAUST [5], SMAL [66], and an in-house benchmark dataset of Bone. The evaluations include both generator settings of VAE and AD. Experimental results show that ARAPReg leads to considerable performance gains across state-of-the-art deformable shape generators both qualitatively and quantitatively. As shown in Figure 1 for example, the interpolated shapes using ARAPReg greatly preserve the local geometric details of the generated shapes and avoids unrealistic shape poses.

2. Related Works

This section organizes the relevant works into three groups, namely, 3D generative models, regularization for generative modeling, and shape space modeling.

3D generative models. Learning 3D generative models relies on developing suitable 3D representations to encode 3D models into vectorized forms. Examples include volumetric grid [53, 45, 12, 34, 32, 19], implicit surfaces [35, 9], point clouds [1, 57, 56, 25], meshes [22, 28, 14], parametric surfaces [16, 31], spherical representations [10, 13, 8], geometric arrangements [47, 62], and multi-views [30].

This paper is mostly relevant to generative models under the mesh representation, which falls into four categories. The first category of approaches [44, 48, 28, 46, 38] is based on defining variational auto-encoders on meshes. A typical strategy is to treat triangular meshes as graphs and define convolution and deconvolution operations to synthesize triangular meshes (c.f. [48, 28, 46]). [44] introduced a geometric encoding scheme that operates in the gradient domain. The second category of approaches builds upon recurrent procedures for geometric synthesis. This methodology has been extensively applied for primitive-based assembly [26, 40, 41, 65]. [18] extended this approach to meshes, in which edge contraction operations are applied recursively. The third category of approaches [60, 51] deforms a base mesh to generate new meshes, where the deformation is learned from data. The last category utilizes surface parameterization [42, 31, 16, 4].

While these approaches focused on adopting generative modeling methodologies under the mesh setting, ARAPReg studies the novel problem of explicitly enforcing an ARAP loss among synthetic shapes with similar latent codes.

Regularization for generative modeling. Regularization losses have been explored in prior works for 3D generative modeling. In [36], Peebles et al. studied a Hessian regularization term for learning generative image models. A spectral regularization loss is introduced in [2] for 3D generative modeling. Several works [52, 50, 21, 3] studied geometric regularizations for image-based reconstruction. In contrast, ARAPReg focuses on regularization terms that are consistent with other terms. Several other works [17, 27, 63] employed ARAP losses between any synthetic shapes with a base shape. The novelty of ARAPReg is that it is consistent with other loss terms even when the underlying shape space presents large deformations. The reason is that the local rigidity constraint is only enforced among neighboring shapes in the underlying shape space. Our initial experiments show that enforcing ARAP losses between synthetic shapes and a base shape leads to worse results than dropping the ARAP losses.

Shape space modeling. Finally, ARAPReg is relevant to early works on modeling tangent spaces of shape manifolds [23, 20, 58]. However, unlike the applications in shape interpolation [23], shape segmentation [20], and mesh-based geometric design [37, 58], ARAPReg focuses on devising an unsupervised loss for network training.

3. Overview

Following [39, 7, 64], we are interested in learning a mesh generator that takes a latent code as input and outputs the vertex positions of a triangular mesh with given mesh connectivity (See Figure 2). Formally speaking, we denote this mesh generator as

$$\boldsymbol{g}^{\theta} : \mathcal{Z} := \mathcal{R}^{k} \rightarrow \mathcal{R}^{3n}.$$

Here $\mathcal{Z}$ represents the latent space, and $\mathcal{R}^{3n}$ encodes the vector that concatenates the vertex positions, i.e., n is the number of vertices. We organize the remainder of this paper as follows.

In Section 4, we introduce the key contribution of this paper, ARAPReg, an unsupervised loss for training $\boldsymbol{g}^{\theta}$. The loss only requires a prior distribution of the latent space $\mathcal{Z}$. In this paper, we assume the prior distribution is the Normal distribution $\mathcal{N}_k$ of dimension k. The key idea of ARAPReg is to ensure that the local rigidity constraint is pre-

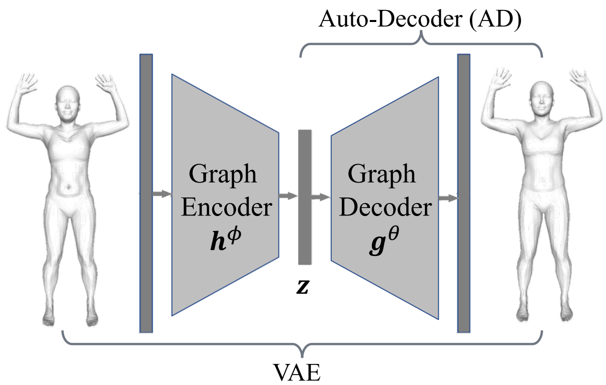

Figure 2: We consider multiple standard shape generators, including Variational Auto-Encoder (VAE) and Auto-Decoder (AD). The graph encoder $\boldsymbol{h}^{\phi}$ maps the input mesh $\boldsymbol{g}$ to a latent parameter $\boldsymbol{h}^{\phi}(\boldsymbol{z})$. The graph decoder maps a latent parameter $\boldsymbol{z}$ to the out mesh $\boldsymbol{g}^{\theta}(\boldsymbol{z})$.

served among neighboring generated shapes in the underlying shape space. As illustrated in the left part of Figure 1, the goal of this loss is to significantly improve the generalization behavior of the mesh generator, e.g., preserving multi-scale geometric details.

In Section 5, we discuss how to plug this unsupervised loss into standard shape generation models based on VAE and AD.

4. Formulation of the ARAPReg Loss

Formulating the preservation of local rigidity is quite challenging because the resulting loss term has to be simple enough to facilitate network training. One straightforward approach is to enforce the local rigidity constraint between a generated shape $\boldsymbol{g}^{\theta}(\boldsymbol{z})$ and its perturbation $\boldsymbol{g}^{\theta}(\boldsymbol{z}+d\boldsymbol{z})$. Here $d\boldsymbol{z}$ characterizes an infinitesimal displacement in the parameter space. However, this approach requires sampling a lot of shape pairs. Besides, typical formulations of shape deformations between $\boldsymbol{g}^{\theta}(\boldsymbol{z})$ and $\boldsymbol{g}^{\theta}(\boldsymbol{z}+d\boldsymbol{z})$ require solving optimization problems that are computationally expensive (c.f. [6]).

ARAPReg stitches several novel ideas to derive a simple unsupervised loss that does not adversely compete with typical losses used in generative modeling (See Section 5.1).

