Learning to Infer Semantic Parameters for 3D Shape Editing

Fangyin Wei[1] Elena Sizikova[2] Avneesh Sud[3] Szymon Rusinkiewicz[1] Thomas Funkhouser[1,3]
[1] Princeton University, NJ, USA [2] New York University, NY, USA [3] Google Research, CA, USA

Abstract

*Many applications in 3D shape design and augmentation require the ability to make specific edits to an object's semantic parameters (*e.g.*, the pose of a person's arm or the length of an airplane's wing) while preserving as much existing details as possible. We propose to learn a deep network that infers the semantic parameters of an input shape and then allows the user to manipulate those parameters. The network is trained jointly on shapes from an auxiliary synthetic template and unlabeled realistic models, ensuring robustness to shape variability while relieving the need to label realistic exemplars. At testing time, edits within the parameter space drive deformations to be applied to the original shape, which provides semantically-meaningful manipulation while preserving the details. This is in contrast to prior methods that either use autoencoders with a limited latent-space dimensionality, failing to preserve arbitrary detail, or drive deformations with purely-geometric controls, such as cages, losing the ability to update local part regions. Experiments with datasets of chairs, airplanes, and human bodies demonstrate that our method produces more natural edits than prior work.*

1. Introduction

The ability to perform semantically-meaningful manipulation of 3D shapes is crucial to many applications ranging from industrial design to dataset augmentation for 3D learning. Although the space of possible manipulations is large, we focus on editing *semantic parameters*, such as the angle of a person's leg or the height of a chair's seat (see Fig. 1). For maximum control, we wish to allow adjusting these independently, such that changing one parameter preserves the others. Moreover, we would like to preserve topology and fine detail throughout the input shape, both within and far away from the edited region. Finally, to make our method broadly applicable, we would like to avoid dependency on large labeled datasets of 3D models or edits.

Traditional shape manipulation methods [4, 27] first fit predefined handles (*e.g.*, cages, primitives) to input shapes through optimization. The user then manipulates the template, which guides the deformation of the original shape. This approach has several disadvantages. First, cages make it difficult or impossible to deform local regions, as opposed to the shape as a whole. In addition, the algorithms that fit handles to an input shape can be initialization-sensitive. Finally, this approach inherently focuses on *geometric* parameters (*e.g.*, cage control points), and establishing the link between them and semantic parameters (*e.g.*, "seat height") would require significant training on large, labeled datasets.

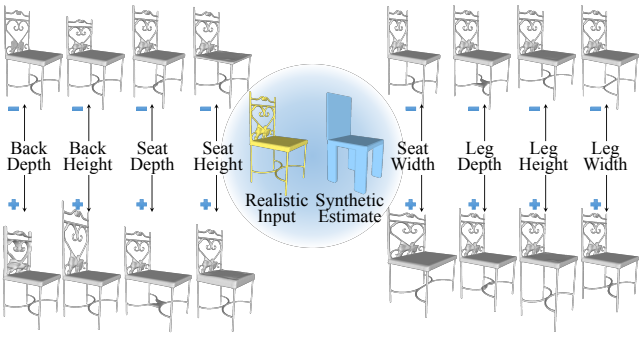

Figure 1: Semantic editing. Taking a realistic shape (yellow) as input, the proposed method allows for editing of different parts (gray). This is achieved by learning to infer the semantic parameters of an auxiliary synthetic shape (blue), without any labels from realistic shapes.

Recently, learning-based methods for shape deformation have become the subject of active research. Some approaches [5, 6, 19] build upon the autoencoder idea: an encoder network maps an input shape to a point in a latent "shape space", while a decoder network re-synthesizes a shape given a latent vector. Using this idea for semantic editing requires training a disentangled encoder, which maps semantic parameters to certain latent dimensions. This requires large, labeled training datasets. More crucially, the fine-scale detail produced by the decoder is fundamentally limited by the latent space dimensionality, and by what detail was present in the training dataset. This means that no resynthesis-based method will be able to preserve the detail present in arbitrary input shapes.

Another popular approach for 3D shape editing is to predict shape deformation [12, 21, 23]. When labeled edits (*e.g.*, "make the seat 50% taller") are available, a network can be trained with full supervision for a deformation

task [24, 25]. However, collecting many such labeled edits is not trivial, and most learnable methods that predict deformation [21, 23] instead solve the task of source-to-target deformation. These methods can only globally deform a shape given another exemplar and do not support local edits with semantic awareness.

In this work, we build upon several key ideas to edit 3D shapes in a semantically meaningful way. First, to allow for full detail preservation, we design our system around deformation and not re-synthesis. The deformation model is flexible enough to allow for both global and local edits, enabling independent control over different semantic parameters. Next, to infer the parameters of an input shape, we rely on an encoder network trained to embed each input into a "semantic parameter space". Crucially, this is a many-to-one network that need not embed all details present in the input as a traditional "shape space" encoder. We train this network using a combination of labeled synthetic shapes and *unlabeled* realistic shapes. We therefore gain the advantage of a semantically-parameterized latent space (whose dimensions are defined by template parameters) without the need for labeled realistic datasets. Finally, at edit time, we use the learned encoder to extract semantic parameters from the input, and then deform the input based on how those parameters change the shape of the synthetic template.

There are several advantages to our method. First, it learns only a semantically interpretable space that is relevant to the editing operations. This is in contrast to learning to encode the entire shape, which suffers from detail loss due to the fixed dimensionality of the latent space. Second, by abstracting realistic shapes into a shared semantic space with synthetic shapes, we sidestep the need for labels for realistic shapes. Finally, the low-dimensional semantic space acts as a regularizer, making the encoder easier to learn. Experiments show that the model generalizes, enabling meaningful edits on shapes that fall out of training distributions.

