# Neural Cages for Detail-Preserving 3D Deformations

Wang Yifan[1] Noam Aigerman[2] Vladimir G. Kim[2] Siddhartha Chaudhuri [2,3] Olga Sorkine-Hornung[1]

[1]ETH Zurich [2]Adobe Research [3]IIT Bombay

## Abstract

*We propose a novel learnable representation for detail-preserving shape deformation. The goal of our method is to warp a source shape to match the general structure of a target shape, while preserving the surface details of the source. Our method extends a traditional cage-based deformation technique, where the source shape is enclosed by a coarse control mesh termed* cage*, and translations prescribed on the cage vertices are interpolated to any point on the source mesh via special weight functions. The use of this sparse cage scaffolding enables preserving surface details regardless of the shape's intricacy and topology. Our key contribution is a novel neural network architecture for predicting deformations by controlling the cage. We incorporate a differentiable cage-based deformation module in our architecture, and train our network end-to-end. Our method can be trained with common collections of 3D models in an unsupervised fashion, without any cage-specific annotations. We demonstrate the utility of our method for synthesizing shape variations and deformation transfer.*

## 1. Introduction

Deformation of 3D shapes is a ubiquitous task, arising in many vision and graphics applications. For instance, *deformation transfer* [25] aims to infer a deformation from a given pair of shapes and apply the same deformation to a novel target shape. As another example, a small dataset of shapes from a given category (*e.g.*, chairs) can be augmented by *synthesizing variations*, where each variation deforms a randomly chosen shape to the proportions and morphology of another while preserving local detail [29, 32].

Deformation techniques usually need to simultaneously optimize at least two competing objectives. The first is alignment with the target, *e.g.*, matching limb positions while deforming a human shape to another human in a different pose. The second objective is adhering to quality metrics, such as distortion minimization and preservation of local geometric features, such as the human's face. These two objectives are contradictory, since a perfect alignment of a deformed source shape to the target precludes preserving the original details of the source.

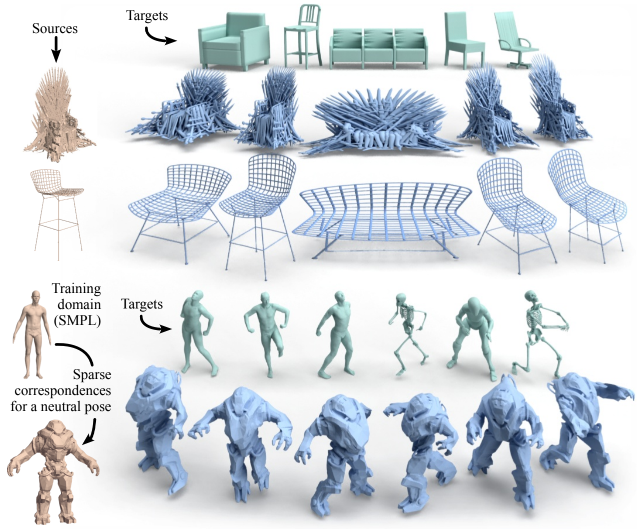


Figure 1: Applications of our neural cage-based deformation method. *Top:* Complex source chairs (brown) deformed (blue) to match target chairs (green), while accurately preserving detail and style with non-homogeneous changes that adapt different regions differently. No correspondences are used at any stage. *Bottom:* A cage-based deformation network trained on many posed humans (SMPL) can transfer various poses of novel targets (SCAPE, skeleton, X-Bot, in green) to a very dissimilar robot of which only a single neutral pose is available. A few matching landmarks between the robot and a neutral SMPL human are required. Dense correspondences between SMPL humans are used only during training.

Due to these conflicting objectives, optimization techniques [17] require parameter tuning to balance the two competing terms, and are heavily reliant on an inferred or manually supplied correspondence between the source and the target. These parameters vary based on the shape category, representation, and the level of dissimilarity between the source and the target.

To address these limitations, recent techniques train a *neural network* to predict shape deformations. This is achieved by predicting new positions for all vertices of a template shape [26] or by implicitly representing the deformation as a mapping of all points in 3D, which is then used to map each vertex of a source shape [6, 29]. Examples of the results of some of these methods can be seen in Fig 4, which demonstrates the limitations of such approaches: the predicted deformations corrupt features and exhibit distortion, especially in areas with thin structures, fine details or gross discrepancies between source and target. These artifacts are due to the inherent limitations of neural networks to capture, preserve, and generate high frequencies.

In this paper, we circumvent the above issues via a classic geometry processing technique called *cage-based deformation* [14, 15, 18], abbreviated to *CBD*. In CBD, the source shape is enclosed in a very coarse scaffold mesh called the *cage* (Fig 2). The deformation of the cage is transferred to the enclosed shape by interpolating the translations of the cage vertices. Fittingly, the interpolation schemes in these classic works are carefully designed to preserve details and minimize distortion.

Our main technical contribution is a novel neural architecture in which, given a source mesh, learnable parameters are optimized to predict both the positioning of the cage around the source shape, as well as the deformation of that cage, which drives the deformation of the enclosed shape in order to match a target shape. The source shape is deformed by deterministically interpolating the new positions of its surface points from those of the cage vertices, via a novel, differentiable, cage-based deformation layer. The pipeline is trained end-to-end on a collection of randomly chosen pairs of shapes from a training set.

The *first key advantage* of our method is that cages provide a much more natural space for predicting deformations: CBD is feature-preserving by construction, the degrees of freedom in deformation only depends on the number of vertices on the coarse cage. In short, our network makes a prediction in a low-dimensional space of highly regular deformations.

The *second key advantage* is that our method is not tied to a single source shape, nor to a single mesh topology. As the many examples in this paper demonstrate, the trained network can predict and deform cages for similar shapes not observed during training. The target shape can be crude and noisy, *e.g.*, a shape acquired with cheap scanning hardware or reconstructed from an image. Furthermore, dense correspondences between the source and target shapes are not required in general, though they can help when the training set has very varied articulations. Thus the method can be trained on large datasets that are not co-registered and do not have consistently labeled landmarks.

We show the utility of our method in two main applications. We generate shape variations by deforming a 3D model using other shapes as well as images as targets. We also use our method to pose a human according to a target humanoid character, and, given a few sparse correspondences, perform deformation transfer and pose an arbitrary novel humanoid. See Figures 1, 7, 9 and 4 for examples.

## 2. Related work

We now review prior work on learning deformations, traditional methods for shape deformation, and applications.

