Automatic Unpaired Shape Deformation Transfer

LIN GAO, Institute of Computing Technology, Chinese Academy of Sciences
JIE YANG, Institute of Computing Technology, Chinese Academy of Sciences and University of Chinese Academy of Sciences
YI-LING QIAO, Institute of Computing Technology, CAS and University of Chinese Academy of Sciences
YU-KUN LAI, Cardiff University
PAUL L. ROSIN, Cardiff University
WEIWEI XU, Zhejiang University
SHIHONG XIA, Institute of Computing Technology, Chinese Academy of Sciences

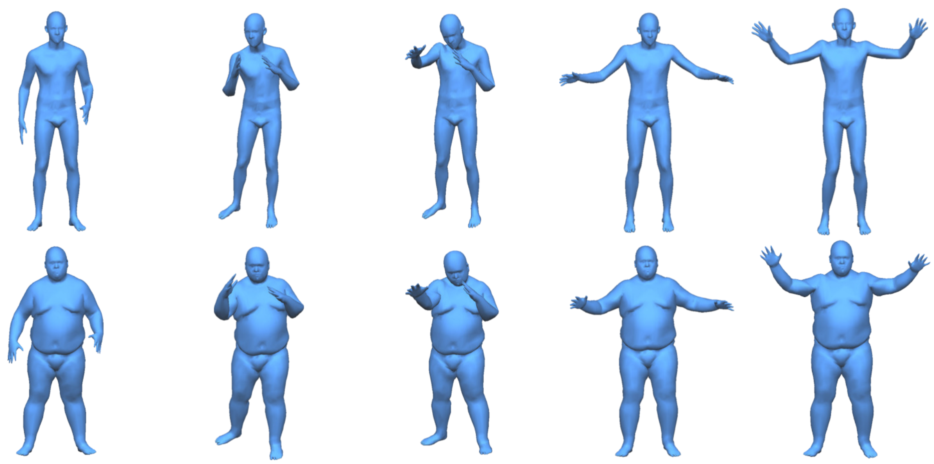


Fig. 1. Deformation transfer from a fit person to a fat person, both from the MPI DYNA dataset [Pons-Moll et al. 2015]. First row: source fit person shapes, second row: our results of deformation transfer to a fat person. Our method automatically transfers rich actions across shapes with substantial geometric differences without the need for specifying correspondences or shape pairs between source and target.

Transferring deformation from a source shape to a target shape is a very useful technique in computer graphics. State-of-the-art deformation transfer methods require either point-wise correspondences between source and target shapes, or pairs of deformed source and target shapes with corresponding deformations. However, in most cases, such correspondences are not available and cannot be reliably established using an automatic algorithm. Therefore, substantial user effort is needed to label the correspondences or to obtain and specify such shape sets. In this work, we propose a novel approach to automatic deformation transfer between two unpaired shape sets without correspondences. 3D deformation is represented in a high-dimensional space. To obtain a more compact and effective representation, two convolutional variational autoencoders are learned to encode source and target shapes to their latent spaces. We exploit a Generative Adversarial Network (GAN) to map deformed source shapes to deformed target shapes, both in the latent spaces, which ensures the obtained shapes from the mapping are indistinguishable from the target shapes. This is still an under-constrained problem, so we further utilize a reverse mapping from target shapes to source shapes and incorporate cycle consistency loss, i.e. applying both mappings should reverse to the input shape. This VAE-Cycle GAN (VC-GAN) architecture is used to build a reliable mapping between shape spaces. Finally, a similarity constraint is employed to ensure the mapping is consistent with visual similarity, achieved by learning a similarity neural network that takes the embedding vectors from the source and target latent spaces and predicts the light field distance between the corresponding shapes. Experimental results show that our fully automatic method is able to obtain high-quality deformation transfer results with unpaired data sets, comparable or better than existing methods where strict correspondences are required.

CCS Concepts: • **Computing methodologies → Shape modeling**; Animation;

Additional Key Words and Phrases: Deformation transfer, generative adversarial network, cycle consistency, visual similarity

Lin Gao, Jie Yang, Yi-Ling Qiao and Shihong Xia are with the Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, Chinese Academy of Sciences. Jie Yang and Yi-Ling Qiao are also with the University of Chinese Academy of Sciences. Yu-Kun Lai and Paul L. Rosin are with the School of Computer Science & Informatics, Cardiff University. Weiwei Xu is with the State Key Lab of CAD&CG, Zhejiang University. Authors' e-mails are: {gaolin, yangjie01, xsh}@ict.ac.cn, qiaoyiling15@mails.ucas.ac.cn, {LaiY4, RosinPL}@cardiff.ac.uk, xww@cad.zju.edu.cn. Shihong Xia is the corresponding author.

2018. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *ACM Transactions on Graphics*, https://doi.org/10.1145/3272127.3275028.

ACM Reference format:
Lin Gao, Jie Yang, Yi-Ling Qiao, Yu-Kun Lai, Paul L. Rosin, Weiwei Xu, and Shihong Xia. 2018. Automatic Unpaired Shape Deformation Transfer. *ACM Trans. Graph.* 0, 0, Article 0 (2018), 15 pages.
https://doi.org/10.1145/3272127.3275028

1 INTRODUCTION

Shape deformation is widely used in computer graphics. It is useful in geometric modeling for generating new shapes from existing ones and in computer animation for producing smooth animation sequences. However, producing realistic animation is time-consuming and requires artistic expertise. Deformation transfer, i.e. transferring deformation of one shape to another, provides a cost-effective solution to producing new deformation results by reusing existing ones.

Given two sets of deformable shapes (source and target) and a new deformed source shape, deformation transfer aims to produce realistic deformation of the target shape, visually corresponding to the given deformed source shape. It is a challenging task, since the source and target shapes can differ significantly. Existing state-of-the-art methods [Chu and Lin 2010; Sumner and Popović 2004] for surface-based deformation transfer rely on point-wise correspondences between source and target shapes. In general cases, there is no reliable automatic method to achieve this, so existing methods require the user to specify a sufficient number of corresponding points such that point-wise correspondences can be deduced. This process is tedious and often requires trial-and-error to ensure specified corresponding points provide sufficient constraints. An alternative approach considers semantic deformation transfer [Baran et al. 2009]. The method does not require point-wise correspondence between source and target shapes. However, it takes *paired* source and target shapes, assuming that each model in the source set is semantically related to the corresponding shape in the target set. In practice, however, if the source and target shape datasets are constructed independently, this property is unlikely to be satisfied.

In this work, we aim to develop a fully automatic algorithm to deform target shapes in a way as similar as possible to the source deformed shapes, which none of the existing deformation transfer methods can achieve. To make this seemingly impossible task a reality, we exploit the learning capability of deep neural networks to learn how shapes deform naturally from a given dataset, provide a differentiable metric measuring visual similarity and build reliable mapping between the latent spaces with cycle-consistency. This is inspired by how humans perform this task, by observing the deformed shapes to learn their characteristics, considering the similarity between source and target shapes, and thinking about how the target shapes should deform to resemble source shapes. An example of our method is shown in Fig. 1 where the deformation of a fit person is automatically transferred to that of a fat person, with substantial body shape differences. Unlike previous methods, we do not require point-wise correspondences between source and target shapes, or paired source and target shapes as input. Instead, source and target shape sets may contain different deformations, as long as they are both sufficient to cover the relevant deformation spaces. This greatly reduces user efforts and allows using two independent shape deformation datasets. To achieve this, we propose a cycle-consistent Generative Adversarial Network (GAN) architecture for mesh deformation transfer. To ensure learning efficiency with a relatively small number of training examples, especially because plausible deformations form a much lower dimensional manifold in the high-dimensional deformation space, we introduce a convolutional autoencoder to represent shape deformations in a compact latent space. To ensure effective transfer, we further propose a neural network to measure visual similarity. The main contributions of this work are summarized as follows:

- This is the first *automatic* work to transfer deformation between *unpaired* shape datasets. To achieve this, we present an efficient and differentiable method, which is composed of a variational autoencoder to encode shape deformations, a differentiable network to measure visual similarity, and a cycle-consistent GAN for reliable mapping between latent spaces.
- We propose a novel neural network to measure the visual similarity between deformed shape pairs, which is differentiable and efficiently approximates light field distances. This network is the key to make the whole approach differentiable and trainable.
- We also propose a novel mesh-based convolutional variational autoencoder (VAE) to encode a shape set with flexible deformations in a compact latent space, which copes well with large deformations, supports generating new shapes in the space, and has good generalizability.

All the network components work together tightly and each is indispensable. The mesh-based convolutional autoencoder is used to learn and describe the plausible deformation space for generating natural shapes; the similarity metric provides a differentiable approximation to the Light Field Distance (LFD) [Chen et al. 2003], enabling our whole deep learning architecture to perceive intricate visual similarities across 3D model domains. Our method also benefits from the cycle consistency applied to the GAN network [Zhu et al. 2017] to build reliable mapping between two spaces. We further consider a simple extension of our method to semantic deformation transfer by utilizing a small number of paired shapes for learning a *semantic* similarity metric which cannot be characterized by visual similarity.

In Sec. 2, we review the work most related to ours. We then give the detailed description of our method, including overall architecture and loss functions in Sec. 3. Implementation details are presented in Sec. 4. We present experimental results, including extensive comparisons with state-of-the-art methods in Sec. 5, and finally, we draw conclusions in Sec. 6.

2 RELATED WORK

Shape deformation is an active research topic. A comprehensive survey of relevant techniques can be found in [Gain and Bechmann 2008]. We now review techniques for deformation transfer and deep learning which are related to this work.

Mesh Deformation Transfer. Sumner et al. [2004] performed pioneering work for mesh deformation transfer. The method requires point-wise correspondences between source and target shapes. Local shape deformation is then represented using deformation gradients, which are transferred from the source to the target shapes. This method however relies on specifying typically a large number of correspondences to cope with the differences between shapes since deformation gradients are local. Moreover, the method may transfer geometric details from the source shape to the target, which is undesirable. To address this, Chu and Lin [2010] proposed a method that further projected the deformed shape to the manifold of the target shape space, under the assumption that the target shape set provides sufficient coverage of plausible deformations. To reduce user effort, Yang et al. [2018] developed a method to automatically choose a set of suitable key points on the source shape, although the corresponding points on the target shape still require to be manually specified. Instead of specifying point-wise correspondences, Baran et al. [2009] proposed a different approach where the input is a set of source shapes and the same number of target shapes, where the corresponding pair of source and target shapes have related semantic meaning. Their method achieves semantic deformation transfer by representing each deformed shape in the source sequence using a combination of given shapes in the source set, and producing the target deformed shape by utilizing the combination weights with shapes in the target set. The method produces interesting results, although the required input is not generally available if the two shape sets are obtained independently, which restricts its use. Our method is fundamentally different from these works: by utilizing the learning capability of a novel GAN-based architecture we are able to take *unpaired* source and target shape sets and do not require point-wise or shape-wise correspondences between the sets.

For shapes which are not manifold triangle meshes, e.g. triangle soups or tetrahedra, methods have been developed [Ben-Chen et al. 2009; Chen et al. 2010] for transferring deformation using cages that enclose the shapes to be transferred. However, effort is needed to construct cages, and such methods may erroneously deform spatially adjacent regions if they happen to fit in the same cage. To cope with shapes involving multiple connected components, Zhou et al. [2010] developed a method based on a graph structure. These methods similarly require input for correspondences. Recent work [Corman et al. 2017; Rustamov et al. 2013] develops effective approaches to measuring shape differences, which are used for embedding shape collections. Given a shape in the first collection, these methods can be used to find a similar shape in the second collection without known correspondence. However, such methods do not synthesize *new* shapes, and therefore may not always be able to find suitable corresponding shapes.