We test our proposed method on three classes covering both rigid and non-rigid shapes, for different editing tasks. For chairs and airplanes, we consider the application of anisotropic part deformation, driven by semantic labels that need not correspond to individual parts. For example, we can have a "leg width" semantic parameter that controls all four legs simultaneously. For human bodies, we show pose editing trained on a simple body model, which nevertheless generalizes to more realistic bodies. Experiments show that the proposed method produces results that are consistent with the user's desired semantic edits while being more useful and more robust than prior work.

2. Related Work

There is a long history of prior work on semantic shape editing [14, 22]. Relevant to our work, we discuss prior work in shape deformation.

Traditional Template-Based Shape Deformation. Representative early efforts fit templates to input shapes through optimization. For example, Zheng et al. [27] fit a controller (*e.g.*, a cylinder) to every component decomposed from the input shape, then propagate user edits among components to guide input shape deformation. Their shape decomposition works for various man-made shapes, but is mostly based on geometry and is not semantically consistent across shapes. To achieve semantic consistency, Ganapathi-Subramanian et al. [4] fit each input shape with a class-specific refined template. However, such fitting algorithms are fragile, utilizing hand-crafted heuristics and thresholds. We provide a robust deep learning approach to semantic template fitting.

Supervised Style Attribute Learning. While traditional methods often involve manual supervision of templates and editing pipelines, learning-based approaches achieve more automatic editing with priors learned from a large amount of data. Yumer et al. [24] learn a mapping from a shape to an attribute score to guide shape deformation, while Yumer et al. [25] use a deep network to directly learn a volumetric deformation field. However, such supervised methods are extremely data-hungry, whereas large-scale 3D shape datasets with annotated deformations are not currently available.

Deformation Retargeting with Global Structure. Another line of work [17, 18, 21, 23] sidesteps the need for dense annotations by instead focusing on deforming a source shape to globally resemble a target. Wang et al. [21] learn to predict a coordinate offset for each vertex of the input using global shape descriptors, and Wang et al. [23] predict a 40-vertex cage to define the deformation field. While relying less on densely labeled data, these methods are not structure-aware and only allow for coarse shape deformation without precise control over local components. In concurrent work, Sung et al. [18] learn class-specific deformations by projecting user-edited shapes onto a plausible learned shape space. Unlike our work, this method requires realistic source-target pairs for training, and does not guarantee shape preservation for unedited components.

Deformation Retargeting with Learned Templates. To achieve more local editing, more recent work learns to locally deform a template and retarget the deformation to the input shape. Mehr et al. [12] propose to learn a disconnected manifold from multiple auto-encoders with each deforming a different learned template. After training, they optimize over template deformation to fit a user edit, and the optimized template deformation is retargeted to edit the input shape. Although this approach creates a user-friendly interface, it does not support semantic edits and is sensitive to the robustness of latent space optimization. Our method also deforms a realistic shape guided by a deformation field learned using synthetic shapes, and we can more specifi-

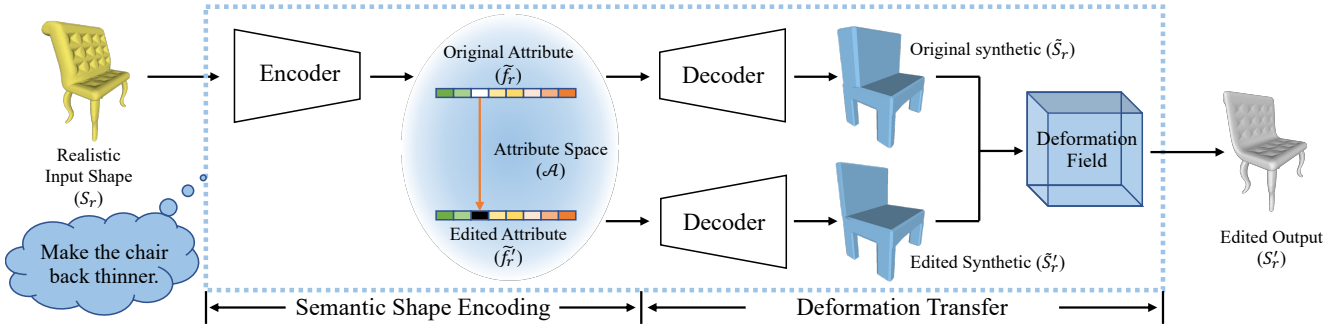

Figure 2: Framework overview. To achieve shape editing, we propose a framework that (a) learns a semantic parameter space for both realistic shapes S_r and auxiliary synthetic shapes S_s (Sec. 4.1) and (b) transfers the deformation of synthetic shapes to realistic shapes (Sec. 4.2).

cally control different semantic components of the realistic shape, with more continuous and precise manipulation. Furthermore, while [12] focuses on learning a disconnected manifold of shapes, we specifically advocate for learning only a small set of parameters for shape editing rather than learning to encode the entire shape. Meanwhile, there exist deformation retargeting approaches using domain-specific templates — statistical models as well as learned embeddings — for humans [1, 7]. In contrast, our approach generalizes across rigid and non-rigid shapes.