**Learning 3D deformations.** Many recent works in learning 3D geometry have focused on generative tasks, such as synthesis [8, 20] and editing [36] of unstructured geometric data. These tasks are especially challenging if one desires high-fidelity content with intricate details. A common approach to producing intricate shapes is to deform an existing generic [28] or category-specific [7] template. Early approaches represented deformations as a single vector of vertex positions of a template [26], which limited their output to shapes constructable by deforming the specific template, and also made the architecture sensitive to the template tessellation. An alternative is to predict a freeform deformation field over 3D voxels [9, 13, 34]; however, this makes the deformation's resolution dependent on the voxel resolution, and thus has limited capability to adapt to a specific shape categories and source shapes.

Alternatively, some architectures learn to map a single point at a time, conditioned on some global descriptor of the target shape [7]. These architectures can also work for novel sources by conditioning the deformation field on features of both source and target [6, 29]. Unfortunately, due to network capacity limits, these techniques struggle to represent intricate details and tend to blur high-frequency features.

**Traditional methods for mesh deformation.** Research on detail-preserving deformations in the geometry processing community spans several decades and has contributed various formulations and optimization techniques [24]. These methods usually rely on a sparse set of control points whose transformations are interpolated to all remaining points of the shape; the challenge lies in defining this interpolation in a way that preserves details. This can be achieved by solving an optimization problem to reduce the distortion of the deformation such as [23]. However, defining the output deformation as the solution to an intricate non-convex optimization problem significantly limits the ability of a network to learn this deformation space.

Instead, we use cage-based deformations as our representation, where the source shape is enclosed by a coarse *cage* mesh, and all surface points are written as linear combinations of the cage vertices, i.e., generalized barycentric coordinates. Many designs have been proposed for these coordinate functions such that shape structure and details are preserved under interpolations [2, 14, 15, 18, 21, 27, 31].

**Shape synthesis and deformation transfer.** Automatically aligning a source shape to a target shape while preserving details is a common task, used to synthesize variations of shapes for amplification of stock datasets [11] or for transferring a given deformation to a new model, targeting animation synthesis [25]. To infer the deformation, correspondence between the two shapes needs to be accounted for, either by explicitly inferring corresponding points [12, 16, 17], or by implicitly conditioning the deformation fields on the latent code of the target shape [6, 9, 29]. Our work builds

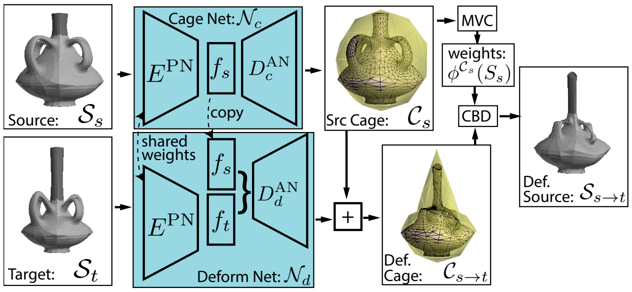


Figure 2: Overview. A source $\mathcal{S}_s$ and a target $\mathcal{S}_t$ are encoded by the same PointNet encoder $E^{\text{PN}}$ into latent codes $f_s$ and $f_t$, resp. An AtlasNet-style decoder $D_c^{\text{AN}}$ decodes $f_s$ to a source cage $\mathcal{C}_s$ in the cage module $\mathcal{N}_c$. Another decoder $D_d^{\text{AN}}$ creates the offset for $\mathcal{C}_s$ in the deformation module $\mathcal{N}_d$ from the concatenation of $f_s$ and $f_t$, yielding a deformed cage $\mathcal{C}_{s\to t}$. Given a source cage and shape, our novel MVC layer computes the mean value coordinates $\phi^{\mathcal{C}_s}(\mathcal{S}_s)$, which are used to produce a deformed source shape $\mathcal{S}_{s\to t}$ from the cage deformation $\mathcal{C}_{s\to t}$.

upon the latter learning-based framework, but uses cages to parameterize the space of deformations.

Gao et al. [4] automate the deformation transfer for unpaired shapes using cycled generative adversarial networks, thus the trained method cannot be easily adapted for new shape targets. Some prior techniques focus on transferring and interpolating attributes between various latent spaces trained for shape generation [5, 33]. These generative models are not capable of fully preserving local geometric features, especially if the source is not pre-segmented into simpler primitives (as assumed by [5]). In general, such methods are only expected to perform well if the input shapes are relatively similar to those observed at training time.

## 3. Method

We now detail our approach for learning cage-based deformations (CBD). We start with a brief overview of the principles of CBD, and then explain how we train a network to control these deformations from data. The implementation is available at https://github.com/yifita/deep_cage.

### 3.1. Cage-based deformations

CBD are a type of freeform space deformations. Instead of defining a deformation solely on the surface $\mathcal{S}$, space deformations warp the entire ambient space in which the shape $\mathcal{S}$ is embedded. In particular, a CBD controls this warping via a coarse triangle mesh, called a *cage* $\mathcal{C}$, which typically encloses $\mathcal{S}$. Given the cage, any point in ambient space $\mathbf{p} \in \mathbb{R}^3$ is encoded via generalized barycentric coordinates, as a weighted average of the cage vertices $\mathbf{v}_j$: $\mathbf{p} = \sum \phi_j^{\mathcal{C}}(\mathbf{p})\, \mathbf{v}_j$, where the weight functions $\{\phi_j^{\mathcal{C}}\}$ depend on the relative position of $\mathbf{p}$ w.r.t. to the cage vertices $\{\mathbf{v}_j\}$. The deformation of any point in ambient space is obtained by simply offsetting the cage vertices and interpolating their new positions $\mathbf{v}'_j$ with the pre-computed weights, *i.e.*

$$\mathbf{p}' = \sum_{0\le j<|V_{\mathcal{C}}|} \phi_j^{\mathcal{C}}(\mathbf{p})\, \mathbf{v}'_j. \tag{1}$$

Previous works on CBD constructed various formulae to attain weight functions $\{\phi_j^{\mathcal{C}}\}$ with specific properties, such as interpolation, linear precision, smoothness and distortion minimization. We choose mean value coordinates (MVC) [15] for their feature preservation and interpolation properties, as well as simplicity and differentiability w.r.t. the source and deformed cages' coordinates.