Deep Learning for 3D Shape Processing. We exploit the learning capability of neural networks in this work, which have achieved great success in 2D image processing. In recent years, effort has been made to process 3D shapes, which are more challenging due to the higher dimension and irregular connectivity.

For 3D shape recognition and analysis, shapes can be represented using multi-view projection images along with 2D-CNNs for classification [Shi et al. 2015]. Such approaches are used in [Huang et al. 2018] to learn local shape descriptors useful for shape correspondence and segmentation. Shapes can also be represented using voxels with 3D-CNNs extended from 2D [Maturana and Scherer 2015]. Tulsiani et al. [2017] use this representation to approximate shapes with cuboids, giving an abstract representation. To improve efficiency, Wang et al. [2017a] propose an octree structure to represent 3D shapes and perform convolutional operations on the octree to build CNNs. Alternatively, meshes can be treated as irregular graphs, and CNNs are extended to handle such graphs either in the spectral [Bruna et al. 2013; Defferrard et al. 2016; Henaff et al. 2015] or the spatial domain [Duvenaud et al. 2015; Niepert et al. 2016]. These representations are used for shape correspondences [Boscaini et al. 2016a,b] and shape segmentation [Yi et al. 2017a]. Maron et al. [2017] parameterize a sphere-type shape to a planar flat-torus with a well-defined convolutional operator to build CNN models.

For 3D shape synthesis, methods have been developed using voxel-based 3D CNNs, including deep belief networks [Wu et al. 2015] and GANs [Wu et al. 2016]. The latter is pioneering work that uses a GAN to generate 3D shapes. However, it uses a voxel representation with limited resolution, whereas our method aims to automatically transfer mesh deformations with rich geometry details. Moreover, [Wu et al. 2016] needs paired images and 3D models, while our method is fully unsupervised. Liu et al. [2017] extend a 3D GAN to support interactive editing, where a projection operator is provided to map user designed voxels to more detailed shapes. Sharma et al. [2016] use an unsupervised voxel-based autoencoder for applications such as denoising. The voxel-based representation has high space complexity due to its cubic nature, and therefore can only be practically used to synthesize coarse shapes. An alternative approach uses geometry images and an image-based ResNet architecture to synthesize 3D shapes [Sinha et al. 2017]. Geometry images allow details to be better preserved, but they also have unavoidable distortions and seams, and require shapes to be aligned to facilitate processing. Taking aligned meshes with consistent segmentation as input, neural networks are also used to synthesize 3D shapes with pre-segmented parts [Li et al. 2017; Nash and Williams 2017]. These methods focus on man-made objects and do not support non-rigid deformation.

Some works address relevant but different tasks from synthesizing general 3D shapes. Han et al. [2017] use a CNN to model 3D faces with sketches. In [Sung et al. 2017], neural networks are employed for assembly-based modeling with suggestions of complementary components and their placement. Other research considers joint embeddings of 2D images and 3D shapes [Li et al. 2015], and maps 2D images to corresponding 3D shapes [Choy et al. 2016; Fan et al. 2017; Girdhar et al. 2016; Yan et al. 2016].

None of these works address the problem we study here, namely deformation transfer.

Deep Learning for Non-Rigid Deformation. Our method deals with deformable mesh datasets. To analyze them, Tan et al. [2018a] firstly proposed a mesh variational autoencoder network with fully connected layers to encode meshes using a recent rotation-invariant representation [Gao et al. 2016], with applications to shape embedding and synthesis. However, the use of fully connected layers restricts its generalizability. An alternative mesh-based convolutional autoencoder was proposed in [Litany et al. 2017] for completion of deformable shapes. Their method however takes Euclidean coordinates directly and thus cannot handle large rotations well. The work

[Tan et al. 2018b] proposed a convolutional autoencoder to encode deformable mesh sets with sparse constraints to extract localized deformation components. Their architecture is not variational and thus not suitable for generating new data. Our method proposes a new convolutional variational autoencoder with a representation that handles large rotations, which effectively embeds deformable shapes to a compact latent space.

Image Transfer using GAN. Synthesizing new images by transferring information from existing ones has been an active topic for decades. Recent work uses the GAN-based architecture, where the joint training of a generator and a discriminator helps ensure that the synthesized images have characteristics indistinguishable from the target sets. This is related to our work, although we deal with deformation transfer between shape sets. The work pix2pix [Isola et al. 2017] uses a conditional GAN for paired image-to-image translation, without the requirement of tuning loss functions. DualGAN [Yi et al. 2017b] uses two identical GANs to achieve image-to-image translation in an unsupervised manner. The method CycleGAN [Zhu et al. 2017] allows image transfer with unpaired training examples, where the key idea is to introduce a cycle consistency loss to regularize the mapping. We instead perform deformation synthesis in the space of 3D shapes utilizing a GAN, since GANs are proved to be capable for such tasks [Wang et al. 2017b]. Unlike existing CycleGAN works that focus on images, we propose the first work for automatic mesh deformation transfer with unpaired source and target shapes, by generalizing the CycleGAN architecture to deal with challenging 3D shapes.

3 METHOD

Given two sets of *unpaired* shapes, a source shape set $\mathcal{S}$ and a target shape set $\mathcal{T}$, as well as a deformed source shape $\mathbf{s}$, our aim is to produce a deformed target shape $\mathbf{t}$ which has visually similar deformation as $\mathbf{s}$. Shapes in the same set $\mathcal{S}$ or $\mathcal{T}$ have the same connectivity. Many shape datasets satisfy this: they are either obtained by deforming a mesh model, or fitting a template mesh model to deformed shapes.

We do not assume shapes in $\mathcal{S}$ correspond to specific shapes in $\mathcal{T}$, although we assume that $\mathcal{S}$ and $\mathcal{T}$ provide sufficient coverage of typical deformations of the relevant shapes. We learn a deep model with $\mathcal{S}$ and $\mathcal{T}$ as training examples. Once the deep model is trained, a shape $\mathbf{t}$ is generated for each input $\mathbf{s}$.

3.1 Overview

The overall architecture of the VAE-Cycle GAN (VC-GAN) is illustrated in Fig. 2. Since mesh models typically contain thousands of vertices and mesh datasets may have less than a hundred shapes, establishing a mapping from source shapes $\mathcal{S}$ to target shapes $\mathcal{T}$ can be underdetermined. Therefore, before setting up the mapping functions, we employ variational autoencoders to encode the source and target shape sets into compact latent spaces $\tilde{\mathcal{S}}$ and $\tilde{\mathcal{T}}$. This not only makes the training process better constrained, but also ensures generated shapes conform to the deformation space. Denote by $Enc_{\mathcal{S}}$ and $Dec_{\mathcal{S}}$ the encoder and decoder for the source shape set. $Enc_{\mathcal{T}}$ and $Dec_{\mathcal{T}}$ are similarly defined for the target shape set.

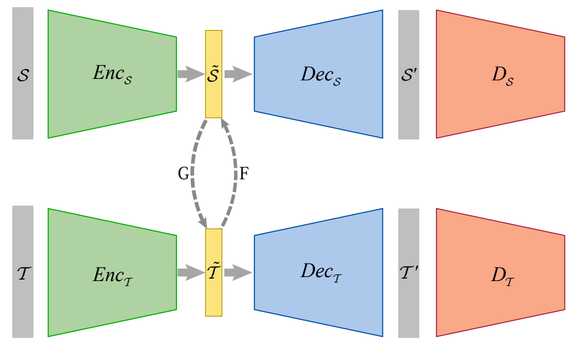


Fig. 2. The overall architecture of our VAE Cycle-Consistent Adversarial (VC-GAN) Network for deformation transfer. $\mathcal{S}$ and $\mathcal{T}$ are two shape datasets. $Enc_{\mathcal{S}}$ and $Dec_{\mathcal{S}}$ are the convolutional variational encoder and decoder for the shape set $\mathcal{S}$, which embed shapes $\mathcal{S}$ in a latent space $\tilde{\mathcal{S}}$. Similarly, $Enc_{\mathcal{T}}$ and $Dec_{\mathcal{T}}$ are the encoder and decoder for the shape set $\mathcal{T}$. Our CycleGAN has generator G that maps vectors from the latent space $\tilde{\mathcal{S}}$ to $\tilde{\mathcal{T}}$, and generator F that reversely maps $\tilde{\mathcal{T}}$ to $\tilde{\mathcal{S}}$. Our discriminators $D_{\mathcal{S}}$ and $D_{\mathcal{T}}$ are defined with the output of the decoders $\mathcal{S}'$ and $\mathcal{T}'$ as input, which are used to distinguish synthetic shapes from dataset shapes.

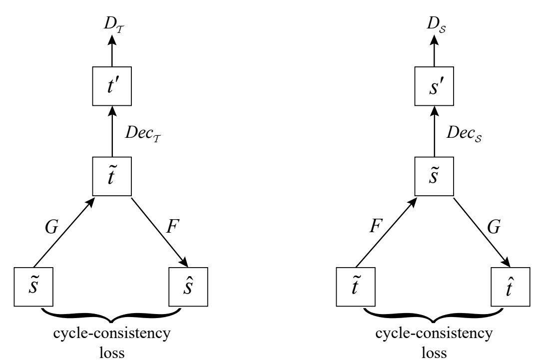


Fig. 3. Illustration of the cycle-consistency loss and adversarial loss. Left: Applying the generator G to a source shape $\tilde{\mathbf{s}}$ in the source latent space generates a target shape $\tilde{\mathbf{t}}$ in the target latent space. Further applying F to $\tilde{\mathbf{t}}$ gives a shape $\hat{\mathbf{s}}$ which should be near identical to $\tilde{\mathbf{s}}$. Cycle-consistency loss measures the difference between $\tilde{\mathbf{s}}$ and $\hat{\mathbf{s}}$. The adversarial loss is defined by first applying the decoder $Dec_{\mathcal{T}}$ to $\tilde{\mathbf{t}}$, followed by a discriminator network $D_{\mathcal{T}}$. Right: Similar to the case on the left, but starting from a shape $\tilde{\mathbf{t}}$ in the target latent space.

See Sec. 3.2 for details of our convolutional variational autoencoder (VAE).

Our mapping functions $G : \tilde{\mathcal{S}} \to \tilde{\mathcal{T}}$ and $F : \tilde{\mathcal{T}} \to \tilde{\mathcal{S}}$ are defined between the latent spaces. To ensure the mapping is meaningful, three types of regularization terms are employed including adversarial loss, cycle-consistency loss and visual similarity loss. Measuring the visual similarity between shapes with different topologies is not trivial and we introduce a dedicated neural network to predict this.

The first loss is the adversarial loss (Sec. 3.3) which discriminates the generated shape $G(\cdot)$ from the target shape sets. In the joint training process, it ensures the generated shape belongs to the target space. As we will show later, since the latent space from the convolutional VAE $\tilde{\mathcal{T}}$ has a Gaussian distribution, the discriminator cannot be effectively defined in the latent space to differentiate genuine and synthesized shapes. Instead, we define the discriminator in the decoded space. Let $D_{\mathcal{S}}$ and $D_{\mathcal{T}}$ be the discriminators in the source $\mathcal{S}$ and target $\mathcal{T}$ shape spaces. $D_{\mathcal{T}}$ and $D_{\mathcal{S}}$ take $Dec_{\mathcal{T}}(G(\tilde{\mathbf{s}}))$ and $Dec_{\mathcal{S}}(F(\tilde{\mathbf{t}}))$ as input respectively, where $\tilde{\mathbf{s}}$ and $\tilde{\mathbf{t}}$ are source and target shapes encoded in the latent spaces; see Fig. 3. The discriminator is a mesh-based convolutional neural network whose task is to discriminate between the transferred meshes (fake) and existing meshes in the target dataset (real).