4.1. Step I: Decoupling Smoothness and Jacobian regularization

First, ARAPReg decouples the enforcement of local rigidity into two terms. The first term enforces the smoothness of the generator. This smoothness penalty enables the second term, which formulates the preservation of local rigidity as potentials on the Jacobian of the generator, i.e.,

$$\frac{\partial \boldsymbol{g}^{\theta}}{\partial \boldsymbol{z}}(\boldsymbol{z}) \in \mathcal{R}^{(3n)\times k}.$$

Specifically, we define the unsupervised loss as

$$\mathcal{L}_{reg}(\theta) := \underset{\boldsymbol{z}\sim\mathcal{N}_k}{E}\Big(\underset{\delta\boldsymbol{z}\sim s\mathcal{N}_k}{E}\|\boldsymbol{g}^{\theta}(\boldsymbol{z}+\delta\boldsymbol{z}) - 2\boldsymbol{g}^{\theta}(\boldsymbol{z}) + \boldsymbol{g}^{\theta}(\boldsymbol{z}-\delta\boldsymbol{z})\|^2 + \lambda_R \cdot r_R(\boldsymbol{g}^{\theta}(\boldsymbol{z}), \frac{\partial \boldsymbol{g}^{\theta}}{\partial \boldsymbol{z}}(\boldsymbol{z}))\Big), \quad (1)$$

where the first term promotes the smoothness of the generator $\boldsymbol{g}^{\theta}$; s is a hyper-parameter of ARAPReg. Note that unlike enforcing

$$\boldsymbol{g}^{\theta}(\boldsymbol{z}+\delta\boldsymbol{z}) \approx \boldsymbol{g}^{\theta}(\boldsymbol{z}) + \frac{\partial \boldsymbol{g}^{\theta}}{\partial \boldsymbol{z}}(\boldsymbol{z})\cdot\delta\boldsymbol{z}, \quad (2)$$

the formulation in (1) does not involve the first-order derivatives of $\boldsymbol{g}$. It follows that network training is more efficient as it only requires computing the first-order derivatives of $\boldsymbol{g}$. On the other hand, it penalizes the second-order derivatives of $\boldsymbol{g}^{\theta}$. It therefore implicitly enforces (2). The second term $r_R(\boldsymbol{g}^{\theta}(\boldsymbol{z}), \frac{\partial \boldsymbol{g}^{\theta}}{\partial \boldsymbol{z}}(\boldsymbol{z}))$ in (1), which will be defined shortly, formulates the regularization loss concerning the generated mesh $\boldsymbol{g}^{\theta}(\boldsymbol{z})$ and infinitesimal perturbations specified by the Jacobian $\frac{\partial \boldsymbol{g}^{\theta}}{\partial \boldsymbol{z}}(\boldsymbol{z})$ (See Figure 3). λ_R is another hyper-parameter of ARAPReg.

In other words, instead of enforcing the local rigidity between shape pairs, ARAPReg enforces the preservation of the local rigidity in the tangent space specified by the Jacobian. The tangent space is a first-order approximation of the shape space. The smoothness potential ensures that this first-order approximation is accurate, i.e., the rigidity constraint propagates to the shape space's local neighborhood. As we will discuss later, another appealing property of this formulation is that the Jacobian enables us to easily model pose and shape variations (where pose variations are more rigid than shape variations). This goal is hard to achieve using generic pairwise regularizations.

Although the smoothness constraint involves shape pairs, our experiments suggest that there is no need to sample a large number of shape pairs. One interpretation is that deep neural network training has implicit regularizations (c.f. [33]), which promotes smoothness.

4.2. Step II: Jacobian Regularization

We proceed to introduce the local rigidity term r_R that regularizes the Jacobian of the generator. To make the notations uncluttered, we focus on formulating $r_R(\boldsymbol{g}, J)$. Here $\boldsymbol{g} \in \mathcal{R}^{3n}$ denotes a vertex position vector, and $J \in \mathcal{R}^{3n\times k}$ is a Jacobian matrix that specifies infinitesimal perturbations to $\boldsymbol{g}$.

Our formulation is inspired by the as-rigid-as possible (or ARAP) potential function [43, 49, 55]. This standard model measures the deformation between a pair of shapes. Consider a mesh with vertex position $\boldsymbol{g} \in \mathcal{R}^{3n}$ and the same mesh with perturbed vertex position $\boldsymbol{g}+\boldsymbol{x} \in \mathcal{R}^{3n}$. Denote $O_i \in SO(3)$ as the latent rotation associated with the i-th

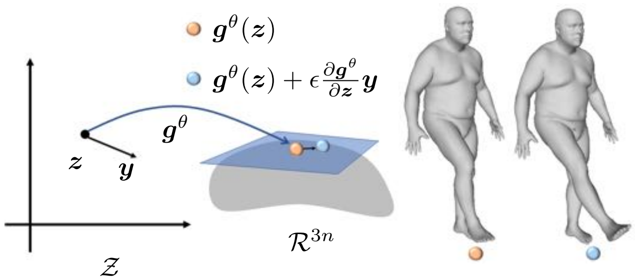

Figure 3: Illustration of configuration for Jacobian regularization. We study infinitesimal deformations incurred by the tangent space at each generated shape $\boldsymbol{g}^\theta(\boldsymbol{z})$.

vertex. The ARAP deformation between them is

$$f_R(\boldsymbol{g},\boldsymbol{x}) := \min_{O_i \in SO(3)} \sum_{(i,j)\in\mathcal{E}} \|\boldsymbol{r}_{ij}(O_i,\boldsymbol{g},\boldsymbol{x})\|^2 \tag{3}$$
$$\boldsymbol{r}_{ij}(O_i,\boldsymbol{g},\boldsymbol{x}) := (O_i - I_3)(\boldsymbol{g}_i - \boldsymbol{g}_j) - (\boldsymbol{x}_i - \boldsymbol{x}_j)$$

where $\mathcal{E}$ denotes the edge set of the mesh generator $\boldsymbol{g}^\theta$. Note that we assume $(i,j)\in\mathcal{E}$ if and only if $(j,i)\in\mathcal{E}$.

To introduce a formulation that only depends on the Jacobian of the generator, we consider the Taylor expansion of the as-rigid-as possible potential energy.

Proposition 1 *([20]) The zero and first-order derivatives of f_R satisfy*

$$f_R(\boldsymbol{g},\boldsymbol{0}) = 0, \qquad \frac{\partial f_R}{\partial \boldsymbol{x}}(\boldsymbol{g},\boldsymbol{0}) = \boldsymbol{0}.$$

Moreover, the Hessian matrix is given by

$$\frac{\partial^2 f_R}{\partial^2 \boldsymbol{x}}(\boldsymbol{g},\boldsymbol{0}) = H_R(\boldsymbol{g}),$$
$$H_R(\boldsymbol{g}) = L\otimes I_3 - A(\boldsymbol{g})^T D(\boldsymbol{g})^{-1} A(\boldsymbol{g}), \tag{4}$$

where $L\in\mathcal{R}^{n\times n}$ is the graph Laplacian associated to $\mathcal{E}$; $A(\boldsymbol{g})$ is a sparse $n\times n$ block matrix; $D(\boldsymbol{g})$ is a diagonal block matrix. The blocks of $A(\boldsymbol{g})$ and $D(\boldsymbol{g})$ are given by

$$A_{ij}(\boldsymbol{g}) = \begin{cases} \sum\limits_{k\in\mathcal{N}(i)} (\boldsymbol{v}_{ik}\times) & i=j \\ -\boldsymbol{v}_{ij}\times & (i,j)\in\mathcal{E} \\ 0 & \text{otherwise} \end{cases}$$
$$D_{ij}(\boldsymbol{g}) = \begin{cases} \sum\limits_{k\in\mathcal{N}(i)} (\|\boldsymbol{v}_{ik}\|^2 I_3 - \boldsymbol{v}_{ik}\boldsymbol{v}_{ik}^T) & i=j \\ 0 & \text{otherwise} \end{cases}$$

where $\boldsymbol{v}_{ij} = \boldsymbol{g}_i - \boldsymbol{g}_j$, and $\mathcal{N}(i)$ collects indices of adjacent vertices of i. Note that H_R is a highly sparse matrix.