### 3.2. Learning cage-based deformation

As our goal is an end-to-end pipeline for deforming shapes, we train the network to predict both the source cage and the target cage, in order to optimize the quality of the resulting deformation. Given a source shape $\mathcal{S}_s$ and a target shape $\mathcal{S}_t$, we design a deep neural network that predicts a cage deformation that warps $\mathcal{S}_s$ to $\mathcal{S}_t$ while preserving the details of $\mathcal{S}_s$. Our network is composed of two branches, as illustrated in Fig 2: a cage-prediction model $\mathcal{N}_c$, which predicts the initial cage $\mathcal{C}_s$ around $\mathcal{S}_s$, and a deformation-prediction model $\mathcal{N}_d$, which predicts an offset from $\mathcal{C}_s$, yielding the deformed cage $\mathcal{C}_{s\to t}$, *i.e.*

$$\mathcal{C}_s = \mathcal{N}_c(\mathcal{S}_s) + \mathcal{C}_0, \qquad \mathcal{C}_{s\to t} = \mathcal{N}_d(\mathcal{S}_t, \mathcal{S}_s) + \mathcal{C}_s \tag{2}$$

Since both branches are differentiable, they can be both learned jointly in an end-to-end manner.

The branches $\mathcal{N}_c$ and $\mathcal{N}_d$ only predict the cage and do not directly rely on the detailed geometric features of the input shapes. Hence, our network does not require high-resolution input nor involved tuning for the network architectures. In fact, both $\mathcal{N}_c$ and $\mathcal{N}_d$ follow a very streamlined design: their encoders and decoders are simplified versions of the ones used in AtlasNet [8]. We remove the batch normalization and reduce the channel sizes, and instead of feeding 2D surface patches to the decoders, we feed a template cage $\mathcal{C}_0$ and the predicted initial cage $\mathcal{C}_s$ to the the cage predictor and deformer respectively, and let them predict the offsets. By default, $\mathcal{C}_0$ is a 42-vertex sphere.

### 3.3. Loss terms

Our loss incorporates three main terms. The first term optimizes the source cage to encourage positive mean value coordinates. The two latter terms optimize the deformation, the first by measuring alignment to target and the second by measuring shape preservation. Together, these terms comprise our basic loss function:

$$\mathcal{L} = \alpha_{\text{MVC}}\mathcal{L}_{\text{MVC}} + \mathcal{L}_{\text{align}} + \alpha_{\text{shape}}\mathcal{L}_{\text{shape}}. \tag{3}$$

We use $\alpha_{\text{MVC}} = 1$, $\alpha_{\text{shape}} = 0.1$ in all experiments.

To optimize the mean value coordinates of the source cage, we penalize negative weight values, which emerge when the source cage is highly concave, self-overlapping, or when some of the shape's points lie outside the cage:

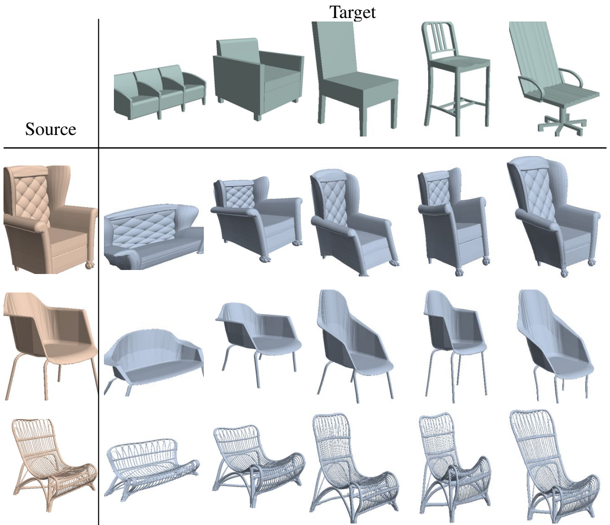

Figure 3: Synthesizing variations of source shapes (brown), by deforming them to match targets (green).

$$\mathcal{L}_{\mathrm{MVC}} = \frac{1}{|\mathcal{C}_s||\mathcal{S}_s|} \sum_{i=1}^{|\mathcal{S}_s|} \sum_{j=1}^{|\mathcal{C}_s|} \left|\min\left(\phi_{ji}, 0\right)\right|^2, \tag{4}$$

where $\alpha_{\mathrm{MVC}}$ is the loss weight, and $\phi_{ji}$ denotes the coordinates of $\mathbf{p}_i \in \mathcal{S}_s$ w.r.t. $\mathbf{v}_j \in \mathcal{C}_s$.

$\mathcal{L}_{\text{align}}$ is measured either via chamfer distance in the unsupervised case sans correspondences, or as the L2 distance when supervised with correspondences.

The above two losses drive the deformation towards alignment with the target, but this may come at the price of preferring alignment over feature preservation. Therefore, we add terms that encourage shape preservation. Namely, we draw inspiration from Laplacian regularizers [7,19,29], but propose to use a point-to-surface distance as an orientation-invariant, second-order geometric feature. Specifically, for each point $\mathbf{p}$ on the source shape, we fit a PCA plane to a local neighborhood $\mathcal{B}$ (we use the one-ring of the mesh), and then compute the point-to-plane distance as $d = \|\mathbf{n}^T\left(\mathbf{p} - \mathbf{p}_{\mathcal{B}}\right)\|$, where $\mathbf{n}$ denotes the normal of the PCA plane and $\mathbf{p}_{\mathcal{B}} = \frac{1}{|\mathcal{B}|}\sum_{\mathbf{q}\in\mathcal{B}(\mathbf{p})} \mathbf{q}$ is the centroid of the local neighborhood around $\mathbf{p}$. We then penalize change in the distance $d_i$ for each vertex on the surface:

$$\mathcal{L}_{\text{p2f}} = \frac{1}{|\mathcal{S}_s|} \sum_{i=1}^{|\mathcal{S}_s|} \|d_i - d_i'\|^2 \tag{5}$$

where $d_i'$ is the distance post deformation. In contrast to the uniform Laplacian, which considers the distance to the centroid and hence yields a non-zero value whenever the local neighborhood is not evenly distributed, the proposed point-to-surface distance better describes the local geometric features.

For man-made shapes, we use two additional losses that leverage priors of this shape class. First, normal consistency is important for, *e.g*., preserving the planarity of elements like tabletops. To encourage this, we penalize the angular difference of PCA normals before and after deformation:

$$\mathcal{L}_{\text{normal}} = \frac{1}{|\mathcal{S}_s|} \sum_{i}^{|\mathcal{S}_s|} (1 - \mathbf{n}_i^T \mathbf{n}_i'), \tag{6}$$

where $\mathbf{n}'$ denotes the PCA-normal after the deformation. As demonstrated later, this normal penalty considerably improves the perceptual quality of the deformation. Second, similarly to Wang *et al*. [29], we also use the symmetry loss $\mathcal{L}_{\text{symm}}$, measured as the chamfer distance between the shape and its reflection around the $x = 0$ plane. We apply this loss to the deformed shape $\mathcal{S}_{s\to t}$ as well as the cage $\mathcal{C}_s$. Thus, our final shape preservation loss is: $\mathcal{L}_{\text{shape}} = \mathcal{L}_{\text{p2f}} + \mathcal{L}_{\text{normal}} + \mathcal{L}_{\text{symm}}$ for man-made shapes and $\mathcal{L}_{\text{shape}} = \mathcal{L}_{\text{p2f}}$ for characters.