The second loss is the cycle-consistency loss (Sec. 3.4), which ensures that the mapping from one domain to the other domain followed by the reverse mapping returns to the starting point. The cycle-consistency loss includes the forward cycle-consistency loss: $\tilde{\mathbf{s}} \rightarrow G(\tilde{\mathbf{s}}) \rightarrow F(G(\tilde{\mathbf{s}})) \approx \tilde{\mathbf{s}}$ and backward cycle-consistency loss: $\tilde{\mathbf{t}} \rightarrow F(\tilde{\mathbf{t}}) \rightarrow G(F(\tilde{\mathbf{s}})) \approx \tilde{\mathbf{t}}$; see Fig. 3.

For the deformation transfer task, the transferred target shape should be similar to the source shape being transferred. The visual similarity loss is employed to measure the visual similarity between the source shape and the transferred target shape. The forward and backward visual similarity losses are defined as $V(G(\tilde{\mathbf{s}}), \tilde{\mathbf{t}})$ and $V(F(\tilde{\mathbf{t}}), \tilde{\mathbf{s}})$, where $V(\cdot)$ is a fully connected neural network to measure the visual similarity between two shapes represented in the latent spaces; see Sec. 3.5 for more details.

3.2 Convolutional Variational Autoencoder

To cope with large deformations, we represent each deformed shape using a recently proposed ACAP (as consistent as possible) shape deformation representation [Gao et al. 2017], which handles large rotations well, and has features defined only on vertices, making convolutional operators easier to define. This is different from alternative representations such as [Gao et al. 2016] where features are also associated with edges. Take $\mathcal{S}$ for example, and the same process is applied to $\mathcal{T}$. Assume $\mathcal{S}$ contains N shapes with the same connectivity, each denoted as S_m. $\mathbf{p}_{m,i} \in \mathbb{R}^3$ is the i^{th} vertex on the m^{th} mesh. The deformation gradient $\mathbf{T}_{m,i} \in \mathbb{R}^{3\times3}$ representing local shape deformations can be obtained by minimizing:

$$\arg\min_{\mathbf{T}_{m,i}} \sum_{j\in\mathcal{N}_i} c_{ij} \|(\mathbf{p}_{m,i} - \mathbf{p}_{m,j}) - \mathbf{T}_{m,i}(\mathbf{p}_{1,i} - \mathbf{p}_{1,j})\|_2^2.$$

where c_{ij} is the cotangent weight and $\mathcal{N}_i$ represents the 1-ring neighbors of the i^{th} vertex. $\mathbf{T}_{m,i}$ can be decomposed into $\mathbf{R}_{m,i}\mathbf{S}_{m,i}$ where $\mathbf{R}_{m,i}$ is the rotation and $\mathbf{S}_{m,i}$ is the scale/shear deformation. The rotation matrix $\mathbf{R}_{m,i}$ can be represented by the rotation axis $\boldsymbol{\omega}_{m,i}$ and the associated rotation angle $\theta_{m,i}$. The mapping from the rotation matrix to rotation axis and angle is one to many, and the possible solutions are in the set $\Omega_{m,i}$:

$$\Omega_{m,i} = \left\{(\boldsymbol{\omega}_{m,i}, \theta_{m,i} + t\cdot 2\pi), (-\boldsymbol{\omega}_{m,i}, -\theta_{m,i} + t\cdot 2\pi)\right\} \quad (1)$$

For shapes with large-scale rotations, the adjacent vertices may have inconsistent rotation angles and rotation axes, which will lead to artifacts during shape blending and synthesis, as shown in [Gao et al. 2017]. To solve this problem, [Gao et al. 2017] proposes a method based on global integer programming to resolve rotation ambiguities and ensure consistency. Two global integer programming optimizations are used to solve for rotation axes and angles to make them as-consistent-as possible. For each vertex, this gives a feature vector $\mathbf{q}_{m,i} \in \mathbb{R}^9$. The components of $\mathbf{q}_{m,i}$ come from the combination of the rotation matrix $\mathbf{R}_{m,i}$ and scale/shear matrix $\mathbf{S}_{m,i}$ of the vertex. The logarithm of $\mathbf{R}_{m,i}$ is a skew-symmetry matrix and $\mathbf{S}_{m,i}$ is a symmetry matrix, so we extract a 3-dimensional vector for $\mathbf{R}_{m,i}$ and a 6-dimensional vector for $\mathbf{S}_{m,i}$ using non-duplicated entries and concatenate them to a 9-dimensional vector. As shown in [Tan et al. 2018b], this ACAP representation is appropriate for mesh-based convolutional neural networks. Following [Tan et al. 2018b], we rescale each dimension of $\mathbf{q}_{m,i}$ independently to $[-0.95, 0.95]$ before feeding in to the convolutional VAE, and scale it back from the output of the convolutional VAE, such that the *tanh* activation function can be applied.

Table 1. Mean RMS (root mean square) reconstruction errors of applying our method to generate unseen data using uniform or cotangent [Meyer et al. 2003] weight matrices for the convolutional operator on the ball dataset [Rustamov et al. 2013] (see Fig. 11(a)). We randomly choose 75% of the dataset as the training set and the remaining 25% of the dataset as the test set.

Dataset	Method	
	Uniform Weight	Cotangent Weight
Balls	0.0059	0.0080

The overall architecture of our convolutional VAE is illustrated in Fig. 4. For meshes with v vertices, the input $\mathbf{s}$ to the VAE is $9 \times v$ dimensional. Unlike [Tan et al. 2018a] which uses fully connected layers, we use *convolutional* layers which have better generalizability. As illustrated in Fig. 5, we use a mesh-based convolution operator [Duvenaud et al. 2015; Tan et al. 2018b] where the output at a vertex is obtained as a linear combination of input in its 1-ring neighbors along with a bias. The output of the operator $\mathbf{y}_i$ for the i^{th} vertex is defined as follows:

$$\mathbf{y}_i = \mathbf{W}_{point}\mathbf{s}_i + \mathbf{W}_{neighbor}\frac{1}{D_i}\sum_{j=1}^{D_i}\mathbf{s}_{n_{ij}} + \mathbf{b}, \quad (2)$$

where $\mathbf{s}_i$ is the input feature vector for the i^{th} vertex, D_i is the degree of the i^{th} vertex, n_{ij} $(1 \le j \le D_i)$ is the j^{th} neighbor of the i^{th} vertex. $\mathbf{W}_{point}, \mathbf{W}_{neighbor} \in \mathbb{R}^{9\times9}$ and $\mathbf{b} \in \mathbb{R}^9$ are the weights and bias. Following a convolutional neural network, these weights and the bias are shared by all the neighborhoods within each convolutional layer and learned during training. Since shapes in the same dataset contain complex deformations, their intrinsic and extrinsic geometries can change substantially. Therefore, uniform weights as given in the definition above, whereby neighboring vertices contribute equally in the convolution operator, are beneficial and used in our experiments as they only depend on the topology. As shown in Table 1, using uniform weights gives better performance than using cotangent weights [Meyer et al. 2003] commonly used in mesh processing.

After connecting several convolutional layers with the same size, these nodes are further connected with a fully connected layer. The architecture of this convolutional variational autoencoder is shown in Fig. 4. The work [Tan et al. 2018b] is not a variational

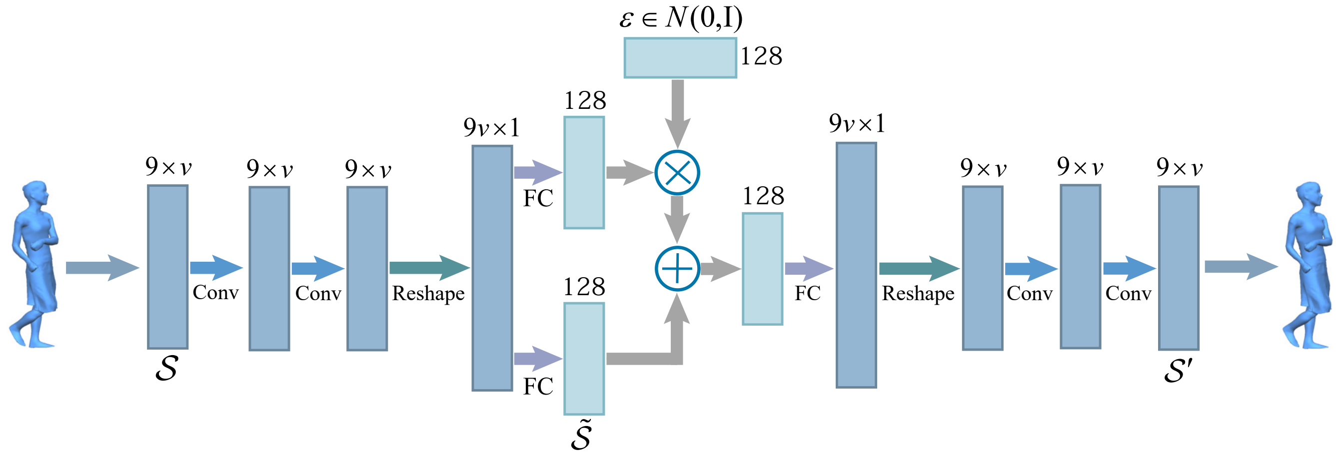


Fig. 4. The architecture of our convolutional variational autoencoder. The input is a deformation representation where each vertex is represented in a 9-dimensional feature vector and v is the number of vertices. Conv and FC refer to convolutional and fully connected layers, respectively. ε is a random variable with a Gaussian distribution with $\mathbf{0}$ mean and unit variance. The encoder encodes shapes from shape set $\mathcal{S}$ in a latent space $\tilde{\mathcal{S}}$, and the decoder recovers the shape $\mathcal{S}'$.

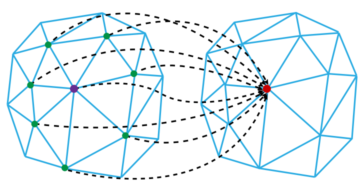
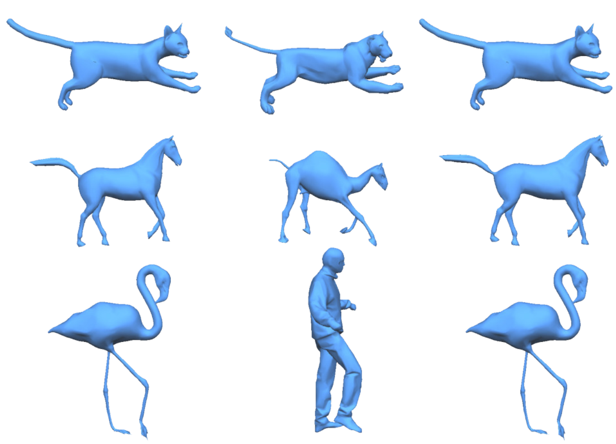


Fig. 5. Illustration of the convolutional operator on meshes. The result of convolution for each vertex is obtained by a linear combination from the input in the 1-ring neighbors of the vertex, along with a bias.



(a) $\tilde{\mathbf{s}}$ (b) $G(\tilde{\mathbf{s}})$ (c) $F(G(\tilde{\mathbf{s}}))$

Fig. 6. Results demonstrating cycle consistency mapping, showing reconstruction results of (a) source shape $\tilde{\mathbf{s}}$, (b) the transferred shape $G(\tilde{\mathbf{s}})$, (c) after mapping back $F(G(\tilde{\mathbf{s}}))$. We apply the appropriate decoder to recover the shape for visualization.

autoencoder, and cannot be used to synthesize varied shapes given the same input.