Proposition 1 indicates that for each vector $\boldsymbol{y}\in\mathcal{R}^d$ in the parameter space, the ARAP potential between the mesh defined by $\boldsymbol{g}$ and its infinitesimal displacement encoded by $\epsilon J\boldsymbol{y}$ (for a small ϵ) can be approximated as

$$f_R(\boldsymbol{g},\epsilon J\boldsymbol{y}) \approx \frac{1}{2}\epsilon^2 \boldsymbol{y}^T \overline{H}_R(\boldsymbol{g},J)\boldsymbol{y}, \tag{5}$$

where $\overline{H}_R(\boldsymbol{g},J) := J^T H_R(\boldsymbol{g})J$.

(5) provides the rigidity potential along a direction $\boldsymbol{y}$ in the latent space. Our formulation of r_R seeks to integrate (5) over all possible directions $\boldsymbol{y}$. To motivate the final formulation of ARAPReg, let us first define an initial potential energy by integrating $\boldsymbol{y}^T\overline{H}_R(\boldsymbol{g},J)\boldsymbol{y}$ over the unit-sphere $\mathcal{S}^k$ in $\mathcal{R}^k$ that specifies all possible $\boldsymbol{y}$:

$$r_R^{L^2}(\boldsymbol{g},J) := \frac{k}{\mathrm{Vol}(\mathcal{S}^k)} \int_{\boldsymbol{y}\in\mathcal{S}^k} \boldsymbol{y}^T\overline{H}_R(\boldsymbol{g},J)\boldsymbol{y}d\boldsymbol{y}. \tag{6}$$

Proposition 2

$$r_R^{L^2}(\boldsymbol{g},J) = \mathrm{Tr}(\overline{H}_R(\boldsymbol{g},J)) = \sum_{i=1}^k \lambda_i(\overline{H}_R(\boldsymbol{g},J)) \tag{7}$$

where $\lambda_i(\overline{H}_R(\boldsymbol{g},J))$ is the i-th eigenvalue of $\overline{H}_R(\boldsymbol{g},J)$.

4.3. Step III: Pose and Shape Variation Modeling

We present a simple formulation that decouples enforcing pose and shape variations. Specifically, the eigenvalues $\lambda_i(\overline{H}_R(\boldsymbol{g},J)) = \boldsymbol{u}_i^T\overline{H}_R(\boldsymbol{g},J)\boldsymbol{u}_i$, where $\boldsymbol{u}_i$ is the corresponding eigenvector of $\lambda_i(\overline{H}_R(\boldsymbol{g},J))$, reveal the deformations in different directions of the tangent space. From the definition of the as-rigid-as possible deformation energy, each vertex's one-ring neighborhood is mostly rigid under pose variations. In contrast, the one-ring neighborhoods may change drastically under shape variations. This means eigenvectors with small eigenvalues correspond to pose variations, while eigenvectors with large eigenvalues correspond to shape variations (See Figure 4).

The limitation of the L2 formulation described in (2) is that all directions are penalized equally. ARAPReg employs a robust norm to model the local rigidity loss to address this issue

$$r_R(\boldsymbol{g},J) = \sum_{i=1}^k \lambda_i^\alpha(\overline{H}_R(\boldsymbol{g},J)), \tag{8}$$

where we set $\alpha = \frac{1}{2}$ in this paper. Similar to the effects of using robust norms for outlier removal, (8) imposes small weights on the subspace spanned by eigenvectors of large eigenvalues, which correspond to shape variations. In other words, minimizing (8) minimizes the small eigenvalues of $\overline{H}_R(\boldsymbol{g},J)$ automatically, which correspond to pose variations. Note that several prior works [63, 11, 2] aimed to decouple pose and shape in the latent space. In contrast, our goal is to model the regularization term by taking pose and shape variations into account.

4.4. Step IV: Final Loss Term

Substituting (6) into (1), we have

$$\mathcal{L}_{reg}(\theta) := \mathop{E}_{\boldsymbol{z}\sim\mathcal{N}_k}\Big(\mathop{E}_{\delta\boldsymbol{z}\sim s\mathcal{N}_k} \|\boldsymbol{g}^\theta(\boldsymbol{z}+\delta\boldsymbol{z}) - 2\boldsymbol{g}^\theta(\boldsymbol{z}) + \boldsymbol{g}^\theta(\boldsymbol{z}-\delta\boldsymbol{z})\|^2 + \lambda_R \sum_{i=1}^k \lambda_i^\alpha\big(\overline{H}_R\big(\boldsymbol{g}^\theta(\boldsymbol{z}), \frac{\partial \boldsymbol{g}^\theta}{\partial \boldsymbol{z}}(\boldsymbol{z})\big)\big)\Big) \tag{9}$$