# 4. Applications

We now showcase two applications of the trained cage-based deformation network.

## 4.1. Stock amplification via deformation

Creating high-quality 3D assets requires significant time, technical expertise, and artistic talent. Once the asset is created, the artist commonly deforms the model to create several variations of it. Inspired by prior techniques on automatic stock amplification [29], we use our method to learn a meaningful deformation space over a collection of shapes within the same category, and then use random pairs of source and target shapes to synthesize plausible variations of artist-generated assets.

**Training details.** We train our model on the *chair, car* and *table* categories from ShapeNet [3] using the same splitting into training and testing sets as in Grouiex *et al*. [6]. We then randomly sample 100 pairs from the test set. Each shape is normalized to fit in a unit bounding box and is represented by 1024 points.

**Variation synthesis examples.** Fig 3 shows variations generated from various source-target pairs, exhibiting the regularizing power of the cages: even though our training omits all semantic supervision such as part labels, these variations are plausible and do not exhibit feature distortions; fine details, such as chair slats, are preserved.

**Comparisons.** We compared our target-driven deformation method to other methods that strive to achieve the same goal. Results are shown in Fig 4. While in many cases alternative techniques do align the deformed shape the target, in all cases they introduce significant artifacts in the deformed meshes.

We first compare to a non-learning-based approach: non-rigid ICP [10], a classic registration technique that alternates between correspondence estimation and optimization of a non-rigid deformation to best align corresponding points. We show results with the optimal registration parameters we found to achieve detail preservation. Clearly,

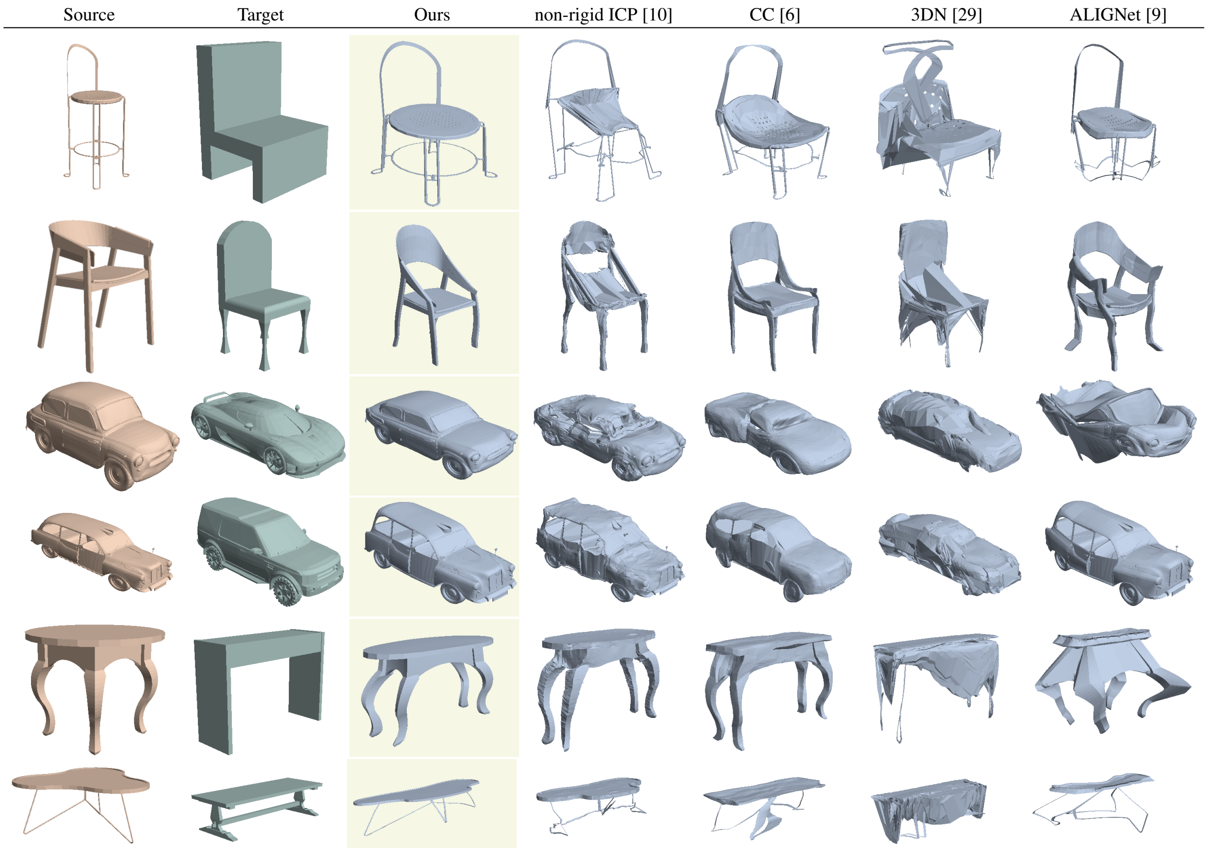


Figure 4: Comparison of our method with other non-homogeneous deformation methods. Our method achieves superior detail preservation of the source shape in comparison to optimization-based [10] and learning-based [6, 9, 29] techniques, while still aligning the output to the target.

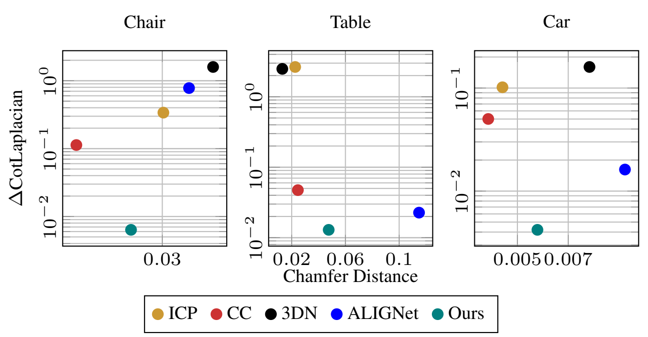


Figure 5: Quantitative evaluation of our method vs alternative methods. Each point represents a method, embedded according to its average alignment error (Chamfer Distance) and distortion (ΔCotLaplacian). Points near the bottom-left corners are better.