Let $Enc(\cdot)$ and $Dec(\cdot)$ be the encoder and decoder of our VAE network. $\mathbf{s}$ represents the input shape from dataset $\mathcal{S}$, $\tilde{\mathbf{s}} = Enc(\mathbf{s})$ is the encoded latent vector and $\mathbf{s}' = Dec(\tilde{\mathbf{s}})$ is the reconstructed shape. Our convolutional VAE minimizes the following loss:

$$L_{VAE} = L_{recon} + \alpha_1 L_{KL} + \alpha_2 L_{RegVAE} \tag{3}$$

where α_1 and α_2 are relative weights of different loss terms, $L_{recon} = \frac{1}{|\mathcal{S}|}\sum_{\mathbf{s}\in\mathcal{S}} ||\mathbf{s} - \mathbf{s}'||_2^2$ denotes the MSE (mean square error) reconstruction loss to ensure faithful reconstruction, $L_{KL} = D_{KL}(q(\tilde{\mathbf{s}}|\mathbf{s})|p(\tilde{\mathbf{s}}))$ is the KL divergence to promote Gaussian distribution in the latent space, where $q(\tilde{\mathbf{s}}|\mathbf{s})$ is the posterior distribution given input shape $\mathbf{s}$, and $p(\tilde{\mathbf{s}})$ is the Gaussian prior distribution. L_{RegVAE} is the squared ℓ_2 norm regularization term of the network parameters used to avoid overfitting. The Gaussian distribution makes it effective to generate new shapes by sampling in the latent space, which is used for training the GAN model, as we will explain later. The minimization of the above loss L_{VAE} is performed by gradient descent using the ADAM (adaptive moment estimation) solver. The parameters and types of layers of this convolutional variational autoencoder are evaluated in the supplementary material.

3.3 Adversarial Loss

The adversarial losses are applied to both mapping functions G and F between the latent spaces. For the mapping function $G : \tilde{\mathcal{S}} \to \tilde{\mathcal{T}}$, it is defined with the discriminator neural network $D_{\mathcal{T}}$, as follows:

$$\begin{aligned} L_{GAN-G}(G, D_{\mathcal{T}}, \tilde{\mathcal{S}}, \tilde{\mathcal{T}}) = \mathbb{E}_{\tilde{\mathbf{t}}\sim P_{data}(\tilde{\mathcal{T}})}[\log D_{\mathcal{T}}(\tilde{\mathbf{t}})] \\ + \mathbb{E}_{\tilde{\mathbf{s}}\sim P_{data}(\tilde{\mathcal{S}})}[\log(1 - D_{\mathcal{T}}(Dec_{\mathcal{T}}(G(\tilde{\mathbf{s}}))))], \end{aligned} \tag{4}$$

where $P_{data}(\cdot)$ represents the data distribution. $\mathbb{E}$ is the expected value of the distribution. The mapping G is used to generate $G(\tilde{\mathcal{S}})$ that has similar deformations in the latent space of $\tilde{\mathcal{T}}$. The discriminator $D_{\mathcal{T}}$ aims to distinguish the generated deformation shapes

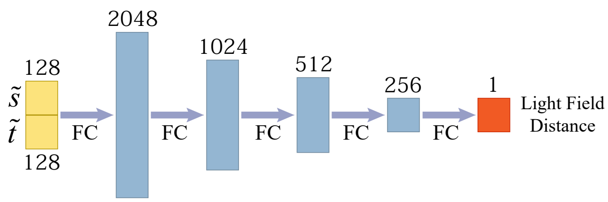


Fig. 7. The architecture of the fully connected neural network to learn the visual similarity distance between two shapes $\tilde{\mathbf{s}}$ and $\tilde{\mathbf{t}}$ in the latent space. FC refers to fully connected layers.

$Dec_{\mathcal{T}}(G(\tilde{\mathbf{s}}))$ after decoding from real shape samples $\mathbf{t}$. It is calculated using a convolutional neural network (see details in Sec. 4). The adversary $D_{\mathcal{T}}$ aims to maximize the objective function while the mapping function G aims to minimize it. This is equivalent to $\min_G \max_{D_{\mathcal{T}}} L_{GAN-G}(G, D_{\mathcal{T}}, \tilde{\mathcal{S}}, \tilde{\mathcal{T}})$. Similarly, the inverse mapping function F and its associated discriminator $D_{\mathcal{S}}$ are optimized by $\min_F \max_{D_{\mathcal{S}}} L_{GAN-F}(F, D_{\mathcal{S}}, \tilde{\mathcal{T}}, \tilde{\mathcal{S}})$. We define L_{GAN} to be the sum of both GAN losses:

$$L_{GAN}(G, F, D_{\mathcal{S}}, D_{\mathcal{T}}) = L_{GAN-G} + L_{GAN-F}. \tag{5}$$

3.4 Cycle Consistency Loss

Cycle-consistency loss is known to be effective to better constrain the network to produce more stable results and avoid over-fitting. As illustrated in Fig. 3, for each shape feature $\tilde{\mathbf{s}}$ in the latent space $\tilde{\mathbf{S}}$, applying G followed by F should bring it back to the original feature $\tilde{\mathbf{s}} \to G(\tilde{\mathbf{s}}) \to F(G(\tilde{\mathbf{s}})) = \hat{\mathbf{s}} \approx \tilde{\mathbf{s}}$. Similarly for $\tilde{\mathbf{t}}$, it satisfies the following cycle consistency: $\tilde{\mathbf{t}} \to F(\tilde{\mathbf{t}}) \to G(F(\tilde{\mathbf{t}})) = \hat{\mathbf{t}} \approx \tilde{\mathbf{t}}$. The cycle-consistency loss L_{Cycle} is defined as:

$$\begin{aligned} L_{Cycle}(G, F) = & \mathbb{E}_{\tilde{\mathbf{s}} \sim P_{data}(\tilde{\mathcal{S}})}[\|F(G(\tilde{\mathbf{s}})) - \tilde{\mathbf{s}}\|_1] \\ & + \mathbb{E}_{\tilde{\mathbf{t}} \sim P_{data}(\tilde{\mathcal{T}})}[\|G(F(\tilde{\mathbf{t}})) - \tilde{\mathbf{t}}\|_1] \end{aligned} \tag{6}$$

Using ℓ_1 loss above gives near identical performance to an alternative negative log likelihood definition, so is used in our experiments.

As shown in Fig. 6, with the cycle consistency loss, given an input shape $\tilde{\mathbf{s}}$ in the latent space, the recovered shape $\hat{\mathbf{s}}$ after applying both mappings G and F is visually identical to $\tilde{\mathbf{s}}$ after applying the decoder. This demonstrates the effectiveness of the cycle consistency loss.

3.5 Visual Similarity Loss

One aim of deformation transfer is to ensure visual similarity between the source and target shapes. In [Zhu et al. 2017] the visual similarity metric between two images can be measured by the squared ℓ_2 norm in the image domain since images are naturally aligned, However, this similarity metric cannot be generalized to the 3D shape domain.

In this work, the light field distance (LFD) [Chen et al. 2003] is employed to measure the visual similarity due to its robustness and accuracy. A 3D shape is projected to multiple views and image features are calculated based on projected images. When calculating distances between two shapes, global rotation is further considered to minimize the image feature differences. LFD is known to be an effective feature for shape retrieval [Shilane et al. 2004].

Fig. 8. The architecture of the generator G, which maps source shapes $\tilde{\mathcal{S}}$ to target shapes $\tilde{\mathcal{T}}$, both in latent spaces. FC refers to fully connected layers. The reverse generator F is similarly defined.

However, LFD is not differentiable and cannot be used in the loss function. We propose to use a neural network SimNet to learn this similarity measure. Since latent space is more compact, SimNet takes two vectors $\tilde{\mathbf{s}}$ and $\tilde{\mathbf{t}}$ from the latent space of $\tilde{\mathcal{S}}$ and $\tilde{\mathcal{T}}$ respectively, and predicts the LFD between the decoded shapes. Since there is no spatial relationship in the latent space, a fully connected neural network is employed, as shown in Fig. 7. The output of the neural network is denoted as $V(\tilde{\mathbf{s}}, \tilde{\mathbf{t}}) \approx LFD(Dec_{\mathcal{S}}(\tilde{\mathbf{s}}), Dec_{\mathcal{T}}(\tilde{\mathbf{t}}))$. The network is trained by minimizing the following loss:

$$L_{SimNet}(V) = L_{Dist}(V) + \beta L_{RegSim}, \tag{7}$$

where β is the weight, L_{Dist} is the average of the absolute difference $\left|V(\tilde{\mathbf{s}}, \tilde{\mathbf{t}}) - LFD(Dec_{\mathcal{S}}(\tilde{\mathbf{s}}), Dec_{\mathcal{T}}(\tilde{\mathbf{t}}))\right|$, and L_{RegSim} is the squared ℓ_2 norm regularization of network parameters to avoid overfitting. Our architecture is not restricted to the light field distance, which could be replaced with other advanced shape features.

Using the visual similarity measure, the loss is defined as follows:

$$\begin{aligned} L_{Sim}(G, F) = & \mathbb{E}_{\tilde{\mathbf{s}} \sim P_{data}(\tilde{\mathcal{S}})}[V(\tilde{\mathbf{s}}, G(\tilde{\mathbf{s}}))] \\ & + \mathbb{E}_{\tilde{\mathbf{t}} \sim P_{data}(\tilde{\mathcal{T}})}[V(F(\tilde{\mathbf{t}}), \tilde{\mathbf{t}})]. \end{aligned} \tag{8}$$

3.6 Overall Loss Function for Cycle GAN

The overall loss for the CycleGAN is:

$$\begin{aligned} & L_{CycleGAN}(G, F, D_{\mathcal{S}}, D_{\mathcal{T}}) \\ & \quad = L_{Sim}(G, F) + \gamma_1 L_{Cycle}(G, F) + \gamma_2 L_{GAN}(G, F, D_{\mathcal{S}}, D_{\mathcal{T}}), \end{aligned} \tag{9}$$

which is a linear combination of visual similarity, cycle consistency and adversarial losses with γ_1 and γ_2 being relative weights.

Our CycleGAN network is optimized as follows:

$$G^*, F^* = \arg \min_{G, F} \max_{D_{\mathcal{S}}, D_{\mathcal{T}}} L_{CycleGAN}(G, F, D_{\mathcal{S}}, D_{\mathcal{T}}), \tag{10}$$

where generators G and F aim to minimize the total loss while the discriminators $D_{\mathcal{S}}$ and $D_{\mathcal{T}}$ aim to maximize the loss by identifying synthetic shapes.

4 IMPLEMENTATION

We now give details of our network architecture and training process.

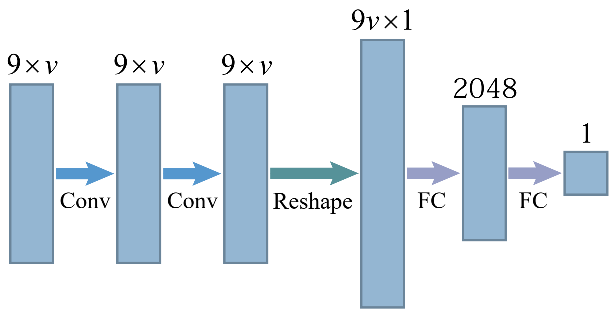


Fig. 9. The architecture of the discriminator, which takes a shape in the deformation representation as input and predicts whether it is genuine or synthesized. Conv and FC represent convolutional and fully connected layers, respectively.

4.1 Network Architecture

Our network consists of three components, convolutional VAE for encoding shapes in latent spaces, SimNet for calculating visual similarity between two shapes (from $\mathcal{S}$ and $\mathcal{T}$ respectively) both in the latent spaces, and CycleGAN for deformation transfer.