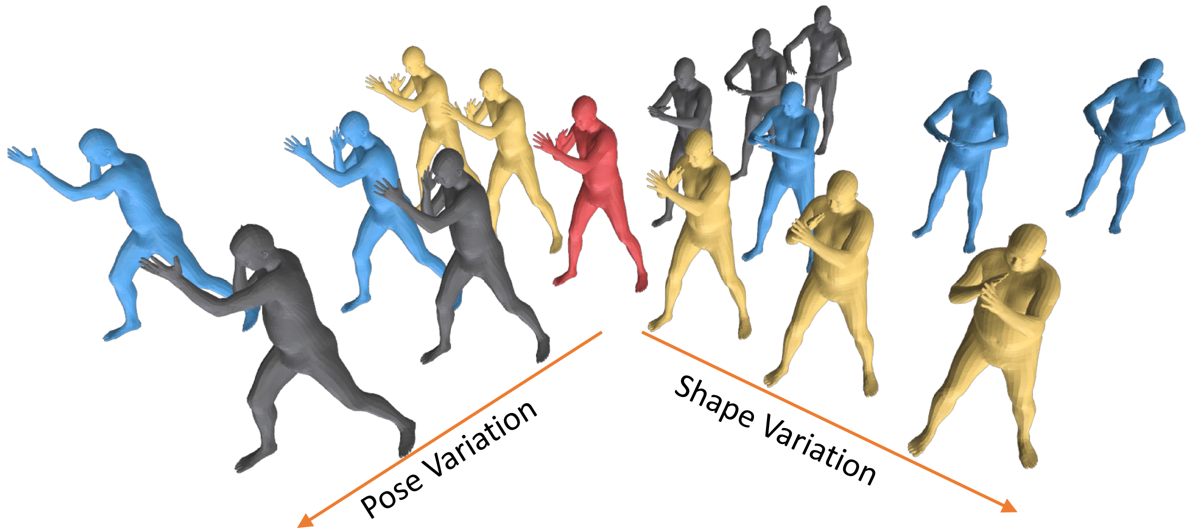

Figure 4: This figure illustrates the local shape space spanned by the eigenvectors of the Hessian of $\overline{H}_R(\boldsymbol{g}, J)$. The red shape in the center is the reference shape. When moving the latent parameter along the first eigenvector, the shape deformation is locally rigid, exhibiting pose variations (see grey shapes). When moving along the largest eigenvector, the shape deformation possesses local stretching, corresponding to shape variations (see yellow shapes). Finally, when moving along a linear combination of both eigenvectors, the shape exhibits both pose and shape variations (see blue shapes).

In this paper, we set $s = 0.05$ and $\lambda_R = 1$ for all of our experiments.

The major challenge of using (9) for training is to compute the gradient of the Jacobian regularization term. Similar to the formulation of generator smoothness, we introduce a gradient computation approach that only requires computing the derivatives of $\boldsymbol{g}^\theta$. Please refer to the supp. material for details.

5. Application in Learning Mesh Generators

This section introduces the applications of the unsupervised loss for learning mesh generators. We first introduce the network architecture used in this paper for experimental evaluation. We then introduce how to insert the unsupervised loss $\mathcal{L}_{reg}$ described above into two formulations of training mesh generators, i.e., variational auto-encoders [28, 39, 7] and auto-decoders [59, 61].

5.1. Network Architecture

We focus on describing the decoder network $\boldsymbol{g}^\theta$. When training variational auto-encoders, we utilize another encoder network $\boldsymbol{h}^\phi : \mathcal{R}^{3n} \rightarrow \mathcal{Z}$, the mirror of $\boldsymbol{g}^\theta$ but has different network weights. In other words, $\boldsymbol{h}^\phi$ has the identical network layers as $\boldsymbol{g}^\theta$, but the connections are reversed.

In this paper, we model $\boldsymbol{g}^\theta$ using six layers. The second to the sixth layers are the same as the network architecture of [28]. Motivated from [64], we let the first layer concatenate the latent features associated with each vertex of the coarse mesh as input. Between the input and the first activation is a fully connected layer. Please refer to the supp. material for details.

5.2. Variational Auto-Encoder

Given a collection of training meshes $\mathcal{T} = \{\boldsymbol{g}_i | 1 \leq i \leq N\}$, we solve the following optimization problem to train the auto-encoder that combines $\boldsymbol{g}^\theta$ and $\boldsymbol{h}^\phi$:

$$\min_{\theta,\phi} \frac{1}{N} \sum_{i=1}^{N} \|g^\theta(h^\phi(\boldsymbol{g}_i)) - \boldsymbol{g}_i\| + \lambda_{KL} KL(\{h^\phi(\boldsymbol{g}_i)\} | \mathcal{N}_k) + \lambda_{reg}\mathcal{L}_{reg}(\theta) \qquad (10)$$

where the first two terms of (10) form the standard VAE loss. In this paper, we set $\lambda_{KL} = 1$ and $\lambda_{reg} = 10$. For network training, we employ ADAM [24].

5.3. Auto-Decoder

The auto-decoder formulation [59, 61] replaces the encoder with latent variables $\boldsymbol{z}_i$ associated with the training meshes:

$$\min_{\theta,\{\boldsymbol{z}_i\}} \frac{1}{N} \sum_{i=1}^{N} \|\boldsymbol{g}^\theta(\boldsymbol{z}_i) - \boldsymbol{g}_i\| + \lambda_{KL} KL(\{\boldsymbol{z}_i\} | \mathcal{N}_k) + \lambda_{reg}\mathcal{L}_{reg}(\theta) \qquad (11)$$

where we use the same hyper-parameters as Section 5.2.

We apply alternating minimization to solve (11). The latent parameters $\boldsymbol{z}_i$ are initialized as an empirical distribution of $\mathcal{N}_k$. When $\boldsymbol{z}_i$ are fixed, (11) reduces to

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \|\boldsymbol{g}^\theta(\boldsymbol{z}_i) - \boldsymbol{g}_i\| + \lambda_{reg}\mathcal{L}_{reg}(\theta) \qquad (12)$$

We again employ ADAM [24] to solve (12). Our implementation applies one epoch of optimizing θ for each alternating optimization iteration.

When the network parameters θ are fixed, (11) reduces to

$$\min_{\{\boldsymbol{z}_i\}} \frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{g}^{\theta}(\boldsymbol{z}_i) - \boldsymbol{g}_i\| + \lambda_{KL} KL(\{\boldsymbol{z}_i\}|\mathcal{N}_k) \quad (13)$$

We again employ ADAM [24] to optimize $\boldsymbol{z}_i$. Similarly, our implementation applies one epoch of optimizing $\boldsymbol{z}_i$ for each alternating iteration. The total number of alternating iterations is set as 30 in this paper.

6. Experimental Evaluation

This section presents an experimental evaluation of ARAPReg. In Section 6.1, we present the experimental setup. We then analyze the experimental results in Section 6.2. Finally, Section 6.3 and Section 6.4 describe an ablation study of the ASARReg loss and an evaluation of the shape interpolation application. Due to space issues, we defer more results and comparisons to the supp. material.

6.1. Experimental Setup

Datasets. The experimental evaluation considers three datasets: **DFAUST** [5], **SMAL** [66], and **Bone**. The DFAUST dataset consists of 37,197 human shapes for training and 4,264 shapes for testing. All the shapes are generated using the SMPL model [29]. For the SMAL dataset, we randomly generate 400 shapes for following the shape sampling method in [15], where latent vectors are sampled from a normal distribution with zero mean and 0.2 standard deviations. We split them into 300 training shapes and 100 testing shapes. The Bone dataset consists of four categories of real bones: Femur, Tibia, Pelvis, and Scapula, where each category has 40 training and 10 testing shapes. The consistent correspondences are obtained from interpolating landmark correspondences marked by experts.

Baseline approaches. We evaluate on four baselines: **SP-Disentangle [63]**, **CoMA [39]**, **3DMM [7]**, and **MeshConv [64]**. They together represent the state-of-the-art results on learning mesh generators from a collection of meshes with dense correspondences. We evaluate the effectiveness of ARAPReg on these baselines and the absolute performance of our approach against these baselines.

Evaluation metrics. Besides qualitative evaluations, we employ the reconstruction error metric (c.f. [39, 7, 64]) for quantitative evaluations. Specifically, we compute the average per-vertex Euclidean distance for the input and reconstructed meshes. For VAE, the latent variable is given by the encoder . For AD, we optimize the latent variable to find the best reconstruction (c.f [59, 61]). The output shape is obtained by feeding the latent variable to the decoder.