ICP is sensitive to wrong correspondences that cause convergence to artifact-ridden local minima. We also compare to learning-based methods that directly predict per-point transformations and leverage cycle-consistency (CC) [6] or feature-preserving regularization (3DN) [29] to learn low-distortion shape deformations. Both methods blur and omit features, while also creating artifacts by stretching small parts. We also compare to ALIGNet [9], a method that predicts a freeform deformation over a voxel grid, yielding a volumetric deformation of the ambient space similarly to our technique. Contrary to our method, the coarse voxel grid cannot capture the fine deformation of the surface needed to avoid large artifacts. Our training setup is identical to CC, and we retrained 3DN and ALIGNet with the same setup using parameters suggested by the authors.

In Fig 6 we compare our results to the simplest of deformation methods – anisotropic scaling, achieved by simply rescaling the source bounding box to match that of the target. While local structure is well preserved, this method cannot account for the different proportion changes required for different regions, highlighting the necessary intricacy of the optimal deformation in this case.

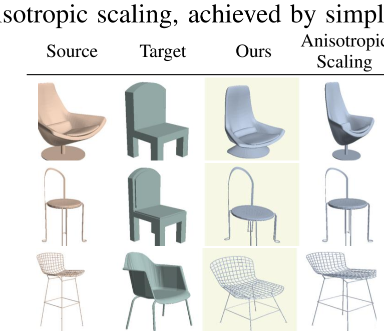


Figure 6: Comparison of our method with anisotropic scaling. Our method better matches corresponding semantic parts.

**Quantitative comparisons.** in Fig 5, we quantitatively evaluate the various methods using two metrics: distance to the target shape, and detail preservation, measured via

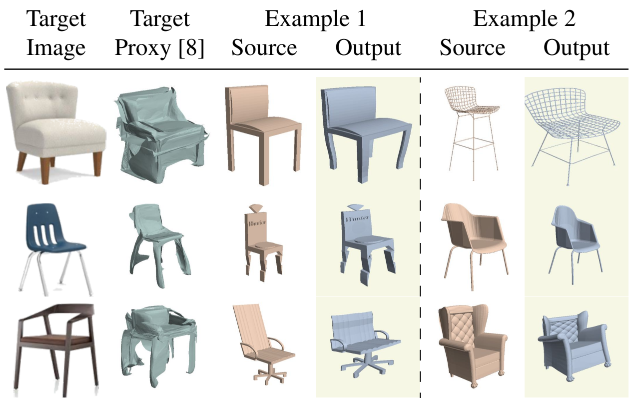

Figure 7: We use our method to deform a 3D shape to match a real 2D image. We first use AtlasNet [8] to reconstruct a 3D proxy target. Despite the poor quality of the proxy, it still serves as a valid target for our network to generate a matching output preserving the fine details of the source.

chamfer distance (computed over a dense set of 5000 uniformly sampled points) and difference in cotangent Laplacians, respectively. Note that these metrics do not favor any method, since all optimize for a variant of chamfer distance, and none of the methods optimize for the difference in the cotangent Laplacian. Each 2D point in the figure represents one method, with the point's coordinates prescribed with respect to the two metrics, the origin being ideal. This figure confirms our qualitative observations: our method is more effective at shape preservation than most alternatives while still capturing the gross structure of the target.

**Using images as targets.** Often, a 3D target is not readily available. Images are more abundant and much easier to acquire, and thus pose an appealing alternative. We use a learning-based single-view reconstruction technique to create a proxy target to use with our method to find appropriate deformation parameters. We use publicly available product images of real objects and execute AtlasNet's SVR reconstruction [8] to generate a coarse 3D proxy as a target. Fig 7 shows that even though the proxy has coarse geometry and many artifacts, these issues do not affect the deformation, and the result is still a valid variation of the source.

## 4.2. Deformation transfer

Given a novel 3D model, it is much more time-efficient to automatically deform it to mimic an existing example deformation, than having an artist deform the novel model directly. This automatic task is called *deformation transfer*. The example deformation is given via a model in a rest pose $\mathcal{S}_s$, and a model in the deformed pose $\mathcal{S}_t$. The novel 3D model is given in a corresponding rest post $\mathcal{S}_{s'}$. The goal is to deform the novel model to a position $\mathcal{S}_{t'}$ so that the deformation $\mathcal{S}_{s'} \to \mathcal{S}_{t'}$ is analogous to $\mathcal{S}_s \to \mathcal{S}_t$. This task can be quite challenging, as the example deformation $\mathcal{S}_t$ may have very different geometry, or even come from an ad-hoc scan, and thus dense correspondences between $\mathcal{S}_s$ and $\mathcal{S}_t$ are unavailable, preventing the use of traditional

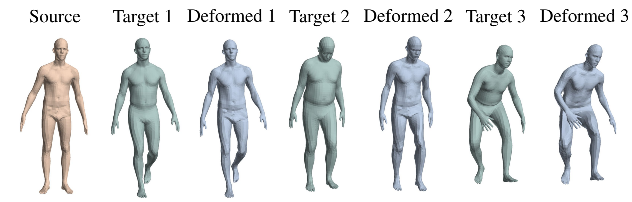

Figure 8: The deformation model, trained to deform a fixed source (left) to various articulations.

mesh optimization techniques such as [25]. Furthermore, as the novel character $\mathcal{S}_{s'}$ may be significantly different from all models observed during training, it is impossible to a-priori learn a deformation subspace for $\mathcal{S}_{s'}$ unless sufficient pose variations of $\mathcal{S}_{s'}$ is available, as in Gao *et al.* [4].

We demonstrate that our learning-based approach can be used to perform deformation transfer on arbitrary humanoid models. The network infers the deformation from the source $\mathcal{S}_s$ to the target $\mathcal{S}_t$, without any given correspondences, and then an optimization-based method transfers this deformation to a novel shape $\mathcal{S}_{s'}$ to obtain the desired deformation $\mathcal{S}_{t'}$. Hence, given *any* arbitrarily-complex novel character, all our method requires are sparse correspondences supplying the necessary alignment between the two rest poses, $\mathcal{S}_s$ and $\mathcal{S}_{s'}$. We now overview the details of our learned cage-based human deformation model and the optimization technique used to transfer the deformations.

**Learning cage-based human deformation.** To train our human-specific deformation model, we use the dataset [7] generated using the SMPL model [1] of 230K models of various humans in various poses. Since our application assumes that the exemplar deformation is produced from a single canonical character, we picked one human in the dataset to serve as $\mathcal{S}_s$. Subsequently, since we only have one static source shape $\mathcal{S}_s$, we use a *static* cage $\mathcal{C}_s$ manually created with 77 vertices, and hence do not need the cage prediction network $\mathcal{N}_c$ and only use the deformation network $\mathcal{N}_d$. We train $\mathcal{N}_d$ to deform the static $\mathcal{S}_s$ using the static $\mathcal{C}_s$ into exemplars $\mathcal{S}_t$ from the dataset (with targets not necessarily stemming from the same humanoid model as $\mathcal{S}_s$). We then train with the loss in (3), but with one modification: in similar fashion to prior work, during training we use ground truth correspondences and hence replace the chamfer distance with the L2 distance w.r.t the known correspondences. Note that these correspondences are *not* used at inference time.

