Learning Shape Templates with Structured Implicit Functions

Kyle Genova[1,2] Forrester Cole[2] Daniel Vlasic[2] Aaron Sarna[2] William T. Freeman[2] Thomas Funkhouser[1, 2]

[1]Princeton University [2]Google Research

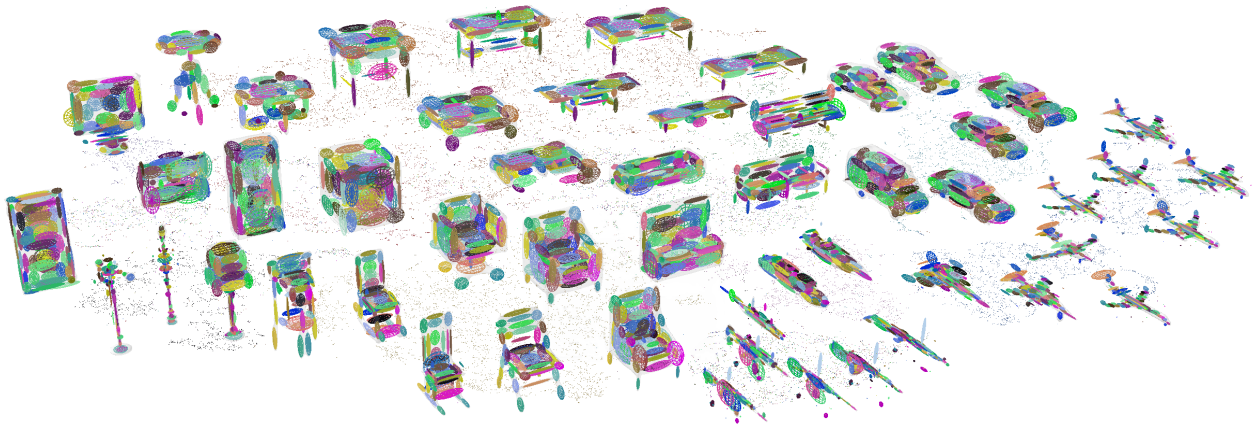

Figure 1. Shapes from the ShapeNet [8] database, fit to a structured implicit template, and arranged by template parameters using t-SNE [51]. Similar shape classes, such as airplanes, cars, and chairs, naturally cluster by template parameters.[1]

Abstract

Template 3D shapes are useful for many tasks in graphics and vision, including fitting observation data, analyzing shape collections, and transferring shape attributes. Because of the variety of geometry and topology of real-world shapes, previous methods generally use a library of hand-made templates. In this paper, we investigate learning a general shape template from data. To allow for widely varying geometry and topology, we choose an implicit surface representation based on composition of local shape elements. While long known to computer graphics, this representation has not yet been explored in the context of machine learning for vision. We show that structured implicit *functions are suitable for learning and allow a network to smoothly and simultaneously fit multiple classes of shapes. The learned shape template supports applications such as shape exploration, correspondence, abstraction, interpolation, and semantic segmentation from an RGB image.*

1. Introduction

Fitting a 3D shape template to observations is one of the oldest and most durable vision techniques [42]. Templates offer a concise representation of complex shapes and a strong prior for fitting. They can be used to directly correspond and compare shapes, and supervised learning approaches may be applied to correspond the template and a photograph [53, 5]. In order to fit a wide range of shapes, however, multiple, hand-made templates are usually required, along with a procedure for choosing the appropriate one [13].

The goal of this paper is to construct a general shape template that fits any shape, and to learn the parameters of this template from data. We view a shape as a level set of a volumetric function and approximate that function by a collection of shape elements with local influence, a formulation we term a *structured implicit* function. The template itself is defined by the number of and formula for the shape elements, and the template parameters are simply the concatenation of the parameters of each element. An example of this type of representation is the classic *metaballs* method [3], but more sophisticated versions have been proposed since [56, 4, 36].