6.2. Analysis of Results

Table 1 compares our approach and baseline approaches in terms of the reconstruction error. Under the AD framework, our approach reduces the reconstruction error of baseline approaches by 16.8%, 16.6%, and 6.7% on DFAUST, SMAL, and Bone, respectively. As the optimal latent-variable is optimized, AD framework achieves better quality than VAE framework.

	DFAUST	SMAL	Bone
SP-Disentangle. [63]	10.02	21.32	5.34
COMA[39]	8.80	14.52	4.14
3DMM[7]	7.39	17.78	4.03
MeshConv[64]	5.43	8.01	4.47
Ours-VAE (L1)	5.45	9.11	4.09
Ours-VAE (L1 + ARAP)	4.87	7.82	3.85
Ours-AD (L1)	5.17	8.74	3.91
Ours-AD (L1 + ARAP)	**4.52**	**6.68**	**3.76**

Table 1: Correspondence-based MSE reconstruction error (mm) on test sets of DFaust, SMAL and Bone.

	3DMM [7]	COMA [39]	MeshConv [64]
No-Reg	7.39	8.80	5.43
ARAPReg	6.72	4.87	5.02

Table 2: The effects of ARAPReg on different baselines on DFAUST dataset. The first row shows reported MSE reconstruction errors in their papers, and the second row shows results with ARAPReg on the same architecture with VAE training. ARAPReg achieves improvements on various baselines with different architectures.

Figure 5 illustrates the reconstruction errors visually. Our approach improves from baseline approaches considerably. In particular, it improves from the top-performing approach MeshConv [64] at locations with large deformations (e.g., arms of humans) and non-rigid deformations (e.g., arms and torsos of humans). These improvements come from modeling the preservation of the local rigidity among neighboring shapes in the underlying shape space. Please refer to the supp. material for more results.

Figure 8 and the supp. material shows randomly generated shapes under our trained full VAE and AD models (i.e., with ARAPReg). We can see that the generated shapes nicely preserve important shape features such as fingers and faces of human shapes and tails of animal shapes. Moreover, the generated shapes are different from the closest training shape, indicating that the learned mesh generator has a strong generalization ability.

6.3. Ablation Study

Table 1 shows our quantitative reconstruction results with and without ARAPReg under the VAE and AD settings. The effects of ARAPReg are salient. Under the AD setting, ARAPReg leads to 12.6%, 23.5%, and 4.5% reductions of the reconstruction error on DFAUST, SMAL, Bone, respectively. Table 2 further shows the effect of ARAPReg on various baselines under the VAE reconstruction pipeline. ARAPReg reduces the reconstruction error by building a better shape space that preserves the local rigidity constraint.

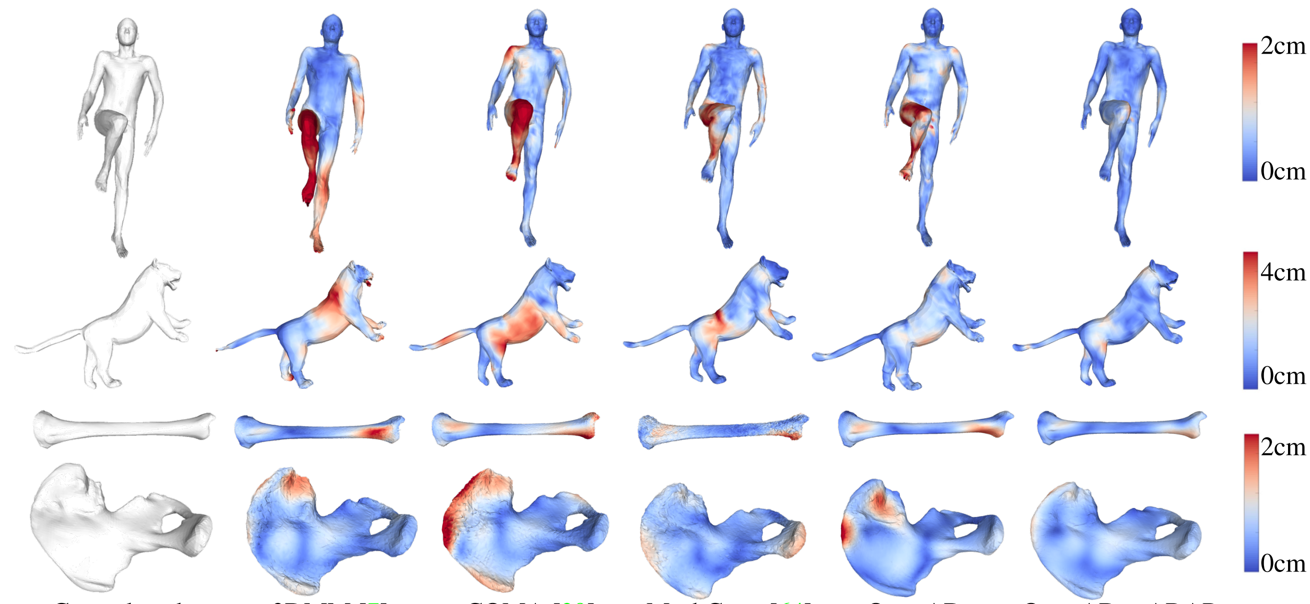

Figure 5: Qualitative comparison of reconstruction results. We show results using the AD generator w/w.o ARAPReg. Compared with baseline approaches, ours results with ARAPReg present less distortions and are locally smoother.