As illustrated in Fig. 4, our proposed VAE architecture uses convolutional layers for improved generalization capability. The encoder takes as input the features defined on vertices, followed by two convolutional layers with *tanh* as the activation function. In the last convolutional layer we abandon the non-linear activation function, similar to [Tan et al. 2018b]. The output of the convolutional layer is then reshaped to a vector and mapped into a 128-dimensional latent space by a fully connected layer. The decoder has a symmetric architecture, sharing the weights and biases with the encoder. We train one VAE for the source shape set $\mathcal{S}$ and one for the target shape set $\mathcal{T}$.

For SimNet, its input includes latent vectors from both domains $\tilde{\mathcal{S}}$ and $\tilde{\mathcal{T}}$. Since dimensions of the latent space do not have spatial relationship, its hidden layers are fully connected with dimensions of 2048, 1024, 512 and 256 respectively, each of them having Leaky ReLU as the activation function, as illustrated in Fig. 7.

Our CycleGAN architecture is similar to [Zhu et al. 2017] although the generators map vectors in the latent spaces. Since the latent space does not have a clear spatial relationship, the generators G and F are fully connected networks with four hidden layers of 512, 1024, 2048, 1024 dimensions respectively, mapping the latent vector of one model to another (see Fig. 8). The discriminators are defined in the feature space (i.e. after applying the decoder), and aim to classify whether the shape is genuine or synthesized. Discriminators $D_{\mathcal{S}}$ and $D_{\mathcal{T}}$ have three convolutional layers and a fully connected layer, similar to the architecture of the encoder (see Fig. 9). More detailed analysis of network architecture, parameters and input characteristics (noise, topological changes etc.) is given in the supplementary material.

4.2 Training Details

In the experiments, we fix the weight parameters $\alpha_1 = 1$, $\alpha_2 = 0.01$, $\beta = 0.01$, $\gamma_1 = 2$, $\gamma_2 = 2$. We train the whole network in three steps (VAE, SimNet and CycleGAN). The Adam solver [Kingma and Ba 2014] is used for all three training steps. Here we train each network separately (i.e. VAE and SimNet, followed by CycleGAN) rather than in an end-to-end manner. Compared with end-to-end training, training the networks separately not only saves memory during training, but also results in smaller loss, especially the SimNet loss, as shown in Table 2, which is based on transferring between a fat person (ID: 50002) and a fit person (ID: 50009) from the MPI DYNA dataset.

Table 2. Comparison of losses with different training strategies, i.e. separate training and end-to-end training.

Training	L_{VAE}	L_{SimNet}	$L_{CycleGAN}$	Total Loss
Separate	532.91	**251.04**	**41.03**	**824.98**
End-to-end	**529.60**	588.25	56.14	1174.09

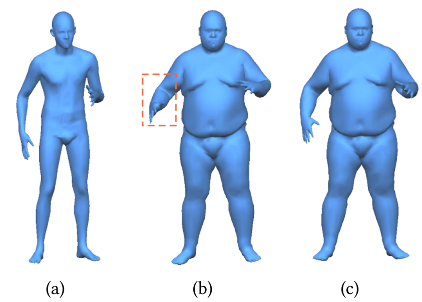


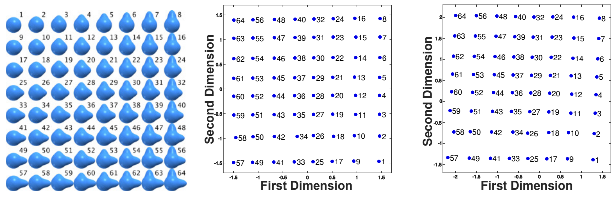
Fig. 10. Comparison of results from two different training methods. (a) source shape (b) the transfer result using end-to-end training, (c) the transfer result of our method (separate training for embedding).

The main reason is that starting training the SimNet and GAN *before* obtaining a well-trained VAE may lead to wrong optimization directions in early iterations, which ultimately results in optimization stuck at a poor local minimum. Moreover, the visual quality of the deformation transfer results is also worse, in terms of pose, than for separate training, as shown in the example in Fig. 10.

The VAE is trained with 5,000 iterations, SimNet with 12,000 iterations and CycleGAN with 7,000 iterations, by minimizing loss functions L_{VAE} (Eq. 3), L_{SimNet} (Eq. 7) and $L_{CycleGAN}$ (Eq. 9), respectively. For both VAE and CycleGAN, we set the batch size to 128 and learning rate to 0.001. For training SimNet, we set the batch size and learning rate to 512 and 0.001 respectively for the first 2,000 iterations and change them to 128 and 0.00005 for the remaining 10,000 iterators. Training batches for the VAE are randomly sampled from the training shape set. For SimNet training, we randomly select pairs of shapes from the two shape sets for training. For CycleGAN, training batches are sampled randomly from the latent space with a Gaussian distribution.

5 RESULTS AND EVALUATIONS

Our experiments were carried out on a computer with an i7 6850K CPU, 16GB RAM, and a Titan Xp GPU. The code is available at http://www.geometrylearning.com/ausdt/. The training time for each network is 30-45min for VAE (depending on the mesh size), 54min for SimNet and 65min for CycleGAN. Once the deep model is trained, transferring a source shape to a target shape is real-time,



(a) (b) DOF=2 (c) DOF=128

Fig. 11. Comparison of VAE embedding results. (a) The shape set with two deformation modes, (b) the embedding with DOFs of latent space set to 2, (c) the embedding result with DOFs of latent space set to 128; we show two dimensions with largest variations.

with feature transfer taking < 0.5ms and shape reconstruction taking 1-10ms (depending on the mesh size). We use publicly available datasets containing deformable shapes, including SCAPE [Anguelov et al. 2005], Lion, Cat, Horse, Camel, Flamingo [Sumner and Popović 2004], a fit person (ID: 50009), a fat person (ID: 50002) and two females (ID: 50020 and 50022) from MPI DYNA [Pons-Moll et al. 2015], Crane, Swing (human actions) [Vlasic et al. 2008] and Pants [White et al. 2007]. For datasets containing too few shapes (< 50), we use [Gao et al. 2016] to obtain interpolated shapes to augment the dataset. We represent shapes in the rotation-invariant representation proposed in [Gao et al. 2016]. Intuitively, blending reasonably close shapes helps ensure the interpolated shapes stay in the plausible deformation manifold, so for each mesh in the dataset, we find the nearest mesh in the feature space, and blend them to create a new shape.

5.1 Quantitative and Qualitative Evaluation of Network Components

We first analyze the effectiveness of our network components.

We compare our convolutional VAE with a state-of-the-art VAE architecture for encoding shapes. Since [Litany et al. 2017] operates on Euclidean coordinates, it does not cope well with large rotations, so we compare with [Tan et al. 2018a] which uses a rotation-invariant representation. Our use of convolutional layers better exploits spatial relationships. For each of the three datasets Camel, Horse and SCAPE, we randomly select half the shapes in the dataset for training and the remaining half for testing (as unseen data), and work out the average reconstruction error for the test dataset. We perform such tests 10 times and report the average errors in Table 3. As can be seen, our method has significantly lower errors than [Tan et al. 2018a], which shows that our convolutional VAE has better generalizability.

In our experiments, we set the dimension of the latent space for the VAE to 128, which is sufficient for all the examples in the paper. We now evaluate the approach for cases with known, lower intrinsic dimensions. An example is shown in Fig. 11. Given deformed balls from [Rustamov et al. 2013] with 2 deformation modes (a), our method gives similar results with the latent space dimension set to 2 (minimum) or 128 (our default). To visualize this, we show the embeddings of the model set in the latent space. Setting the dimension of the latent space to 2 gives the distribution in (b) which exactly describes the distribution of these balls. When setting the dimension of the latent space to 128 (default), except for two dimensions of the network which are activated, the variances of other dimensions are nearly zero. We show the distributions in the two dominant dimensions in (c).

Our VAE allows new shapes not in the training set to be generated, enriching the synthesized deformation results. To demonstrate this, we generate new shapes by randomly sampling in the latent space. As shown in Fig. 12, the models in the rows labeled "Our Result" are generated by the decoder of the VAE with random samples in the latent space. The models underneath labeled "Nearest" are the closest models (in terms of Euclidean distance) from the training set. The latent space is effective to describe the intrinsic shape set distributions and such capabilities play a central role in automatic transfer. Although linear combinations of input shapes could also be used to generate new shapes, those shapes are usually implausible or distorted due to the lack of a compact, meaningful space.

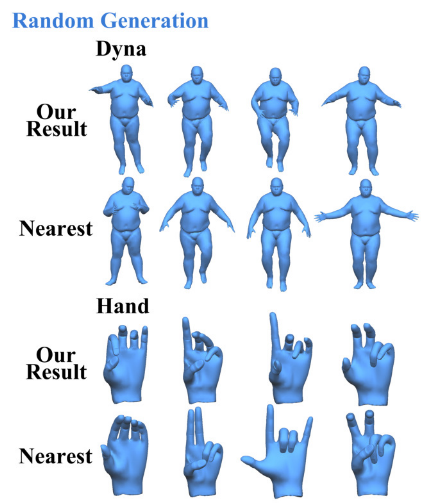


Fig. 12. The models in the row with label "Our Result" are generated by the decoder of the VAE with random samples in the latent space. The models underneath with label "Nearest" are the models with nearest mean Euclidean distance from the model in the same column.

We propose SimNet to approximate the light field distance between a pair of shapes, which ensures efficient evaluation and differentiability. To demonstrate its convergence, we take the Horse and Camel shape sets, randomly choose 75% shape pairs for training and the remaining shapes for testing, and in Fig. 13 plot errors over training iterations on both the training and test sets. It clearly shows that our training process is stable, and the learned model has good generalizability as it also works well for the test set containing unseen shapes. The average relative errors on the training and test sets are 7.63% and 8.02% respectively, showing its capability to efficiently calculate shape similarity.

To demonstrate the effectiveness of each loss term in our VC-GAN, we performed an experiment of transferring deformation from Horse to Camel, as shown in Fig. 14. These two datasets have pairs of corresponding shapes. Although our method does not exploit this property, it is useful to provide ground truth to make evaluation easier. To demonstrate the capability of VC-GAN, we use 75% of randomly selected shapes for training, and the test shapes used as

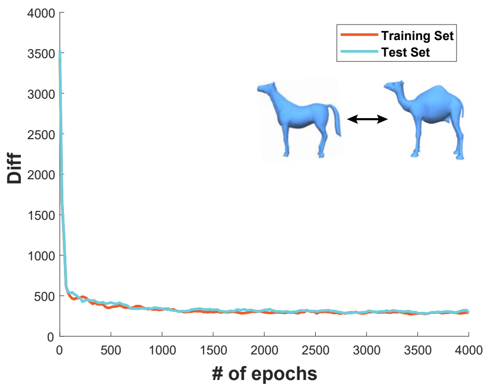


Fig. 13. Errors of SimNet over training iterations for both the training and test sets.

Table 3. Comparison of average reconstruction errors for unseen models between our convolutional VAE and [Tan et al. 2018a].

Dataset	[Tan et al. 2018a]	Our Method
Camel	0.0241	0.0167
Horse	0.0603	0.0117
SCAPE	0.1104	0.0486

Table 4. Comparison of average per-vertex errors between generated shapes and ground truth, corresponding to Figs. 14 (b-f). For comparative results (b-e), a specific loss is modified while keeping other losses unchanged. (f) shows our results.