Learning Representations for Shape Resynthesis. A highly related field is learned shape reconstruction, which aims to reconstruct a 3D shape from an inferred latent code. Several recent shape reconstruction methods [5, 6, 9, 13, 15] allow for simple shape editing by manipulating the latent representation. Due to the limited scope of the latent space, these reconstruction algorithms can only reproduce shapes similar to ones observed in training, thus losing details after editing. Finally, the shape representation cannot be easily disentangled into semantic factors in the absence of labeled data [5, 10]. Therefore, we argue that for the purpose of shape editing, we should not encode the entire shape. Instead, it is sufficient to only encode the modes of shape variation corresponding to the desired edits, which can be efficiently represented with a small set of parameters. We analytically decode from the space of semantic parameters, which guides the deformation to be applied to the input.

3. Approach

Our approach decomposes shape editing into two stages: semantic shape encoding and deformation transfer (Fig. 2). The original realistic shape is first passed into an encoder that infers its semantic parameters. Manipulation is then performed in the parameter space based on the target editing. Afterward, both original and edited semantic parameters are taken as input by an analytical decoder that outputs synthetic shapes reflecting the editing. The final deformation of the original input is guided by a deformation field defined by the predicted synthetic shapes.

The main learning challenge in this approach is to train an encoder that can extract semantic parameters from arbitrary input shapes. To address this challenge, we must answer two main questions: 1) how to allow a user to define semantic parameters, and 2) how to obtain training data for a network to learn to infer them.

Our approach for the first question is to ask the model designer to create a synthetic template with semantic parameters defining a space of possible shapes (*e.g.*, a template for chairs may have parameters controlling the seat depth, back height, leg length, *etc.* for a set of boxy primitives). The templates are simple and produced once per object category, so the burden of creating them is small (a few minutes) and certainly far less than labeling a large dataset of examples.

Our approach for the second question is to procedurally build a semi-supervised training set with a mix of unlabeled realistic shapes and labeled synthetic shapes derived from the template. We train the semantic encoder to predict the semantic parameters of the template examples and to match the shapes of all examples. Through joint training on realistic and synthetic examples, the network learns to infer semantic parameters for realistic shapes with only precise semantic supervision from synthetic data.

4. Method

Our editing pipeline takes in a realistic input shape S_r and generates its deformed version S'_r defined by changes to semantic parameters (Fig. 2). To achieve this, we first use a trained encoder E to infer semantic parameters $\tilde{f}_r$ of the input shape S_r. Those parameters can be edited or optimized (depending on the application) to obtain $\tilde{f}'_r$. We then use an analytical decoder to produce synthetic shapes $\tilde{S}_r$ and $\tilde{S}'_r$ from $\tilde{f}_r$ and $\tilde{f}'_r$ respectively. Finally, the pair of synthetic shapes are used to define a deformation field $\mathcal{D}$ that is applied to the input shape S_r to output shape S'_r.

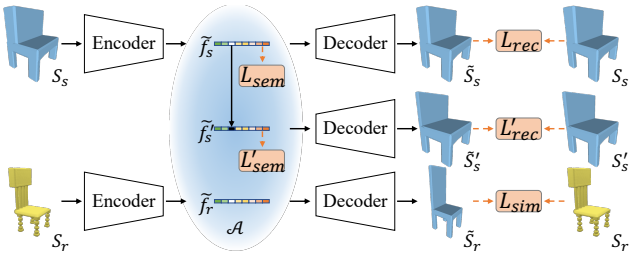

Figure 3: Learning framework. We learn a semantic space $\mathcal{A}$ (blue) by jointly training on synthetic and realistic shapes. Synthetic shapes are supervised in both semantic and shape space, with one reconstruction branch (top) and one editing prediction (middle). Realistic shapes (bottom) are supervised in shape space only.

4.1. Semantic Shape Encoding

Learning Framework. During training, we optimize the encoder to learn a semantic parameter space shared by both synthetic and realistic shapes. Specifically, in each mini-batch, we set half of the examples to be synthetic and half realistic. The encoded semantic parameters of synthetic shapes are passed to two branches. The first branch (top row in Fig. 3) reconstructs the input shape from the encoded semantic feature $\tilde{f}_s (d = |\tilde{f}_s|)$. The second branch (middle row in Fig. 3) predicts the edited shape from a new semantic feature $\tilde{f}'_s$ which is obtained by modifying the original feature $\tilde{f}_s$ according to a target edit. Both features are then decoded into synthetic shapes by the decoder. We show in Sec. 6.5 that the second editing branch is essential for maintaining training stability and improving performance. Both branches are supervised by a semantic parameter loss L_{sem} and a shape reconstruction loss L_{rec}. For each realistic shape (bottom row in Fig. 3), the encoder estimates a semantic feature $\tilde{f}_r$, which is then decoded to a synthetic shape. A shape similarity loss L_{sim} is used to make sure the synthetic shape is close to the realistic shape semantically, since no ground truth is available for realistic shapes.

Semantic Parameter Loss. With the ground truth semantic parameters for synthetic shapes, we define

$$L_{sem} = dist(f_s, \tilde{f}_s) \text{ and } L'_{sem} = dist(f'_s, \tilde{f}'_s)$$

as semantic parameter losses for reconstruction and editing branches, respectively, where *dist* is a distance metric (*e.g.*, L2 distance for scaling parameters, geodesic distance for rotational parameters). By training with supervision from synthetic shapes expressing a dense range of semantic parameter values, we learn to encode the full distribution of the space, which enables highly generalizable semantic edits by moving along dimensions in the space.

Shape Reconstruction Loss. In the 3D shape space, since we know the exact vertex correspondence between input and predicted shapes, we have:

$$L_{rec} = \sum_{i=1}^{n} \left\| \tilde{v}_s^{(i)} - v_s^{(i)} \right\|^2 \text{ and } L'_{rec} = \sum_{i=1}^{n} \left\| \tilde{v}_s'^{(i)} - v_s'^{(i)} \right\|^2$$

for the reconstruction and editing branches, respectively, where n is the total number of vertices, and $v_s^{(i)}$ and $\tilde{v}_s^{(i)}$ are the vertices from the ground truth mesh and predicted synthetic mesh, respectively.