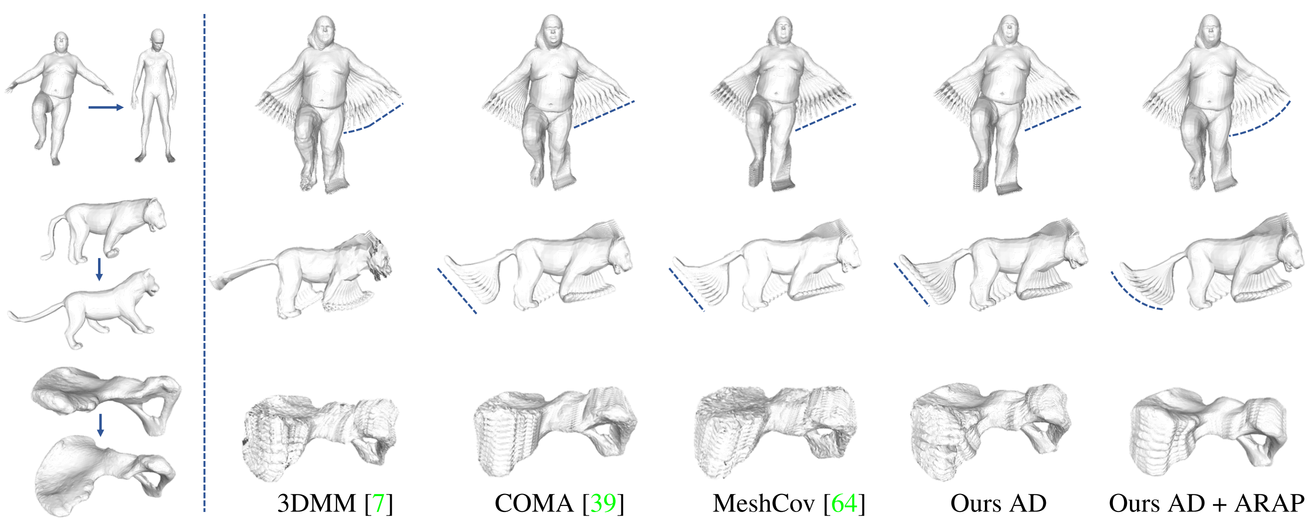

Figure 6: Interpolation results. The left column shows three groups of source and target shapes (connected by blue arrows). The remaining columns show ten intermediate shapes by linearly interpolating the latent codes of the source and target shapes. We show results using the AD generator w/w.o ARAPReg. Compared with baseline approaches, our results with ARAPReg show much smoother and more shape-preserving deformations.

6.4. Shape Interpolation

We proceed to evaluate the effects of ARAPReg for the application of shape interpolation. Given two shapes $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$, we first obtain their corresponding latent parameters $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$. For the VAE model, $\boldsymbol{z}_i$ comes from the encoder. For the AD model, $\boldsymbol{z}_i$ comes from optimizing the reconstruction error. The interpolation is then done by linearly interpolating $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$.

Figure 6 compares our approach and baseline approaches on shape interpolation. We can see that our approach's interpolated shapes are smoother and more shape-preserving than those of the baseline approaches. Specifically, prominent shape features such as fingers are better preserved in our approach. Moreover, our approach introduces less distortion among joint regions.

6.5. Shape Extrapolation

We also evaluate the effects of ARAPReg for the application of shape extrapolation. Given a center shape $\boldsymbol{g}$, we first obtain its corresponding latent parameters $\boldsymbol{z}$. For the VAE model, $\boldsymbol{z}$ comes from the encoder. For the AD

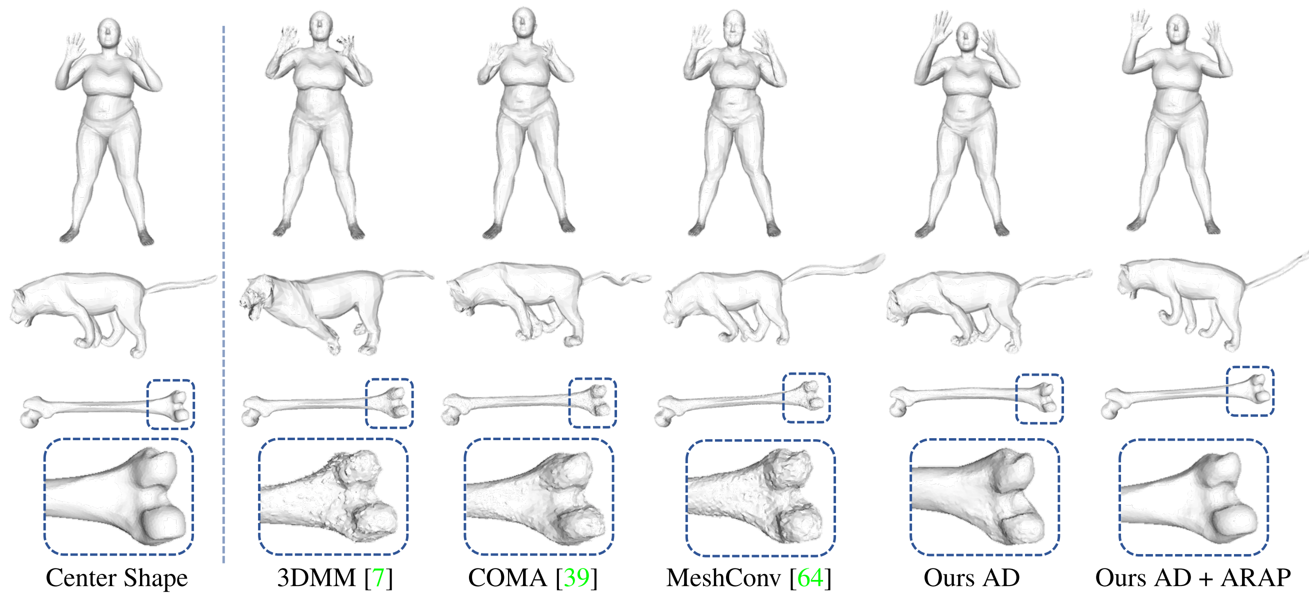

Figure 7: Extrapolation results. Around one test center shape (left column), we randomly perturb its latent code z within an Euclidean ball to generate perturbed shapes. We show results using the AD generator w/w.o ARAPReg. Our results with ARAPReg exhibit smooth and feature-preserving deformations.

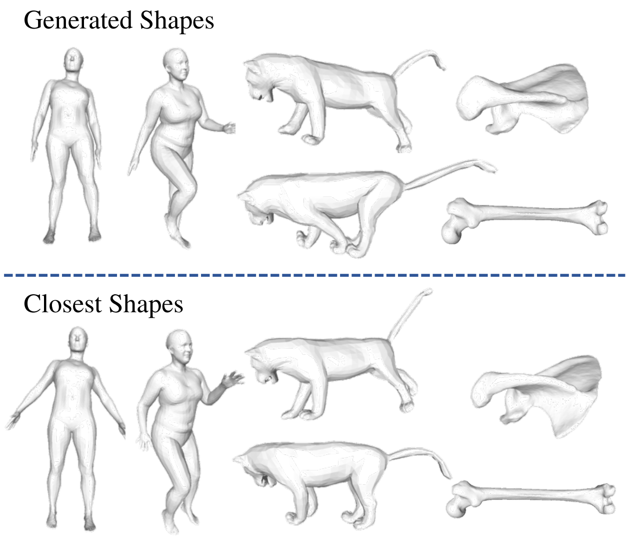

Figure 8: Randomly generated shapes from our AD + ARAPReg framework and their closest shapes in the training set. Our network is able to generate reasonable shapes that are not in the training shape collection. Due to space constraints, results of our VAE framework are in the supp. material.

model, z comes from optimizing the reconstruction error. The extrapolation is then done by randomly sampling $\tilde{z} \sim z + \mathcal{N}(0, \sigma^2 S)$, where S denotes scale for each latent dimension. We choose $\sigma = 0.2$ for all datasets.

Figure 7 compares our approach and baseline approaches on shape extrapolation. We can see that our approach's generated shapes are smoother and more reasonable than baseline approaches in areas such as tails of animals, hands and arms of human.