Example	Fig. 14b	Fig. 14c	Fig. 14d	Fig. 14e	Fig. 14f
Camel to Horse	0.0738	0.0717	0.0551	0.1005	0.0082
Horse to Camel	0.0641	0.0673	0.0588	0.0769	0.0054

input (a) do not appear in the training sets. It can be seen that our results are visually very close to the ground truth. The comparative results are obtained by changing one loss term of our VC-GAN while keeping the remaining terms unchanged. Applying discriminators performing in the latent space does not work effectively due to the Gaussian distribution in the latent space, resulting in shapes that are visually different from those in the shape space (b). Results obtained without the cycle consistency loss (c), the adversarial loss (d) or the visual similarity loss (e) do not have sufficient constraints to properly transfer deformations. Table 4 gives quantitative evaluation of average per-vertex errors between generated shapes and ground truth, with shapes scaled to fit in a unit cube. Our results have much lower error than alternative results with one loss term changed, showing that each loss term is essential. The sensitivity analysis on vertex number, noise level, topology and triangulation of input meshes is documented in the supplementary material.

5.2 Deformation Transfer Results and Comparisons

We now demonstrate our system using various deformation transfer examples. Note that our method is the only method that is fully automatic with unpaired shape sets. In comparison, the method [Sumner and Popović 2004] requires point-wise correspondence to be established, which typically needs 70-80 correspondences to be manually specified. The method [Ben-Chen et al. 2009] requires building cages, and further specifying 20-40 pairs of corresponding points. The method [Baran et al. 2009] requires shapes in the source and target datasets have one-to-one correspondences, which is not normally satisfied for independent shape sets. Even if this is possible, it involves user effort to choose semantically related shape pairs. For all the methods compared, when the input is a deformation sequence, we feed in each shape and obtain the output shape to form a transferred deformation sequence. All these methods work well with this simple strategy; see the accompanying video.

Figure 15 shows comparative results of deformation transfer from Flamingo to Crane (human action) datasets. In this example, the source and target shapes differ substantially, so methods that require point-wise correspondences not only need a large number of correspondences, but can also suffer from large, local shape deformations to transform from one shape to the other. The results [Ben-Chen et al. 2009; Sumner and Popović 2004] have clear distortions on the left leg. Moreover, since the Flamingo shape does not have arms, it is not possible to control the arm deformation, even if the target shape set has a suitable arm movement that accompanies a corresponding leg movement. The result of [Baran et al. 2009] does not have artifacts. However, despite carefully choosing 7 pairs of shapes that are semantically similar and have broad coverage of poses, the deformation result is still somewhat dissimilar (e.g. legs) to the input. Our method is not only fully automatic, but also follows the deformation of the source shape, taking into account both shape similarity and plausible deformation of the target shapes.

We compare our method with existing methods requiring manual correspondence specification. The deformation transfer results are shown in Fig. 17. The result of our method (d) is fully automatic and artifact free. Compared with other methods, the VAE component in our network can better fit the data distribution and mitigate self-intersection. The results (b) and (c) are obtained using manually labeled correspondences (15 pairs and 40 pairs respectively) along with the method [Sumner and Popović 2004]. The labeled correspondences are visualized in Fig. 17. It can be seen that (c) is better than (b), with less severe artifacts (see the closeups), however it is more time-consuming to label. It takes a skilled user 7 minutes 36 seconds and 18 minutes 13 seconds to label 15 pairs and 40 pairs of correspondences, respectively. Our fully automatic method produces a better result.

Figure 18 shows deformation transfer from the Lion to Cat datasets. Our automatic method produces similar and sometimes better results than [Sumner and Popović 2004] which requires manually specifying correspondences. Differences between results are highlighted in Fig. 19 using color coding to show the vertex displacements. It clearly highlights the front right leg, where our result looks closer to the source shape.

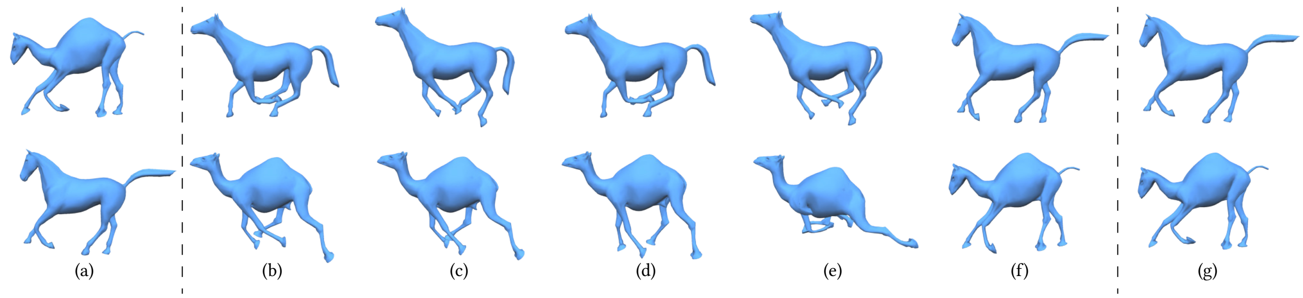


Fig. 14. The evaluation of each loss term for our VC-GAN model for deformation transfer. (a) input shapes, (b) results with discriminators performed in the latent space, (c) results without cycle consistency loss, (d) results without adversarial loss, (e) results without visual similarity loss, (f) our results, (g) ground truth.

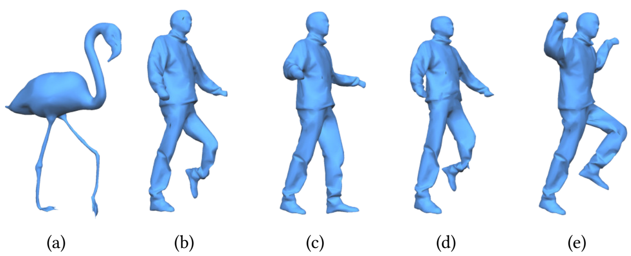


Fig. 15. Comparison of different methods for deformation transfer from Flamingo to Crane (human action) datasets. (a) source shape, (b) deformation transfer result of [Sumner and Popović 2004], (c) deformation transfer result of [Baran et al. 2009], (d) deformation transfer result of [Ben-Chen et al. 2009], (e) our result.

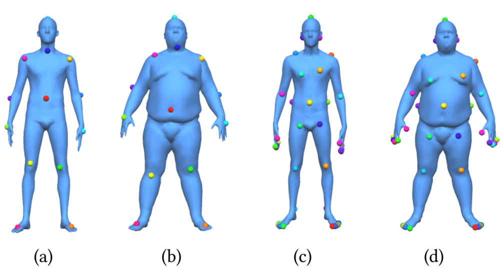


Fig. 16. Two pairs of shapes with manually labeled landmarks. (a)(b) 15 pairs of manually labeled landmarks between the source and target shapes (c)(d) 40 pairs of correspondence points.

In Fig. 20, we also compare our results with alternative methods without correspondences. We would like to stress that neither of these methods are intended for mesh deformation transfer. The results in the second row are obtained using [Rustamov et al. 2013] where each shape is mapped to the nearest shape from the other set in the joint embedding space. The method produces some plausible results, but can also match a shape to an incorrect one (e.g. first column). It also cannot be used to generate deformation sequences since it does not generate new shapes. The results in the third row are generated using a baseline method by representing shapes in a rotation-invariant representation [Gao et al. 2016], applying PCA (Principal Component Analysis) to extract main deformation modes independently for the source and target shape sets and transferring PCA weights from the source shape to the target shape. A similar idea has been used for face deformation transfer [Cosker et al. 2008]. However, for general shapes, it is challenging to identify meaningful basis deformations that can be used to reliably establish the mapping. These are in fact expected due to the nature of the shape sets themselves. We aim to produce plausible shapes sampled from the target space that are visually similar to the source shape. While we wish the deformed target to resemble the source, it should also be consistent with the target samples to ensure it is semantically plausible for the target shape. There is a tradeoff between these two factors, which explains the discrepancies. For example, a person naturally walks in a different way than a Flamingo, and therefore when transferring the walking between them, the poses may look different, but the results are indeed plausible. Figures 21 and 22 show results for our automatic deformation transfer of a galloping elephant to a horse and a collapsing horse to a camel. Our synthesized shapes are visually similar to the target shape space while faithfully following the input deformation. We also evaluate the robustness of our method on the size of the training data. We independently remove 30% of randomly chosen shapes from the input datasets of the collapsing horse and camel, and the deformation transfer results on the reduced shape sets are similar to the transfer results with the whole datasets as shown in Fig. 22. Figure 23 shows an example of transferring hand deformation to a pair of pants, with finger movement effectively transferred to “walking” pants.

Figure 1 shows an example of transferring deformation from a fit person (ID: 50009) to a fat person (ID: 50002) (both from the MPI DYNA dataset). The training datasets contain thousands of shapes with different poses, and our method successfully generates suitable shapes in the shape space, effectively transferring rich actions (punching, running, etc.), despite substantial differences in their body shapes.

Figure 24 shows an example of transferring the deformation from a running person to a robot. The mesh dataset with various poses of the robot is from [Xu et al. 2009]. Our method produces plausible robot deformation following the human action. In this example, the robot mesh is composed of multiple components and our method cannot be directly applied. To cope with this, the first mesh in the robot dataset is converted to a singly connected mesh using a volumetric mesh repair approach [Ju 2004]. The correspondences between the singly connected mesh and the original multi-component mesh

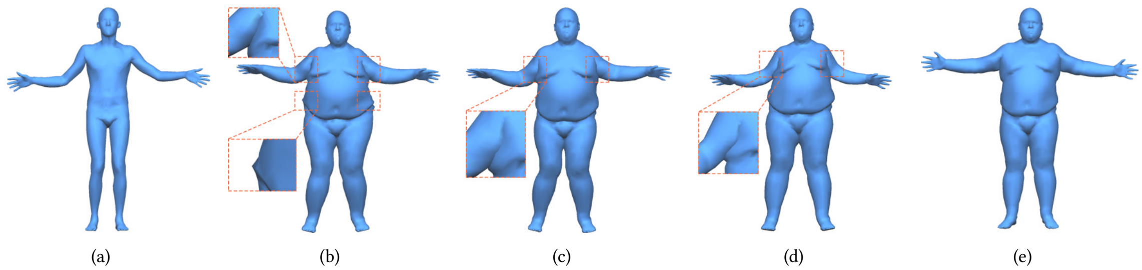


Fig. 17. Deformation transfer results. (a) source shape, (b)(c) deformation transfer results by [Sumner and Popović 2004] with 15 labeled landmarks and 40 labeled landmarks respectively (as shown in Fig. 16), (d) deformation transfer result of [Ben-Chen et al. 2009] with 40 labeled landmarks (as shown in Fig. 16), (e) our automatic deformation transfer result.

are then easily established based on nearest points. Using these correspondences as soft constraints, the singly connected mesh can be deformed to approximate the dataset with multi-component meshes. Our automatic unpaired deformation transfer technique is applied to the human and the singly connected robot mesh datasets. Once the deformed robot is obtained, we obtain the rigid rotation and translation of each component based on the vertex correspondences to obtain the deformed multi-component robot shape.

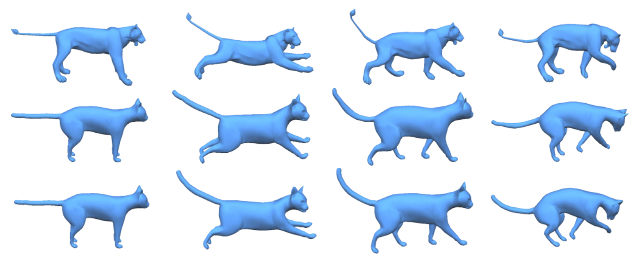


Fig. 18. Deformation transfer from Lion to Cat. First row: source shapes, second row: results of [Sumner and Popović 2004], third row: our results.