Shape Similarity Loss. For realistic shapes, since no ground truth semantic parameters are available, we only impose a shape similarity loss in the 3D space between the output synthetic predictions and the input realistic shapes. We choose the shape similarity loss as the chamfer distance between randomly sampled point clouds from both shapes:

$$L_{sim} = \sum_{v \in \tilde{S}_r} \min_{u \in S_r} \|v - u\|^2 + \sum_{u \in S_r} \min_{v \in \tilde{S}_r} \|u - v\|^2.$$

Overall Learning Objective. The total loss L is a weighted combination of all loss terms as described above:

$$L = \alpha \left(L_{sem} + L'_{sem} \right) + \beta \left(L_{rec} + L'_{rec} \right) + \gamma \, L_{sim},$$

where α, β, γ are the weights for individual terms.

Training Details. The encoder can be flexible according to the input representation. For a point cloud input, we use the PointNet [16], which is composed of seven fully-connected layers interlaced with Batch Normalization and ReLU layers. The sizes of layers from bottom to top are (64, 128, 128, 256, d, d, d). For a mesh input, we use a graph CNN, which is composed of a fully-connected layer with output feature size 16, four FeaStConv [20] layers with output feature size (32, 64, 96, 128), and one final fully-connected layer with a task-specific feature size d. All layers are interlaced with a Leaky ReLU layer. For non-rigid experiments, when all realistic input has the same mesh connectivity (*e.g.*, DFAUST), we can use a mesh encoder. When the realistic shapes do not follow the same mesh structure (*e.g.*, scanned data), we use a point encoder. The differentiable decoder takes as input the semantic parameters and analytically outputs a synthetic mesh.

Our implementation uses the PyTorch framework and the Adam [8] optimizer, with learning rate 0.001 and batch size 16 (half synthetic and half realistic), running on one GPU until convergence. We set the weights in our loss function to be $\alpha = 0.3$, $\beta = 30$, $\gamma = 50$ for chairs, $\alpha = 0.3$, $\beta = 200$, $\gamma = 10$ for airplanes, and $\alpha = 0.03$, $\beta = 4$, $\gamma = 1$ for human bodies. (See Sec. 5 for details about datasets.) All input shapes lie within a sphere of radius 0.6 and are re-centered such that points have zero mean. For all tasks, the actual prediction by the encoder includes the semantic parameters as described above plus a global translation. All datasets are split 4:1 into training and testing sets.

4.2. Deformation Transfer

To tackle the challenge of detail preservation while performing the editing, we transfer the deformation defined by synthetic shapes to realistic shapes that are semantically close. A new input realistic shape S_r is first encoded into a semantic feature $\tilde{f}_r$. Then, a new feature $\tilde{f}'_r$ is obtained by modifying $\tilde{f}_r$ based on the target editing operation. Both $\tilde{f}_r$ and $\tilde{f}'_r$ are decoded into synthetic shapes. Because the decoder is completely analytical, we can trace the transformation for each point on the synthetic prediction during the editing. Then we can decide on a specific algorithm to define a deformation field which further guides the deformation of the input realistic shape. For example, suppose the per-vertex deformation is $D_s : S_r \mapsto S'_r$, *i.e.*, $v'_s = D_s(v_s), v_s \in S_r, v'_s \in S'_r$. Then we can define the deformation field as:

$$D_r(x) = \frac{\sum_{v \in KNN_S(x,k)} W(x,v) \cdot D_s(v)}{\sum_{v \in KNN_S(x,k)} W(x,v)},$$

where $x \in \mathbb{R}^3$, $KNN_S(x,k)$ is the set of k nearest neighbors on shape S_r of point x ($k = 1, 2, \dots$), $W(\cdot,\cdot)$ defines the weights among k neighbors and can be either constant or a function of points in the field and on the synthetic shape. In the experiments, we set weights as $W(x,v) = (1 + \langle \mathbf{n}_x, \mathbf{n}_v \rangle)^{k_n} \cdot e^{-\frac{\|v-x\|^2}{\sigma^2}} (k_n = 2, \sigma = 0.03)$ where $\mathbf{n}_p$ is the normal of vertex p ($p = x, v$) for rigid shapes and $W(x,v) = 1$ for non-rigid shapes.

5. Datasets

We consider both rigid and non-rigid examples to thoroughly evaluate our proposed method.

Rigid Shapes. We consider chairs and airplanes from the ShapeNet [3] dataset, a commonly used large-scale dataset for 3D model evaluation. In order to generate editable synthetic data, we create a template that captures important semantic parameters for each class. Each synthetic chair is composed of 6 cuboids with a predefined structure and eight semantic parameters: the height, depth, and width of the back, seat, and leg.[1] The synthetic airplane is a simplified airplane with a smoothed fuselage, two wings, and one vertical and two horizontal stabilizers. We define six semantic parameters: the height, length, and width of the fuselage, and the lengths of the wings, vertical stabilizer, and horizontal stabilizers. The dataset includes 200 and 620 realistic shapes for chairs and airplanes, respectively, and 40000 synthetic shapes for each class. All inputs are point clouds with 2840 and 2750 vertices for chairs and airplanes.

[1] Since chair back and seat share the same width in practice, the width of both parts is merged into one parameter.