Lastly, during training we also optimize the static source cage $\mathcal{C}_c$ by treating its vertices as degrees of freedom and directly optimizing them to reduce the loss so as to attain a more optimal, but still static cage after training.

Fig 8 shows examples of human-specific cage deformations predicted for test targets (not observed while training). Note how our model successfully matches poses even without knowing correspondences at inference time, while preserving fine geometric details such as faces and fingers.

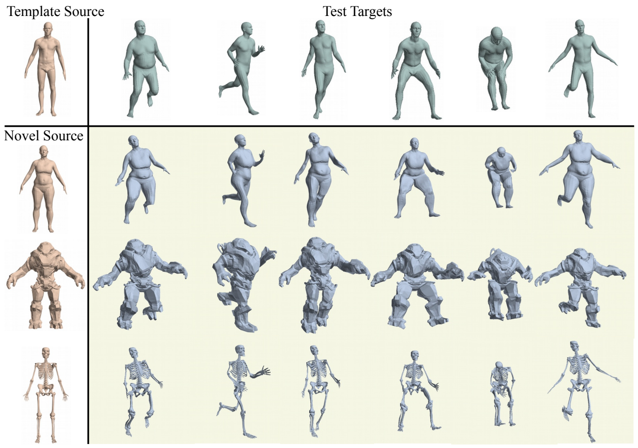


Figure 9: Deformation transfer. We first learn the cage deformation space for a template source shape (top left) with known pose and body shape variations. Then, we annotate predefined landmarks on new characters in neutral poses (left column, rows 2-4). At test time, given novel target poses (top row, green) without known correspondences to the template, we transfer their poses to the other characters (blue).

**Transferring cage deformations.** After training, we have at our disposal the deformation network $\mathcal{N}_d$ and the static $\mathcal{C}_s, \mathcal{S}_s$. We assume to be given a novel character $\mathcal{S}_{s'}$ with 83 landmark correspondences aligning it to $\mathcal{S}_s$, and an example target pose $\mathcal{S}_t$. Our goal is to deform $\mathcal{S}_{s'}$ into a new pose $\mathcal{S}_{t'}$ that is analogous to the deformation of $\mathcal{S}_s$ into $\mathcal{S}_t$.

We first generate a new cage $\mathcal{C}_{s'}$ for the character $\mathcal{S}_{s'}$. Instead of a network-based prediction, we simply optimize the static cage $\mathcal{C}_s$, trying to match mean value coordinates between corresponding points of $\mathcal{S}_s, \mathcal{S}_{s'}$:

$$\mathcal{L}_{\text{consistency}} = \sum_j \sum_{(p,q)} \|\phi_j^{\mathcal{C}_s}(p) - \phi_j^{\mathcal{C}'_s}(q)\|^2 \qquad (7)$$

where $(p, q)$ are corresponding landmarks. We also regularize with respect to the cotangent Laplacian of the cage:

$$\mathcal{L}_{\mathcal{C}\text{lap}} = \sum_{0 \le j < |\mathcal{C}_s|} \left(\|L_{\text{cot}}\mathbf{v}_j\| - \|L_{\text{cot}}\mathbf{v}'_j\|\right)^2. \qquad (8)$$

Then, we compute $\mathcal{C}_{s'}$ by minimizing $\mathcal{L} = \mathcal{L}_{\text{consistency}} + 0.05\mathcal{L}_{\mathcal{C}\text{lap}}$, with $\mathcal{C}_s$ used as initialization, solved via the Adam optimizer with step size $5 \cdot 10^{-4}$ and up to $10^4$ iterations (or until $\mathcal{L}_{\text{consistency}} < 10^{-5}$).

Finally, given the cage $\mathcal{C}_{s'}$ for the novel character, we compute the deformed cage $\mathcal{C}_{s' \to t'}$, using our trained deformation network, by applying the predicted offset to the optimized cage: $\mathcal{C}_{s' \to t'} = \mathcal{N}_d(\mathcal{S}_t, \mathcal{S}_{s'}) + \mathcal{C}_{s'}$. The final deformed shape $\mathcal{S}_{t'}$ is computed by deforming $\mathcal{S}_{s'}$ using the cage $\mathcal{C}_{s' \to t'}$ via (1). This procedure is illustrated in Fig 10, while more examples can be found in the supplemental material. Due to the agnostic nature of cage-deformations to the underlying shape, we are able to seamlessly combine machine learning and traditional geometry processing to generalize to never-observed characters. To demonstrate the expressiveness of our method, we show examples on extremely dissimilar target characters in Figures 1 and 9.

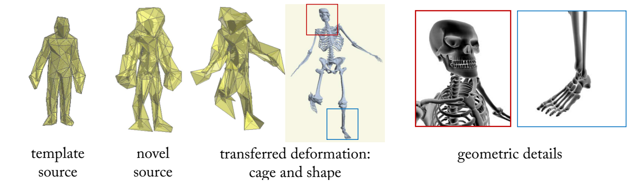


Figure 10: In deformation transfer, the manually created cage for a template shape (leftmost) is fitted to a novel source shape (second left) by optimizing MVC of a sparse set of aligned landmarks. The learnt deformation can be directly applied to the fitted source cage (columns 3-4), preserving rich geometric features (right).

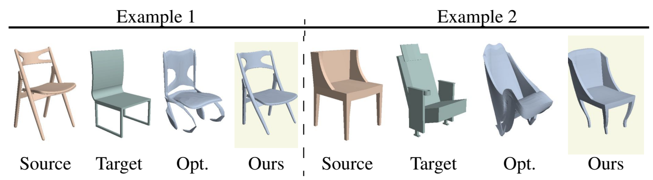


Figure 11: Our approach produces more plausible inter-shape correspondences and deformations than per-pair optimization.

## 5. Evaluation

In this section, we study the effects and necessity of the most relevant components of our methods. To measure the matching error we use chamfer distance computed on 5000 uniformly resampled points, and to measure the feature distortion we use the distance between cotangent Laplacians. All models are normalized to a unit bounding box.