5.3 Extension of Our Method to Semantic Transfer

Our proposed deformation transfer method based on visual similarity is fully automatic. However, sometimes shapes which are semantically related may not be visually similar. To address this, we present a simple extension of our method that takes semantically related pairs as in [Baran et al. 2009] in addition to two unpaired shape sets to perform semantic deformation transfer. Inspired by the Triplet Network [Hoffer and Ailon 2014], we modify the SimNet to two independent fully connected networks, each embedding the source or target latent space to a lower dimensional Euclidean space, where the distance can represent the semantic differences. An example is shown in Fig. 25 where the deformation is transferred from a face to a running female [Pons-Moll et al. 2015]. We expect the semantic pairs in Fig. 25 to express that closing an eye corresponds to lifting a leg. Similar to [Baran et al. 2009], we select 19 pairs of faces and female shapes that are semantically similar as input to train the semantic similarity network. To train it, we use a simple strategy such that the network aims to predict distance 0 for the 19 given pairs of shapes, and the maximum LFD for all the other shape pairs. We compare our method with another method [Baran et al. 2009], which also requires corresponding shape pairs. Results in Fig. 26 show that our method generates more semantically similar shapes than [Baran et al. 2009]. This is because generally two datasets may not have pairs with perfect one-to-one semantic correspondence, where shape pairs selected by a human will inevitably introduce conflicts and confuse the computation of shape bases [Baran et al. 2009]. In such cases, our method can exploit unpaired shapes in the 3D model datasets with VAE to characterize shape distributions in the datasets. In Fig. 26, motions in the second row produced by [Baran et al. 2009] tend to freeze and the legs do not match the eyes. In contrast, our results in the third row have higher semantic similarity.

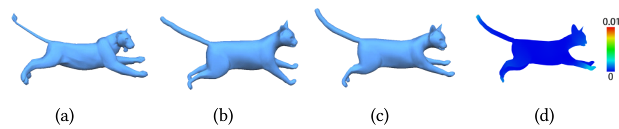


Fig. 19. Comparison of deformation transfer results from Lion to Cat. (a) source shape, (b) deformation transfer result of [Sumner and Popović 2004], (c) deformation transfer result of our method, (d) color-coding of the Euclidean distance of each vertex between the results of our method and [Sumner and Popović 2004].

6 CONCLUSIONS AND FUTURE WORK

In this paper, we propose a novel architecture for automatic mesh deformation transfer, which is also flexible, allowing source and target shape sets to be unpaired. Our method achieves state-of-the-art deformation transfer results. However, the method may still be improved. The current visual similarity measure works well when the deformation is visually similar. It may not work as well for semantically similar but visually very different shapes. In the future, we would like to investigate using neural networks to estimate more advanced visual similarity measures. Also, our VC-GAN in the latent space has the potential to be useful in other 3D shape synthesis applications, which we would like to explore in the near future.

In cases where the two shape sets have significant visual differences such as a horse and a human (ID: 50009), it is challenging to construct a reliable visual similarity metric between them. An example is shown in Fig. 27. Although both the horse and the person are running, the corresponding body parts between the horse and the person do not move simultaneously. The semantic transfer techniques in Sec. 5.3 could be applied to transfer the deformation from the horse to the human in a semantic manner, although automatic methods not requiring user effort would still be preferred. The main problems of the current visual similarity metric are that the light field distance may not be able to measure the differences

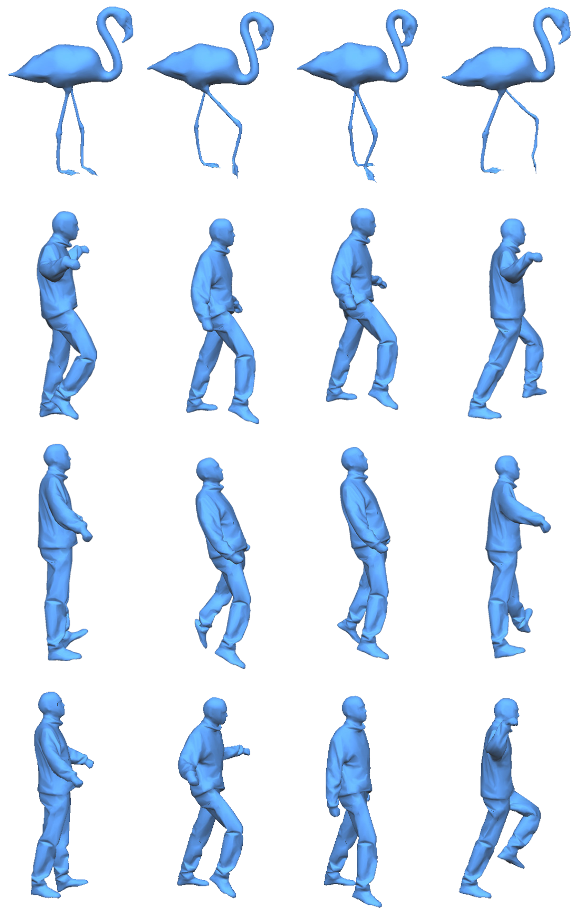


Fig. 20. Deformation transfer from Flamingo to Crane (human action) datasets. First row: source flamingo shapes, second row: nearest shapes from the embedding of [Rustamov et al. 2013], third row: shapes obtained by transferring PCA weights [Cosker et al. 2008], fourth row: our deformation transfer results.

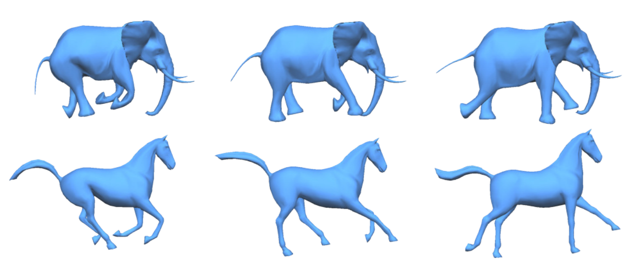


Fig. 21. Deformation transfer from elephant to horse. First row: source elephant shapes, second row: our results of deformation transfer to the horse shape.

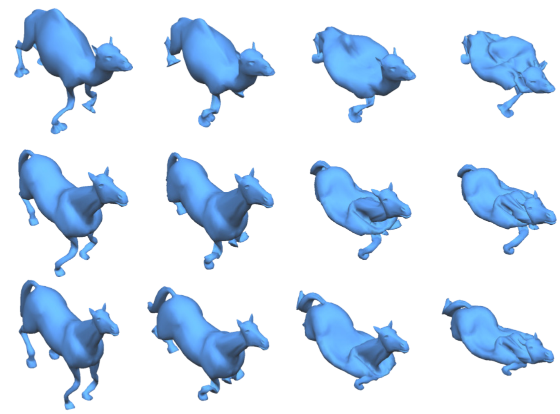


Fig. 22. Deformation transfer from camel to horse with collapse effect. First row: source camel shapes, second row: our results of deformation transfer to the horse shape, third row: the results obtained by independently removing 30% of randomly chosen shapes in the datasets, and the results are very similar to those in the second row, demonstrating that our method does not require shape sets to have shapes with corresponding poses.

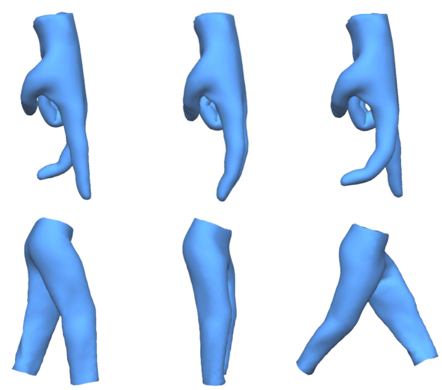


Fig. 23. Deformation transfer from Hand to Pants datasets. First row: source hand shapes, second row: our results of deformation transfer to pants.

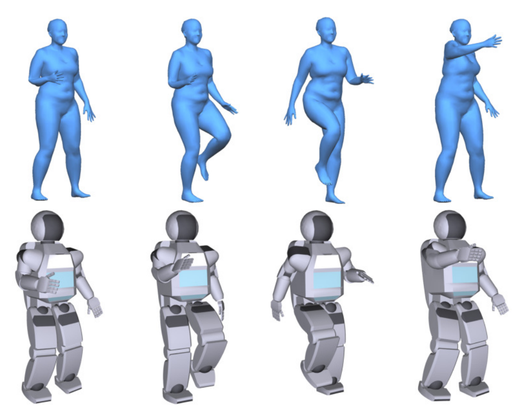


Fig. 24. Deformation transfer from a person to a robot. First row: source person shapes, second row: the transferred robot shapes.

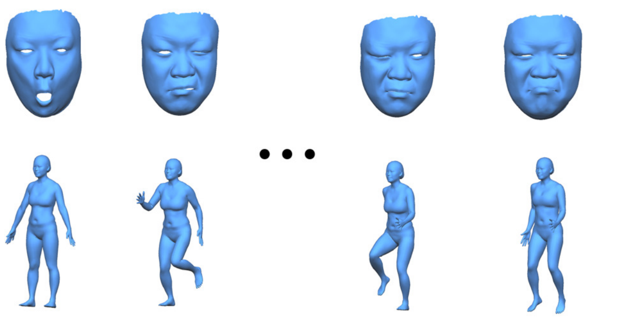


Fig. 25. Semantically related pairs labeled manually. First row: source face shapes, second row: semantically related target female shapes. Closing an eye corresponds to lifting a leg.

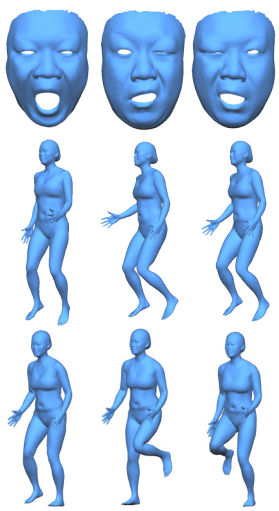


Fig. 26. Semantic transfer from a face to a running female. First row: source face shapes, second row: semantic deformation transfer [Baran et al. 2009], third row: our results of deformation transfer to a person.

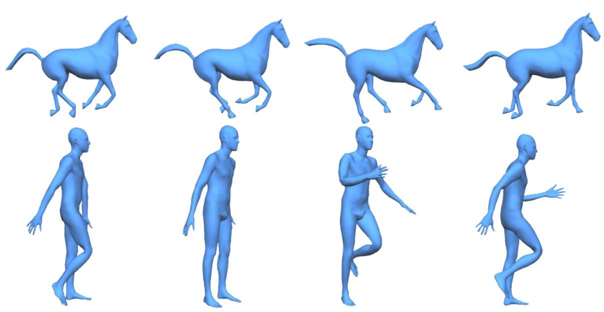


Fig. 27. Deformation transfer from a horse to a person (ID: 50009) from the MPI DYNA dataset. First row: source horse shapes, second row: our results of deformation transfer to a person.

in a semantic manner and it is not easy to distinguish small visual differences.

We also perform another experiment where all the pose shapes of two persons (ID: 50020 and 50026) in the DYNA dataset are used as input. As shown in Fig. 28, in general our method reasonably transfers deformation of the first person to the second. However, there are some visual differences between two shapes due to the limitations of the visual similarity metric. In future work, we will develop more powerful visual similarity metrics to better distinguish subtle visual differences and better capture semantic similarities.