Table 1: Dataset and task overview. Results are presented for 3 shape classes; edits include structure-aware part deformation for man-made objects and human body animation.

Class	Operation[2]	Synthetic Data	Realistic Data
Chair	Aniso. Scale	Cuboids	ShapeNet [3]
Airplane	Aniso. Scale	Simplified Airplane	ShapeNet [3]
Human	Aniso. Scale, Rot.	SMPL [11]	DFAUST [2], Buff [26]

Non-Rigid Shapes. We conduct experiments on a realistic dataset of human bodies sampled from two sources. First, we include 5663 3D scans of human bodies from DFAUST [2]. To incorporate the additional challenge of clothing, we also include 962 clothed bodies from Buff [26]. To generate a dataset with edits, we consider SMPL [11], a synthetic shape template that models pose and shape parameters of the human body. Specifically, the semantic space covers 69 pose parameters (23 joints with 3 degrees of freedom each) and 3 shape parameters, the linear combination of which corresponds to a 6890-vertex synthetic mesh. It is straightforward to deform meshes reconstructed from SMPL [11] with known shape/pose parameters, which is a much more challenging task for real-world scanned data. The synthetic data is randomly sampled from SMPL parameter space with a Gaussian distribution prior, and all testing shapes are held-out characters not seen during training.

6. Evaluation

We test the proposed 3D shape editing system on the tasks of piecewise anisotropic scaling and non-rigid deformation, for the three datasets in Tab. 1. Sec. 6.1 and 6.2 qualitatively demonstrate editing of semantic parameters on rigid and non-rigid datasets, while Sec. 6.3 further tests the method's generalization on out-of-distribution examples. Sec. 6.4 shows comparisons to Neural Cages [23], a recent technique for source-to-target matching with a learned deformation, and DualSDF [6], a recent deformation method with learned resynthesis. For each of the above sections, we provide additional results in the Supplementary Material. Finally, Sec. 6.5 presents ablation studies on key components of the proposed method. The proposed method requires approximately 3 ms for encoding, and deformation can take tens of milliseconds through seconds, depending on object size. For comparison, DualSDF requires approximately 175 ms for their gradient descent procedure, and seconds to tens of seconds for ray-tracing results.

6.1. Piecewise Anisotropic Scaling Results

A common editing strategy for objects with well-defined semantic parts is anisotropic scaling of different parts independently. In Fig. 4, we show edits produced by the pro-

[2] Aniso. Scale stands for local (per-part) anisotropic scaling, and Rot. stands for rotation.

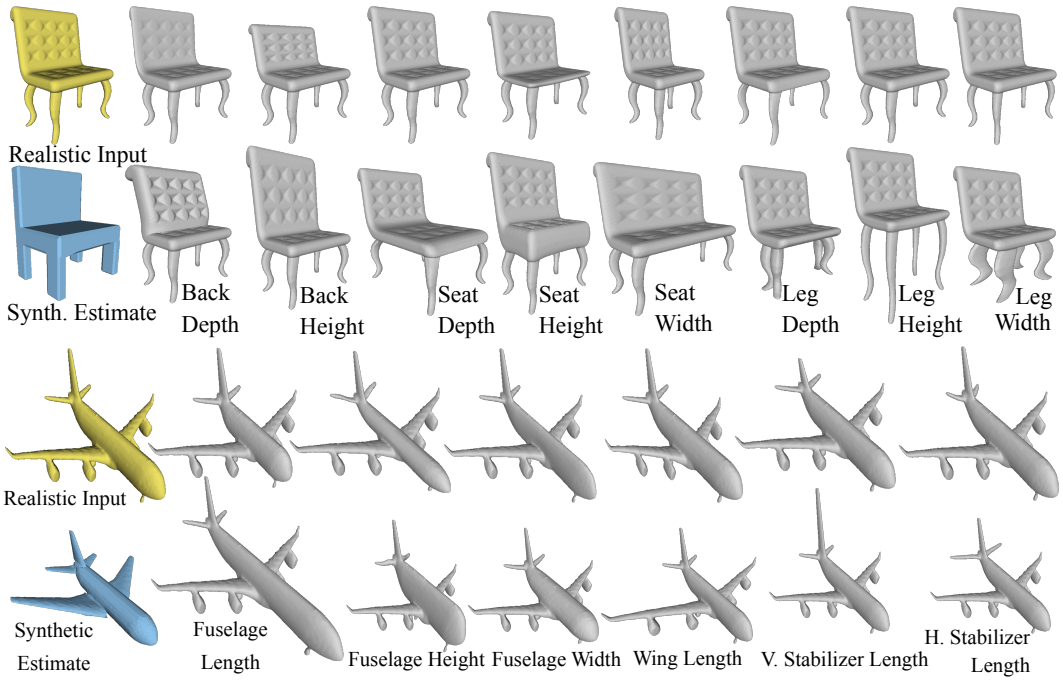

Figure 4: Editing results with piecewise anisotropic scaling. Realistic input shapes (yellow) are fit to synthetic templates (blue), then edited by decreasing (top row) and increasing (bottom row) each shape parameter in turn. The proposed method provides for both detail preservation and independent control over each semantic parameter.