7. Conclusions and Limitations

This paper introduces ARAPReg, an unsupervised loss functional for training shape generators. Experimental results show that enforcing this loss on meshed shape generators improves their performance. The resulting mesh generators produce novel generated shapes that are shape-preserving at multiple scales.

ARAPReg has several limitations which can inspire future work. First, so far, ARAPReg only applies to training datasets with given correspondences. An interesting problem is to address unorganized shape collections that do not possess dense correspondences. Besides pre-computing correspondences, a promising direction is to explore the simultaneous learning of the shape correspondences and the shape generator. Another limitation of ARAPReg is that it targets realistically deformable shapes. Future directions are to study how to extend the formulation to handle synthetically generated shapes of any form and function.

Acknowledgement. Chandrajit Bajaj would like to acknowledge the support from NIH-R01GM117594, by the Peter O'Donnell Foundation, and in part from a grant from the Army Research Office accomplished under Cooperative Agreement Number W911NF-19-2-0333. Junfeng Jiang would like to acknowledge support from Jiangsu NSF under Grant BK20181158 and NSF of China (NSFC) under Grant 61772172. Qixing Huang is supported by NSF Career IIS-2047677, NSF HDR TRIPODS-1934932, and Gifts from Wormpex AI Research and Snap Inc.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J. Guibas. Learning representations and generative models for 3d point clouds. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 40–49, Stockholm, Sweden, 2018. PMLR.

[2] Tristan Aumentado-Armstrong, Stavros Tsogkas, Allan D. Jepson, and Sven J. Dickinson. Geometric disentanglement for generative latent shape models. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 8180–8189. IEEE, 2019.

[3] Elena Balashova, Vivek Singh, Jiangping Wang, Brian Teixeira, Terrence Chen, and Thomas A. Funkhouser. Structure-aware shape synthesis. In *2018 International Conference on 3D Vision, 3DV 2018, Verona, Italy, September 5-8, 2018*, pages 140–149. IEEE Computer Society, 2018.

[4] Heli Ben-Hamu, Haggai Maron, Itay Kezurer, Gal Avineri, and Yaron Lipman. Multi-chart generative surface modeling. *ACM Trans. Graph.*, 37(6):215:1–215:15, 2018.

[5] Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Dynamic FAUST: Registering human bodies in motion. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[6] Mario Botsch and Olga Sorkine. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):213–230, 2008.

[7] Giorgos Bouritsas, Sergiy Bokhnyak, Stylianos Ploumpis, Michael Bronstein, and Stefanos Zafeiriou. Neural 3d morphable models: Spiral convolutional networks for 3d shape representation learning and generation. In *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

[8] Zhangjie Cao, Qixing Huang, and Karthik Ramani. 3d object classification via spherical projections. In *2017 International Conference on 3D Vision, 3DV 2017, Qingdao, China, October 10-12, 2017*, pages 566–574, Qingdao, China, 2017. IEEE Computer Society.

[9] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 5939–5948, Long Beach, CA, USA, 2019. Computer Vision Foundation / IEEE.

[10] Taco S. Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical cnns. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[11] Luca Cosmo, Antonio Norelli, Oshri Halimi, Ron Kimmel, and Emanuele Rodolà. LIMP: learning latent shape representations with metric preservation priors. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part III*, volume 12348 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2020.

[12] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6545–6554, Honolulu, HI, USA, 2017. IEEE Computer Society.

[13] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. 3d object classification and retrieval with spherical cnns. *CoRR*, abs/1711.06721:1–9, 2017.

[14] Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao Zhang. Sdm-net: Deep generative network for structured deformable mesh. *ACM Trans. Graph.*, 38(6), Nov. 2019.

[15] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. 3d-coded : 3d correspondences by deep deformation. In *In ECCV 2018*, 2018.

[16] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 216–224, 2018.

[17] Marc Habermann, Weipeng Xu, Michael Zollhöfer, Gerard Pons-Moll, and Christian Theobalt. Deepcap: Monocular human performance capture using weak supervision. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 5051–5062. IEEE, 2020.

[18] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: A network with an edge. *ACM Trans. Graph.*, 38(4), July 2019.

[19] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(2):386–397, 2020.

[20] Qi-Xing Huang, Martin Wicke, Bart Adams, and Leonidas J. Guibas. Shape decomposition using modal analysis. *Comput. Graph. Forum*, 28(2):407–416, 2009.

[21] Angjoo Kanazawa, Michael J. Black, David W. Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 7122–7131. IEEE Computer Society, 2018.

[22] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 3907–3916, Salt Lake City, UT, USA, 2018. IEEE Computer Society.

[23] Martin Kilian, Niloy J. Mitra, and Helmut Pottmann. Geometric modeling in shape space. *ACM Trans. Graph.*, 26(3):64–es, July 2007.

[24] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[25] Chun-Liang Li, Manzil Zaheer, Yang Zhang, Barnabás Póczos, and Ruslan Salakhutdinov. Point cloud GAN. In *Deep Generative Models for Highly Structured Data, ICLR 2019 Workshop, New Orleans, Louisiana, United States, May 6, 2019*, pages 1–19, New Orleans, Louisiana, United States, 2019. OpenReview.net.

[26] Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. Grass: Generative recursive autoencoders for shape structures. *ACM Trans. Graph.*, 36(4):52:1–52:14, July 2017.

[27] Xueting Li, Sifei Liu, Shalini De Mello, Kihwan Kim, Xiaolong Wang, Ming-Hsuan Yang, and Jan Kautz. Online adaptation for consistent mesh reconstruction in the wild. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[28] Or Litany, Alexander M. Bronstein, Michael M. Bronstein, and Ameesh Makadia. Deformable shape completion with graph convolutional autoencoders. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 1886–1895, Salt Lake City, UT, USA, 2018. IEEE Computer Society.

[29] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Smpl: A skinned multi-person linear model. *ACM Trans. Graph.*, 34(6), Oct. 2015.

[30] Zhaoliang Lun, Matheus Gadelha, Evangelos Kalogerakis, Subhransu Maji, and Rui Wang. 3d shape reconstruction from sketches via multi-view convolutional networks. In *2017 International Conference on 3D Vision, 3DV 2017, Qingdao, China, October 10-12, 2017*, pages 67–77, Qingdao, China, 2017. IEEE Computer Society.

[31] Haggai Maron, Meirav Galun, Noam Aigerman, Miri Trope, Nadav Dym, Ersin Yumer, Vladimir G. Kim, and Yaron Lipman. Convolutional neural networks on surfaces via seamless toric covers. *ACM Trans. Graph.*, 36(4), July 2017.

[32] Lars M. Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 4460–4470, Long Beach, CA, USA, 2019. Computer Vision Foundation / IEEE.

[33] Behnam Neyshabur, Ryota Tomioka, Ruslan Salakhutdinov, and Nathan Srebro. Geometry of optimization and implicit regularization in deep learning. *CoRR*, abs/1705.03071, 2017.

[34] Eunbyung Park, Jimei Yang, Ersin Yumer, Duygu Ceylan, and Alexander C. Berg. Transformation-grounded image generation network for novel 3d view synthesis. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 702–711, Honolulu, HI, USA, 2017. IEEE Computer Society.