**Benefit of learning CBD from data.** Instead of learning the CBD from a collection of data, one could minimize (3) for a single pair of shapes, which is essentially a non-rigid Iterative-Closest-Point (ICP) parameterised by cage vertices. As shown in Fig 11, when correct correspondence estimation becomes challenging, the optimization alternative produces non-plausible outputs. In contrast, the learnt approach utilizes domain knowledge embedded in the network's parameters [22, 35], amounting to better reasoning about the plausibility of inter-shape correspondences and deformations. The learned domain knowledge can generalize to new data. As demonstrated in Sec 4.2, even though our network is trained with ground-truth correspondences, it is able to automatically associate the source shape to a new target *without correspondences* during inference, while optimization methods require accurate correspondence estimation for every new target.

**Effect of the negative MVC penalty, $\mathcal{L}_{\text{MVC}}$.** in Fig 12 we show the effect of penalizing negative mean value coordinates. We train our architecture on 300 vase shapes from COSEG [30], while varying the weight $\alpha_{\text{MVC}} \in \{0, 1, 10\}$. Increasing this term brings the cages closer to the shapes' convex hulls, leading to more conservative deformations. Quantitative results in Table 1a also suggest that increasing the weight $\alpha_{\text{MVC}}$ favors shape preservation over alignment accuracy. Completely eliminating this term hurts convergence, and increases the alignment error further.

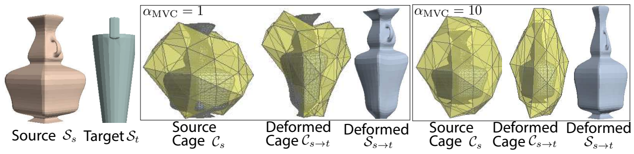


Figure 12: Effect of $\mathcal{L}_{\text{MVC}}$. Higher regularization yields more conservative deformations.

| Ablation | CD | $\Delta$CotLaplacian |
|---|---|---|
| $\alpha_{\text{MVC}} = 0$ | 1.64 | 9.04 |
| $\alpha_{\text{MVC}} = 1$ | 1.44 | 8.74 |
| $\alpha_{\text{MVC}} = 10$ | 2.65 | 8.27 |
| (a) Effect of the MVC loss, $\mathcal{L}_{\text{MVC}}$ | | |
| $\mathcal{L}_{\text{shape}} = \mathcal{L}_{\text{lap}} + \mathcal{L}_{\text{symm}}$ | 5.16 | 4.75 |
| $\mathcal{L}_{\text{shape}} = \mathcal{L}_{\text{p2f}} + \mathcal{L}_{\text{symm}}$ | 4.86 | 4.70 |
| $\mathcal{L}_{\text{shape}} = \mathcal{L}_{\text{normal}} + \mathcal{L}_{\text{symm}}$ | 5.45 | 4.33 |
| (b) Effect of the shape preservation losses, $\mathcal{L}_{\text{shape}}$ | | |
| $\mathcal{N}_c$=Identity | 3.27 | 5.65 |
| $\mathcal{N}_c$=Source-invariant | 3.11 | 12.05 |
| $\mathcal{N}_c$=Ours | 3.06 | 10.45 |
| (c) Design choices for cage prediction network, $\mathcal{N}_c$ | | |

Table 1: We evaluate effect of different losses ($\mathcal{L}_{\text{MVC}}, \mathcal{L}_{\text{shape}}$) and components ($\mathcal{N}_c$) of our pipeline with respect to chamfer distance (CD, scaled by $10^2$) and cotangent Laplacian (scaled by $10^3$).

**Effect of the shape preservation losses, $\mathcal{L}_{\text{shape}}$.** In Fig 13 we compare deformations produced with the full loss ($\mathcal{L}_{\text{shape}} = \mathcal{L}_{\text{p2f}} + \mathcal{L}_{\text{normal}} + \mathcal{L}_{\text{symm}}$) to ones produced with only one of the first two loss terms. While we did not use the Laplacian regularizer $\mathcal{L}_{\text{lap}}$ as in [29], it seems to have an effect equivalent to $\mathcal{L}_{\text{p2f}}$. As expected, $\mathcal{L}_{\text{normal}}$ prevents bending of rigid shapes. We quantitatively evaluate these regularizers in Table 1b, which suggests that $\mathcal{L}_{\text{p2f}}$ is slightly better as the deformed shape is more aligned with the target than $\mathcal{L}_{\text{lap}}$, even though shape preservation has not been sacrificed. $\mathcal{L}_{\text{normal}}$ reduces distortion even further.

**Design choices for the cage prediction network, $\mathcal{N}_c$.** The cage prediction network $\mathcal{N}_c$ morphs the template cage mesh (a 42-vertex sphere) into the initial cage enveloping the source shape. In Fig 14 and Table 1c we compare to two alternative design choices for this module: an *Identity* module retains the template cage, and a *source-invariant* module in which we optimize the template cage's vertex coordinates with respect to all targets in the dataset, but then use the same fixed cage for testing. Learning source-specific cages produces deformations closest to the target with minimum detail sacrifice. As expected, fixing the template cage produces more rigid deformations, yielding lower distortion at the price of less-aligned results.

## 6. Conclusion

We show that classical cage-based deformation provides a low-dimensional, detail-preserving deformation space directly usable in a deep-learning setting. We implement cage

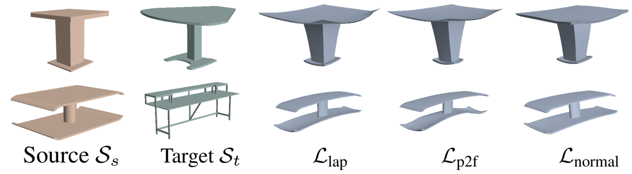


Figure 13: The effect of different shape preservation losses, note that all results include $\mathcal{L}_{\text{symm}}$.

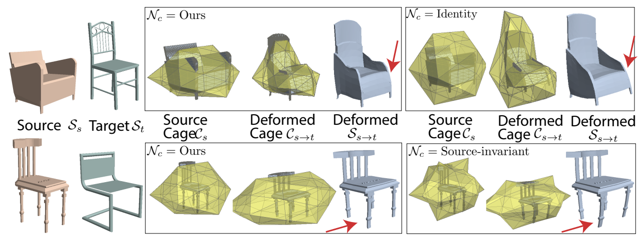


Figure 14: The effect of source-cage prediction. We compare our per-instance prediction of $\mathcal{N}_c$ with (1) a static spherical cage (top right) and (2) a single optimized cage prediction over the entire training set (bottom right). Our approach achieves better alignment with the target shape.

weight computation and cage-based deformation as differentiable network layers, which could be used in other architectures. Our method succeeds in generating feature-preserving deformations for synthesizing shape variations and deformation transfer, and better preserves salient geometric features than competing methods.