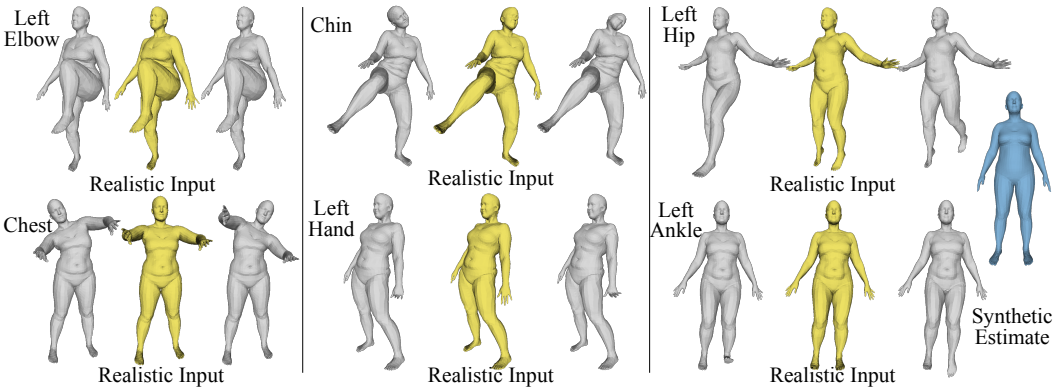

Figure 5: Editing results for deformation of human bodies. In each set of 3 shapes, an input human body (yellow) is deformed by changing one semantic parameter in both directions. An example synthetic template (blue) is shown at right.

posed method for chairs and airplanes (which were held out from training). For each class, we show an input shape (yellow) and the estimated synthetic template (blue). Note that the latter matches the realistic shape in all structural components relevant to editing, such as the seat of the chair and wing length of the airplane. This shows that the model correctly infers semantic parameters of realistic shapes, though trained with only synthetic parameter annotations. In gray, we show the deformation result of one semantic parameter per column, with the first row decreasing and the second row increasing that parameter. As hoped, modifying one semantic parameter does not affect the shape of other parts; *e.g.*, modifying the depth of a chair's seat does not change the shape of its legs. At the same time, parameters correctly update groups of similar parts. For example, modifying the "leg height" parameter for chairs updates all four legs, while modifying "wing length" updates both airplane wings. Note that details are well-preserved after deformation (*e.g.*, tufts on the chair back and seat, curved chair legs, engines and landing gear of the airplane). The proposed method allows for substantial parameter change (*e.g.*, height of chair back and legs, width of chair seat, length of airplane fuselage), in contrast to methods that train on shape databases that might lack such extreme deformations.

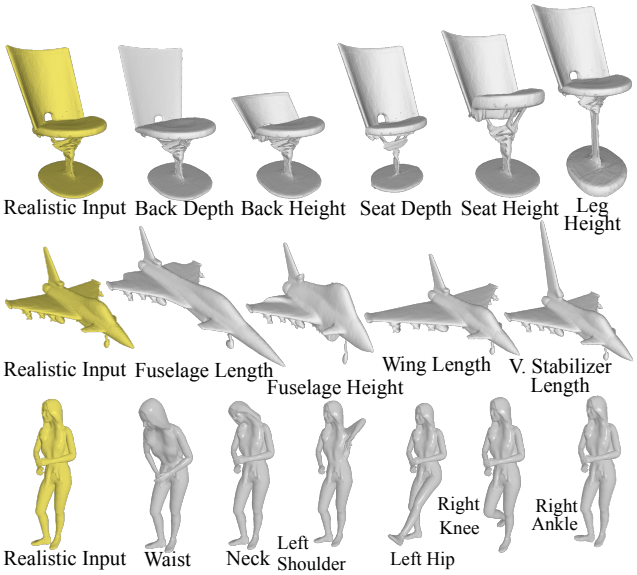

Figure 6: Editing results for out-of-distribution shapes. Our method produces reasonable results, and correctly preserves detail, even if the input shape has missing parts or different topology relative to the synthetic template.

6.2. Non-Rigid Deformation Results

In Fig. 5, we show results for editing semantic pose parameters of human bodies. Our system is able to apply the deformations defined by a simple, parameterized skinned model (SMPL) to realistic human input shapes (DFAUST), accommodating both large (hip movement) and subtle (ankle movement) motions. Note that each joint has up to three degrees of freedom, though because of space constraints we demonstrate only one degree of freedom per joint. Our model correctly decouples the effects of all parameters, even those affecting the same joint.

6.3. Out-of-Distribution Shapes

To evaluate the generalization ability of the proposed method, we test it on shapes falling outside of the shape distribution of the training dataset. Fig. 6 shows examples from all three classes, illustrating cases in which the test shapes are topologically different from the training examples or are missing some of the components present in the synthetic shapes. For example, the chair at the top has only one leg, the airplane has a delta wing and is missing horizontal stabilizers, while the human has long hair and an initial pose with the hands almost close together. In all cases, the proposed method produces reasonable deformed results and preserves input shape details.

6.4. Comparison to Prior Work

To compare against Neural Cages [23], which was designed for the task of source-to-target deformation rather than direct manipulation of semantic parameters, we modify the proposed method to use a deformation determined by templates fit to both the source and target shapes. Fig. 7 shows four examples, including both in- and out-of-distribution chairs and two human body shapes evaluated in the original Neural Cages work. Both methods are able to