[35] Jeong Joon Park, Peter Florence, Julian Straub, Richard A. Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 165–174. Computer Vision Foundation / IEEE, 2019.

[36] William S. Peebles, John Peebles, Jun-Yan Zhu, Alexei A. Efros, and Antonio Torralba. The hessian penalty: A weak prior for unsupervised disentanglement. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part VI*, volume 12351 of *Lecture Notes in Computer Science*, pages 581–597. Springer, 2020.

[37] Helmut Pottmann, Qixing Huang, Bailin Deng, Alexander Schiftner, Martin Kilian, Leonidas Guibas, and Johannes Wallner. Geodesic patterns. *ACM Trans. Graph.*, 29(4), July 2010.

[38] Marie-Julie Rakotosaona and Maks Ovsjanikov. Intrinsic point cloud interpolation via dual latent space navigation. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part II*, volume 12347 of *Lecture Notes in Computer Science*, pages 655–672. Springer, 2020.

[39] 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.

[40] Daniel Ritchie, Anna Thomas, Pat Hanrahan, and Noah D. Goodman. Neurally-guided procedural models: Amortized inference for procedural graphics programs using neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 622–630, USA, 2016. Curran Associates Inc.

[41] Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. Csgnet: Neural shape parser for constructive solid geometry. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 5515–5523, Los Alamitos, CA, 2018. IEEE Computer Society.

[42] Ayan Sinha, Asim Unmesh, Qixing Huang, and Karthik Ramani. Surfnet: Generating 3d shape surfaces using deep residual networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 791–800, 2017.

[43] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, SGP '07, page 109–116, Goslar, DEU, 2007. Eurographics Association.

[44] Qingyang Tan, Lin Gao, Yu-Kun Lai, and Shihong Xia. Variational autoencoders for deforming 3d mesh models. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 5841–5850, Salt Lake City, UT, USA, 2018. IEEE Computer Society.

[45] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2107–2115, Venice,Italy, 2017. IEEE Computer Society.

[46] Edgar Tretschk, Ayush Tewari, Michael Zollhöfer, Vladislav Golyanik, and Christian Theobalt. DEMEA: deep mesh autoencoders for non-rigidly deforming objects. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part IV*, volume 12349 of *Lecture Notes in Computer Science*, pages 601–617, Glasgow,UK, 2020. Springer.

[47] Shubham Tulsiani, Hao Su, Leonidas J. Guibas, Alexei A. Efros, and Jitendra Malik. Learning shape abstractions by assembling volumetric primitives. In *The IEEE Conference*

on Computer Vision and Pattern Recognition (CVPR), July 2017.

[48] Nitika Verma, Edmond Boyer, and Jakob Verbeek. Feastnet: Feature-steered graph convolutions for 3d shape analysis. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 2598–2606, Salt Lake City, UT, USA, 2018. IEEE Computer Society.

[49] Michael Wand, Philipp Jenke, Qi-Xing Huang, Martin Bokeloh, Leonidas J. Guibas, and Andreas Schilling. Reconstruction of deforming geometry from time-varying point clouds. In Alexander G. Belyaev and Michael Garland, editors, *Proceedings of the Fifth Eurographics Symposium on Geometry Processing, Barcelona, Spain, July 4-6, 2007*, volume 257 of *ACM International Conference Proceeding Series*, pages 49–58, Avenue de Frontenex 32, 1207 Geneve, Switzerland, 2007. Eurographics Association.

[50] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single RGB images. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XI*, pages 55–71, 2018.

[51] Yifan Wang, Noam Aigerman, Vladimir G. Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. Neural cages for detail-preserving 3d deformations. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 72–80, Seattle, WA, USA, 2020. IEEE.

[52] Jiajun Wu, Tianfan Xue, Joseph J. Lim, Yuandong Tian, Joshua B. Tenenbaum, Antonio Torralba, and William T. Freeman. 3d interpreter networks for viewer-centered wireframe modeling. *Int. J. Comput. Vis.*, 126(9):1009–1026, 2018.

[53] 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 Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 82–90. OpenReview.net, Barcelona, Spain, 2016.

[54] Kai Xu, Vladimir G. Kim, Qixing Huang, Niloy Mitra, and Evangelos Kalogerakis. Data-driven shape analysis and processing. In *SIGGRAPH ASIA 2016 Courses*, SA '16, New York, NY, USA, 2016. Association for Computing Machinery.

[55] Weiwei Xu, Kun Zhou, Yizhou Yu, Qifeng Tan, Qunsheng Peng, and Baining Guo. Gradient domain editing of deforming mesh sequences. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, page 84–es, New York, NY, USA, 2007. Association for Computing Machinery.

[56] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge J. Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 4540–4549, Seoul, Korea (South), 2019. IEEE.

[57] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 206–215, Salt Lake City, UT, USA, 2018. IEEE Computer Society.

[58] Yong-Liang Yang, Yi-Jun Yang, Helmut Pottmann, and Niloy J. Mitra. Shape space exploration of constrained meshes. In *Proceedings of the 2011 SIGGRAPH Asia Conference*, SA '11, New York, NY, USA, 2011. Association for Computing Machinery.

[59] Zhenpei Yang, Lihang Liu, and Qixing Huang. Learning generative neural networks for 3d colorization. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 2580–2587, New York, USA, 2018. AAAI.

[60] Mehmet Ersin Yümer and Niloy J. Mitra. Learning semantic deformation flows with 3d convolutional networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI*, volume 9910 of *Lecture Notes in Computer Science*, pages 294–311, Amsterdam,The Netherlands, 2016. Springer.

[61] Amir Zadeh, Yao-Chong Lim, Paul Pu Liang, and Louis-Philippe Morency. Variational auto-decoder. *arXiv preprint arXiv:1903.00840*, 2019.

[62] Zaiwei Zhang, Zhenpei Yang, Congyang Ma, Linjie Luo, Alexander Huth, Etienne Vouga, and Qixing Huang. Deep generative modeling for scene synthesis via hybrid representations. *ACM Transaction on Graphics*, 2, 2020.

[63] Keyang Zhou, Bharat Lal Bhatnagar, and Gerard Pons-Moll. Unsupervised shape and pose disentanglement for 3d meshes. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXII*, volume 12367 of *Lecture Notes in Computer Science*, pages 341–357. Springer, 2020.

[64] Yi Zhou, Chenglei Wu, Zimo Li, Chen Cao, Yuting Ye, Jason Saragih, Hao Li, and Yaser Sheikh. Fully convolutional mesh autoencoder using efficient spatially varying kernels. *arXiv preprint arXiv:2006.04325*, 2020.

[65] Chuhang Zou, Ersin Yumer, Jimei Yang, Duygu Ceylan, and Derek Hoiem. 3d-prnn: Generating shape primitives with recurrent neural networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 900–909, Venice,Italy, 2017. IEEE Computer Society.

[66] 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.