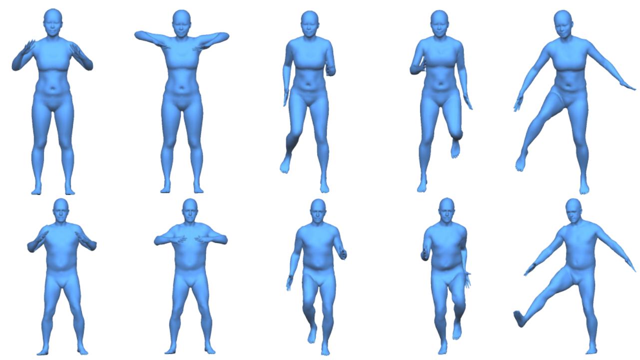


Fig. 28. Deformation transfer from a female to a male by using all the pose shapes in the MPI DYNA dataset. First row: source female shapes (ID: 50020), second row: our results of deformation transfer to a male (ID: 50026).

ACKNOWLEDGMENTS

We thank the reviewers for their constructive comments and suggestions. We would also like to thank Prof. Mirela Ben-Chen for providing the code of [Ben-Chen et al. 2009], Dr. Bob Sumner for providing the executable of [Sumner and Popović 2004] and the mesh models, Dr. Raif Rustamov for providing the deformed ball shapes of [Rustamov et al. 2013]. This work was supported by National Natural Science Foundation of China (No. 61872440, No. 61828204, No. 61502453 and No. 61772499), Young Elite Scientists Sponsorship Program by CAST (No. 2017QNRC001), Royal Society-Newton Mobility Grant (No. IE150731), CCF-Tencent Open Fund and NVIDIA Corporation with the GPU donation. Weiwei Xu was partially supported by National Natural Science Foundation of China (No. 61732016), Alibaba IDEA Lab and fundamental research fund for the central universities (2017YFB1002600).

REFERENCES

Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. 2005. SCAPE: shape completion and animation of people. *ACM Trans. Graph.* 24, 3 (2005), 408–416.

Ilya Baran, Daniel Vlasic, Eitan Grinspun, and Jovan Popović. 2009. Semantic deformation transfer. *ACM Trans. Graph.* 28, 3 (2009), 36:1–36:6.

Mirela Ben-Chen, Ofir Weber, and Craig Gotsman. 2009. Spatial deformation transfer. In *Symposium on Computer Animation*. 67–74.

Davide Boscaini, Jonathan Masci, Emanuele Rodolà, and Michael Bronstein. 2016a. Learning shape correspondence with anisotropic convolutional neural networks. In *Advances in Neural Information Processing Systems*. 3189–3197.

Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Michael M Bronstein, and Daniel Cremers. 2016b. Anisotropic diffusion descriptors. *Comp. Graph. Forum* 35, 2 (2016), 431–441.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv:1312.6203* (2013).

Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. 2003. On Visual Similarity Based 3D Model Retrieval. *Computer Graphics Forum* 22, 3 (2003), 223–232.

Lu Chen, Jin Huang, Hanqiu Sun, and Hujun Bao. 2010. Cage-based deformation transfer. *Computers & Graphics* 34, 2 (2010), 107–118.

Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 2016. 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction. In *ECCV*. 628–644.

Hung-Kuo Chu and Chao-Hung Lin. 2010. Example-based Deformation Transfer for 3D Polygon Models. *J. Inf. Sci. Eng.* 26, 2 (2010), 379–391.

Etienne Corman, Justin Solomon, Mirela Ben-Chen, Leonidas Guibas, and Maks Ovsjanikov. 2017. Functional Characterization of Intrinsic and Extrinsic Geometry. *ACM Trans. Graph.* 36, 2 (2017), 14:1–14:17.

D. Cosker, R. Borkett, D. Marshall, and P. L. Rosin. 2008. Towards automatic performance-driven animation between multiple types of facial model. *IET Computer Vision* 2, 3 (2008), 129–141.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*. 3844–3852.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems*. 2224–2232.

Haoqiang Fan, Hao Su, and Leonidas Guibas. 2017. A point set generation network for 3D object reconstruction from a single image. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

James E. Gain and Dominique Bechmann. 2008. A survey of spatial deformation from a user-centered perspective. *ACM Trans. Graph.* 27, 4 (2008), 107:1–107:21.

Lin Gao, Yu-Kun Lai, Jie Yang, Ling-Xiao Zhang, Leif Kobbelt, and Shihong Xia. 2017. Sparse Data Driven Mesh Deformation. *arXiv:1709.01250* (2017).

Lin Gao, Yu-Kun Lai, Dun Liang, Shu-Yu Chen, and Shihong Xia. 2016. Efficient and Flexible Deformation Representation for Data-Driven Surface Modeling. *ACM Trans. Graph.* 35, 5 (2016), 158:1–158:17.

Rohit Girdhar, David F. Fouhey, Mikel Rodriguez, and Abhinav Gupta. 2016. Learning a Predictable and Generative Vector Representation for Objects. In *ECCV*.

Xiaoguang Han, Chang Gao, and Yizhou Yu. 2017. DeepSketch2Face: A Deep Learning Based Sketching System for 3D Face and Caricature Modeling. *ACM Trans. Graph.* 36, 4 (2017), 126:1–126:12.

Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep convolutional networks on graph-structured data. *arXiv:1506.05163* (2015).

Elad Hoffer and Nir Ailon. 2014. Deep Metric Learning Using Triplet Network. In *International Workshop on Similarity-Based Pattern Recognition*. 84–92.

Haibin Huang, Evangelos Kalegorakis, Siddhartha Chaudhuri, Duygu Ceylan, Vladimir G. Kim, and Ersin Yumer. 2018. Learning Local Shape Descriptors with View-based Convolutional Neural Networks. *ACM Trans. Graph.* 37, 1 (2018), 6:1–14.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. 2017. Image-to-Image Translation with Conditional Adversarial Networks. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Tao Ju. 2004. Robust Repair of Polygonal Models. *ACM Trans. Graph.* 23, 3 (2004), 888–895.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980* (2014).

Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. 2017. GRASS: Generative Recursive Autoencoders for Shape Structures. *ACM Trans. Graph.* 36, 4 (2017), 52:1–52:14.

Yangyan Li, Hao Su, Charles Ruizhongtai Qi, Noa Fish, Daniel Cohen-Or, and Leonidas J Guibas. 2015. Joint embeddings of shapes and images via CNN image purification. *ACM Trans. Graph.* 34, 6 (2015), 234.

Or Litany, Alex Bronstein, Michael Bronstein, and Ameesh Makadia. 2017. Deformable Shape Completion with Graph Convolutional Autoencoders. *arXiv:1712.00268* (2017).

Jerry Liu, Fisher Yu, and Thomas A. Funkhouser. 2017. Interactive 3D Modeling with a Generative Adversarial Network. *arXiv:1706.05170* (2017).

Haggai Maron, Meirav Galun, Noam Aigerman, Miri Trope, Nadav Dym, Ersin Yumer, Vladimir G. Kim, and Yaron Lipman. 2017. Convolutional Neural Networks on Surfaces via Seamless Toric Covers. *ACM Trans. Graph.* 36, 4 (2017), 71:1–71:10.

Daniel Maturana and Sebastian Scherer. 2015. Voxnet: a 3D convolutional neural network for real-time object recognition. In *IEEE Conference on Intelligent Robots and Systems*. 922–928.

Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H Barr. 2003. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and mathematics III*. 35–57.

Charlie Nash and Chris KI Williams. 2017. The shape variational autoencoder: A deep generative model of part-segmented 3D objects. *Comp. Graph. Forum* 36, 5 (2017), 1–12.

Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning convolutional neural networks for graphs. In *ICML*. 2014–2023.

Gerard Pons-Moll, Javier Romero, Naureen Mahmood, and Michael J Black. 2015. Dyna: A model of dynamic human shape in motion. *ACM Trans. Graph.* 34, 4 (2015), 120.

Raif M Rustamov, Maks Ovsjanikov, Omri Azencot, Mirela Ben-Chen, Frédéric Chazal, and Leonidas Guibas. 2013. Map-based exploration of intrinsic shape differences and variability. *ACM Trans. Graph.* 32, 4 (2013), 72:1–72:12.

Abhishek Sharma, Oliver Grau, and Mario Fritz. 2016. Vconv-dae: Deep volumetric shape learning without object labels. In *ECCV Workshops*. 236–250.

Baoguang Shi, Song Bai, Zhichao Zhou, and Xiang Bai. 2015. Deeppano: Deep panoramic representation for 3-d shape recognition. *IEEE Signal Processing Letters* 22, 12 (2015), 2339–2343.

Philip Shilane, Patrick Min, Michael Kazhdan, and Thomas Funkhouser. 2004. The Princeton Shape Benchmark. In *Shape Modeling International*.

Ayan Sinha, Asim Unmesh, Qixing Huang, and Karthik Ramani. 2017. SurfNet: Generating 3D Shape Surfaces Using Deep Residual Networks. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Robert W Sumner and Jovan Popović. 2004. Deformation transfer for triangle meshes. *ACM Trans. Graph.* 23, 3 (2004), 399–405.

Minhyuk Sung, Hao Su, Vladimir G. Kim, Siddhartha Chaudhuri, and Leonidas Guibas. 2017. Complementme: Weakly-supervised Component Suggestions for 3D Modeling. *ACM Trans. Graph.* 36, 6 (2017), 226:1–226:12.

Qingyang Tan, Lin Gao, Yu-Kun Lai, and Shihong Xia. 2018a. Variational Autoencoders for Deforming 3D Mesh Models. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Qingyang Tan, Lin Gao, Yu-Kun Lai, Jie Yang, and Shihong Xia. 2018b. Mesh-based Autoencoders for Localized Deformation Component Analysis. In *AAAI*.

Shubham Tulsiani, Hao Su, Leonidas J Guibas, Alexei A Efros, and Jitendra Malik. 2017. Learning Shape Abstractions by Assembling Volumetric Primitives. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popović. 2008. Articulated mesh animation from multi-view silhouettes. *ACM Trans. Graph.* 27, 3 (2008), 1–9.

Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. 2017a. O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis. *ACM Trans. Graph.* 36, 4 (2017), 72:1–72:11.

Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. 2017b. High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Ryan White, Keenan Crane, and David Forsyth. 2007. Capturing and Animating Occluded Cloth. *ACM Trans. Graph.* 26, 3, Article 34 (2007).

Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T Freeman, and Joshua B Tenenbaum. 2016. Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In *Advances in Neural Information Processing Systems*. 82–90.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. 3D ShapeNets: A deep representation for volumetric shapes. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 1912–1920.

Weiwei Xu, Jun Wang, KangKang Yin, Kun Zhou, Michiel van de Panne, Falai Chen, and Baining Guo. 2009. Joint-aware manipulation of deformable models. *ACM Trans. Graph.* 28, 3 (2009), 35:1–35:9.

Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. 2016. Perspective Transformer Nets: Learning Single-View 3D Object Reconstruction without 3D Supervision. In *Advances in Neural Information Processing Systems*. 1696–1704.

Jie Yang, Lin Gao, Yu-Kun Lai, Paul L. Rosin, and Shihong Xia. 2018. Biharmonic deformation transfer with automatic key point selection. *Graphical Models* 98 (2018), 1–13.

Li Yi, Hao Su, Xingwen Guo, and Leonidas J. Guibas. 2017a. SyncSpecCNN: Synchronized Spectral CNN for 3D Shape Segmentation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Zili Yi, Hao (Richard) Zhang, Ping Tan, and Minglun Gong. 2017b. DualGAN: Unsupervised Dual Learning for Image-to-Image Translation.. In *ICCV*. 2868–2876.

Kun Zhou, Weiwei Xu, Yiying Tong, and Mathieu Desbrun. 2010. Deformation Transfer to Multi-Component Objects. *Computer Graphics Forum* 29, 2 (2010), 319–325.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In *IEEE International Conference on Computer Vision (ICCV)*.