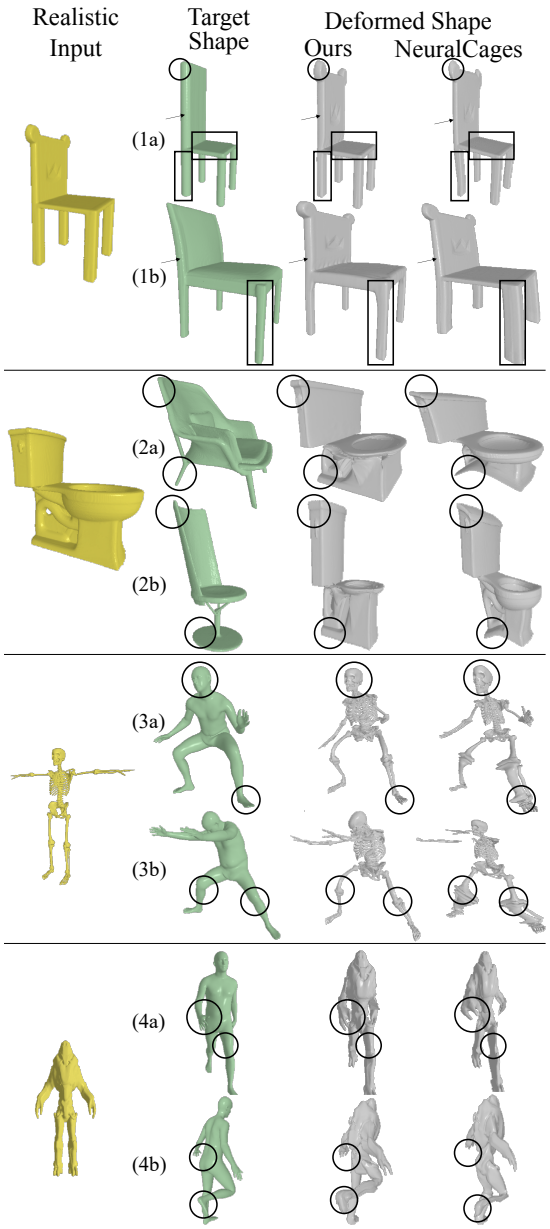

Figure 7: Comparison to Neural Cages [23]. Each source shape (yellow) is deformed to match two target shapes (green), using both the proposed method and Neural Cages. Both are able to globally match the target shape, but Neural Cages often exhibits distortion in regions of large deformation, and an inability to match semantic parameters if local deformation is required. Our results support more accurate and granular manipulation, even for extreme poses.

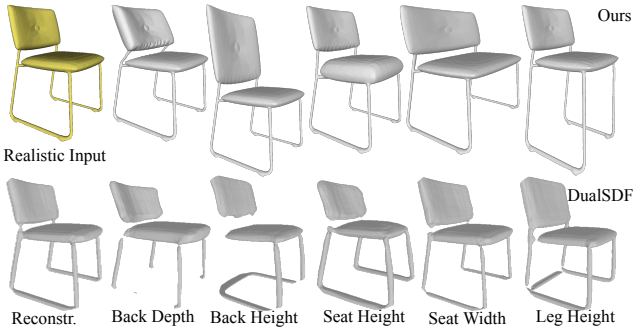

Figure 8: Comparison to DualSDF [6]. The input (yellow) is edited by changing a semantic parameter in our system, or adjusting the radius of a primitive in DualSDF. When changing a local parameter with DualSDF, the global shape is also affected, as seen in the alteration or omission of the bottom crossbars on the legs. In addition, in contrast to the proposed method, the DualSDF results lack details such as the button on the chair back.

match the target shape globally. However, note the spurious deformation in the legs in the Neural Cages results for (1a) and (1b), as well as the fact that in (1b) the Neural Cages result matches the seat thickness of the source, not the target. In contrast, the proposed method does not introduce unnecessary deformations, yet is able to deform the source locally, not just globally, to match the seat thickness. In the human body results, Neural Cages can introduce distortion in regions with large deformation, such as the knees in example (3b) or disproportionately scaled hands in examples (4a) and (4b). We observe that, compared to deformation methods based on coarse geometric handles, our results support more granular manipulation even with extreme poses.

We also compare the proposed method with DualSDF [6] on the task of part manipulation. We consider examples common to DualSDF's training set and our testing set. To generate results for DualSDF, we first obtain the embedding of an input shape and then manually adjust the radius of a primitive on the corresponding part along the corresponding dimension. Fig. 8 shows the results for one chair example. The DualSDF reconstruction, which requires re-synthesis, loses details of the original shape such as the button on the back and the four anti-skid pads on the legs (please zoom in for a clear view). In addition, manipulating a single primitive in one part will cause deformation in other parts as well, since the latent embeddings from DualSDF are highly entangled. For example, when manipulating primitives on the back, the leg topology is also changed by DualSDF, but correctly preserved by the proposed method.

6.5. Ablation Studies

We analyze the influence of model losses on the accuracy of edits to synthetic shapes (chairs). As shown in Tab. 2, the semantic parameter loss (L_{sem}) alone is more effective than shape reconstruction loss (L_{rec}) alone. Combining both loss terms achieves the best performance on synthetic shapes than either single loss. From the last two rows, it can be seen that the editing branch for synthetic shapes is essential for reducing mean vertex error (MVE). Also, we observe that models with an editing branch are more stable during training. Finally, we have evaluated the effects of using either ℓ_1 or ℓ_2 distance for L_{sem}; we find that ℓ_2 performs better in our training.

Table 2: Effectiveness of loss terms and the editing branch on the proposed model. We show the percentage of vertices in synthetic testing shapes (from the chair class) having Mean Vertex Error (MVE) below a given threshold. For example, the rightmost entry of the bottom row indicates that 95.4% of vertices are within a distance of 0.03 of their ground-truth locations.