A limitation of our approach is that we focus on the deformation quality produced by the predicted cages: hence, the cage geometry itself is not designed to be comparable to professionally-created cages for 3D artists. Second, our losses are not quite sufficient to always ensure rectilinear/planar/parallel structures in man-made shapes are perfectly preserved (Fig 13). Third, for certain types of deformations other parameterizations might be a more natural choice, such as skeleton-based deformation for articulations, nonetheless the idea presented in this paper can be adopted for similarly.

Our method provides an extensible and versatile framework for data-driven generation of high detail 3D geometry. In the future we would like to incorporate alternative cage weight computation layers, such as Green Coordinates [18]. Unlike MVC, this technique is not affine-invariant, and thus would introduce less affine distortion for large articulations (see the second row fourth column in Fig 9). We also plan to use our method in other applications such as registration, part assembly, and generating animations.


## Acknowledgments

We thank Rana Hanocka and Dominic Jack for their extensive help. The robot model in Figures 1 and 9 is from SketchFab, licensed under CC-BY. This work was supported in part by gifts from Adobe, Facebook and Snap.

## References


[1] Federica Bogo, Javier Romero, Matthew Loper, and Michael J Black. FAUST: Dataset and evaluation for 3D mesh registration. In *CVPR*, pages 3794–3801, 2014.

[2] Stéphane Calderon and Tamy Boubekeur. Bounding proxies for shape approximation. *ACM Trans. Graph.*, 36(4):57, 2017.

[3] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An information-rich 3D model repository. Technical Report arXiv:1512.03012 [cs.GR], 2015.

[4] Lin Gao, Jie Yang, Yi-Ling Qiao, Yu-Kun Lai, Paul L Rosin, Weiwei Xu, and Shihong Xia. Automatic unpaired shape deformation transfer. In *SIGGRAPH Asia*, 2018.

[5] Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao(Richard) Zhang. SDM-NET: Deep generative network for structured deformable mesh. *ACM Trans. Graph.*, 38(6):243:1–243:15, 2019.

[6] Thibault Groueix, Matthew Fisher, Vladimir Kim, Bryan Russell, and Mathieu Aubry. Unsupervised cycle-consistent deformation for shape matching. In *SGP*, 2019.

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

[8] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. AtlasNet: A papier-mâché approach to learning 3D surface generation. In *CVPR*, 2018.

[9] Rana Hanocka, Noa Fish, Zhenhua Wang, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. ALIGNet: partial-shape agnostic alignment via unsupervised learning. *ACM Trans. Graph.*, 38(1):1, 2018.

[10] Haibin Huang, Evangelos Kalogerakis, Siddhartha Chaudhuri, Duygu Ceylan, Vladimir G Kim, and Ersin Yumer. Learning local shape descriptors from part correspondences with multiview convolutional networks. *ACM Trans. Graph.*, 37(1), 2018.

[11] Qixing Huang, Hai Wang, and Vladlen Koltun. Single-view reconstruction via joint analysis of image and shape collections. *ACM Trans. Graph.*, 34(4):87:1–87:10, 2015.

[12] Qi-Xing Huang, Bart Adams, Martin Wicke, and Leonidas J. Guibas. Non-rigid registration under isometric deformations. In *SGP*, 2008.

[13] Dominic Jack, Jhony K Pontes, Sridha Sridharan, Clinton Fookes, Sareh Shirazi, Frederic Maire, and Anders Eriksson. Learning free-form deformations for 3D object reconstruction. In *ACCV*, 2018.

[14] Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. Harmonic coordinates for character articulation. *ACM Trans. Graph.*, 26(3), 2007.

[15] Tao Ju, Scott Schaefer, and Joe Warren. Mean value coordinates for closed triangular meshes. *ACM Trans. Graph.*, 24(3):561–566, 2005.

[16] Hao Li, Linjie Luo, Daniel Vlasic, Pieter Peers, Jovan Popović, Mark Pauly, and Szymon Rusinkiewicz. Temporally coherent completion of dynamic shapes. *ACM Trans. Graph.*, 31(1), 2012.

[17] Hao Li, Robert W. Sumner, and Mark Pauly. Global correspondence optimization for non-rigid registration of depth scans. In *SGP*, 2008.

[18] Yaron Lipman, David Levin, and Daniel Cohen-Or. Green coordinates. *ACM Trans. Graph.*, 27(3), 2008.

[19] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft Rasterizer: A differentiable renderer for image-based 3D reasoning. In *ICCV*, 2019.

[20] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3D reconstruction in function space. In *CVPR*, 2019.

[21] Leonardo Sacht, Etienne Vouga, and Alec Jacobson. Nested cages. *ACM Trans. Graph.*, 34(6):170:1–170:14, 2015.

[22] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *ICCV*, 2019.

[23] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *SGP*, 2007.

[24] Olga Sorkine and Mario Botsch. Interactive shape modeling and deformation. In *EUROGRAPHICS Tutorials*, 2009.

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

[26] Qingyang Tan, Lin Gao, Yu-Kun Lai, and Shihong Xia. Variational autoencoders for deforming 3D mesh models. In *CVPR*, 2018.

[27] Jean-Marc Thiery, Julien Tierny, and Tamy Boubekeur. CageR: Cage-based reverse engineering of animated 3D shapes. *Computer Graphics Forum*, 31(8):2303–2316, 2012.

[28] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2Mesh: Generating 3D mesh models from single RGB images. In *ECCV*, 2018.

[29] Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. 3DN: 3D deformation network. In *CVPR*, 2019.

[30] Zizhao Wu, Ruyang Shou, Yunhai Wang, and Xinguo Liu. Interactive shape co-segmentation via label propagation. *Computers & Graphics*, 38:248–254, 2014.

[31] Chuhua Xian, Hongwei Lin, and Shuming Gao. Automatic cage generation by improved OBBs for mesh deformation. *The Visual Computer*, 28(1):21–33, 2012.

[32] Kai Xu, Honghua Li, Hao Zhang, Daniel Cohen-Or, Yueshan Xiong, and Zhi-Quan Cheng. Style-content separation by anisotropic part scales. *ACM Trans. Graph.*, 29(6):184:1–184:10, 2010.

[33] Kangxue Yin, Zhiqin Chen, Hui Huang, Daniel Cohen-Or, and Hao Zhang. LOGAN: Unpaired shape transform in latent overcomplete space. *ACM Trans. Graph.*, 38(6):198:1–198:13, 2019.

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

[35] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

[36] Chenyang Zhu, Kai Xu, Siddhartha Chaudhuri, Renjiao Yi, and Hao Zhang. SCORES: Shape composition with recursive substructure priors. *ACM Trans. Graph.*, 37(6), 2018.