Loss Terms	Editing Branch	MVE<0.01	MVE<0.02	MVE<0.03
L_{sem}	✓	35.2%	72.3%	87.2%
L_{rec}	✓	11.9%	47.9%	77.6%
$L_{sem} + L_{rec}$		11.5%	55.0%	83.6%
$L_{sem} + L_{rec}$	✓	40.3%	82.2%	95.4%

7. Conclusion and Future Work

In this paper, we present a learning framework for detail-preserving semantic 3D shape editing. We propose to infer semantic parameters of input examples by leveraging a simple synthetic shape set and learning a joint low-dimensional embedding for synthetic and realistic shapes. This approach allows the proposed method to relieve the need for semantic part labels or example user edits on realistic shapes, while allowing semantically consistent edits for all shapes (including out-of-distribution examples). Experiments on both rigid and non-rigid shapes demonstrate that the proposed method provides detail-preserving, structure-aware semantic editing and compares favorably with prior work.

This work is a first step toward learning a semantic 3D shape editing system, and there are several ways the proposed method can be extended. Currently, the semantic encoding is learned and the deformation is analytic. Including both in an end-to-end learning pipeline is a valuable future direction. Also, it is interesting to consider how our semantic encoding can be combined with other deformation transfer strategies, possibly involving training on unlabeled shape datasets, to achieve new types of shape edits.

Acknowledgement We would like to thank Kyle Genova for sharing watertight ShapeNet meshes and Francesco Locatello for discussions about disentangled representation. We also thank the National Science Foundation (grant #IIS-1815070) for partial support of this work.

References

[1] T. Alldieck, M. Magnor, B. L. Bhatnagar, C. Theobalt, and G. Pons-Moll. Learning to reconstruct people in clothing from a single RGB camera. *CVPR*, 2019. 3

[2] F. Bogo, J. Romero, G. Pons-Moll, and M. J. Black. Dynamic FAUST: Registering human bodies in motion. In *CVPR*, 2017. 5

[3] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], 2015. 5

[4] V. Ganapathi-Subramanian, O. Diamanti, S. Pirk, C. Tang, M. Niessner, and L. Guibas. Parsing geometry using structure-aware shape templates. In *3DV*, 2018. 1, 2

[5] L. Gao, J. Yang, T. Wu, Y.-J. Yuan, H. Fu, Y.-K. Lai, and H. Zhang. SDM-NET: Deep generative network for structured deformable mesh. *ACM Transactions on Graphics (TOG)*, 2019. 1, 3

[6] Z. Hao, H. Averbuch-Elor, N. Snavely, and S. Belongie. DualSDF: Semantic shape manipulation using a two-level representation. *arXiv preprint arXiv:2004.02869*, 2020. 1, 3, 5, 8

[7] Z. Huang, Y. Xu, C. Lassner, H. Li, and T. Tung. ARCH: Animatable reconstruction of clothed humans. In *CVPR*, 2020. 3

[8] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, 2015. 4

[9] J. Liu, F. Yu, and T. Funkhouser. Interactive 3D modeling with a generative adversarial network. In *3DV*, 2017. 3

[10] F. Locatello, S. Bauer, M. Lucic, G. Raetsch, S. Gelly, B. Schölkopf, and O. Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *ICML*, 2019. 3

[11] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 2015. 5

[12] E. Mehr, A. Jourdan, N. Thome, M. Cord, and V. Guitteny. DiscoNet: Shapes learning on disconnected manifolds for 3d editing. In *ICCV*, 2019. 1, 2, 3

[13] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3D reconstruction in function space. In *CVPR*, 2019. 3

[14] N. J. Mitra, M. Wand, H. Zhang, D. Cohen-Or, V. Kim, and Q.-X. Huang. Structure-aware shape processing. In *ACM SIGGRAPH 2014 Courses*. 2014. 2

[15] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. 3

[16] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 4

[17] R. W. Sumner and J. Popović. Deformation transfer for triangle meshes. *ACM Transactions on graphics (TOG)*, 23(3):399–405, 2004. 2

[18] M. Sung, Z. Jiang, P. Achlioptas, N. J. Mitra, and L. J. Guibas. Deformsyncnet: Deformation transfer via synchronized shape deformation spaces. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 2020. 2

[19] Q. Tan, L. Gao, Y.-K. Lai, and S. Xia. Variational autoencoders for deforming 3D mesh models. In *CVPR*, 2018. 1

[20] N. Verma, E. Boyer, and J. Verbeek. Feastnet: Feature-steered graph convolutions for 3D shape analysis. In *CVPR*, 2018. 4

[21] W. Wang, D. Ceylan, R. Mech, and U. Neumann. 3DN: 3D deformation network. In *CVPR*, 2019. 1, 2

[22] K. Xu, V. G. Kim, Q. Huang, N. Mitra, and E. Kalogerakis. Data-driven shape analysis and processing. In *SIGGRAPH ASIA 2016 Courses*. 2016. 2

[23] W. Yifan, N. Aigerman, V. Kim, S. Chaudhuri, and O. Sorkine-Hornung. Neural cages for detail-preserving 3D deformations. *arXiv preprint arXiv:1912.06395*, 2019. 1, 2, 5, 7

[24] M. E. Yumer, S. Chaudhuri, J. K. Hodgins, and L. B. Kara. Semantic shape editing using deformation handles. *ACM Transactions on Graphics (TOG)*, 2015. 2

[25] M. E. Yumer and N. J. Mitra. Learning semantic deformation flows with 3D convolutional networks. In *ECCV*, 2016. 2

[26] C. Zhang, S. Pujades, M. J. Black, and G. Pons-Moll. Detailed, accurate, human shape estimation from clothed 3D scan sequences. In *CVPR*, 2017. 5

[27] Y. Zheng, H. Fu, D. Cohen-Or, O. K.-C. Au, and C.-L. Tai. Component-wise controllers for structure-preserving shape manipulation. In *Computer Graphics Forum*, 2011. 1, 2