Given a template definition, we show that a network can be trained to fit the template to shapes with widely varying geometry and topology (Figure 1). Critically, the net-

[1]See templates.cs.princeton.edu for video, supplemental, and high resolution results.

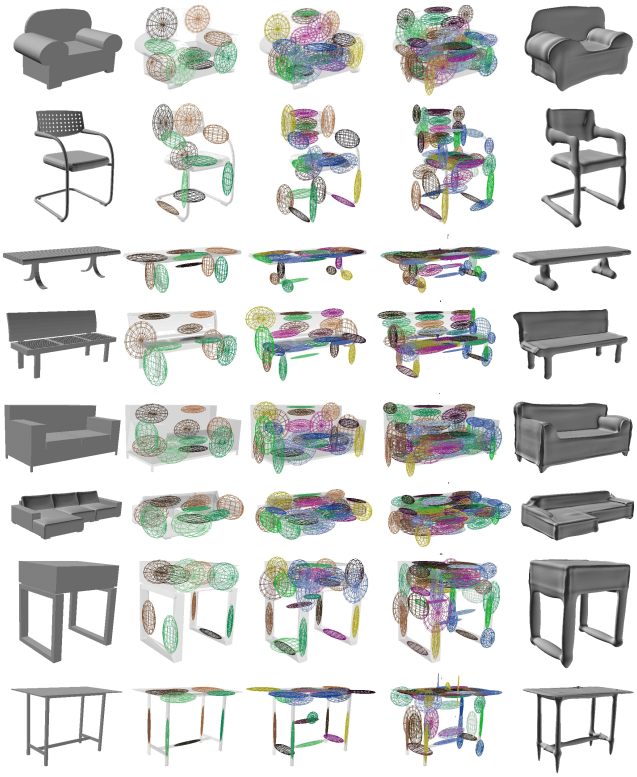

Figure 2. Templates fit to a variety of geometry and topology. Middle columns: three shape templates trained across classes with 10, 25, and 100 elements, respectively. Right: surface reconstruction of the implicit function defined by the 100 element template. Note how the structure of each template is consistent between shapes.

work learns a fitting function that is smooth: the template parameters of similar shapes are similar, and vary gradually through shape-space (Figure 2). Further, we show that the network learns to associate each shape element with similar structures in each shape: for example, the tail fin of an aircraft may be represented by one element, while the left wingtip may be represented by another. This consistency allows us to interpolate shapes, estimate vertex correspondences, or predict the influence region of a given element in a 2D image, providing semantic segmentation of shapes.

The closest related work to ours is the volumetric primitive approach of Tulsiani, et al. [50]. Like that work, we aim to learn a consistent shape representation with a small number of primitives. We expand on their work by specifying the surface as a structured implicit function, rather than as a collection of explicit surface primitives. This change allows for an order of magnitude increase in the number of shape elements, allowing our template to capture fine details.

Our method is entirely self-supervised and requires only a collection of shapes and a desired number of shape elements (N, usually 100). The output template is concise (7N values) and can be rendered or converted to a mesh using techniques such as raytracing or marching cubes [30].

2. Related Work

There is a long history of work on shape analysis aimed at extracting templates or abstract structural representations for classes of shapes [18, 17, 24, 33, 57].

Primitive Fitting: Fitting of basic primitives is perhaps the oldest topic in 3D computer vision, beginning with Roberts [42] and continuing to today [2, 19, 29, 27]. These methods focus on explaining individual observations with primitives, and do not necessarily provide consistency across different input shapes, so they cannot be used for the correspondence, transfer, and exploration applications targeted in this paper.

Part Segmentation: Others have studied how to decompose mesh collections into consistent sets of semantic parts, either through geometric [14] or learned methods [1, 12, 20, 25]. These methods differ from ours in that they depend on labeled examples to learn the shapes and arrangements of *semantic parts* within specific classes. In contrast, we aim to learn a *structural* template shape for any class without human input.

Template Fitting: The most related techniques to ours are methods that explicitly fit templates to shapes [7]. The templates can be provided by a person [13, 37], derived from part segmentations [59, 26, 12], or learned automatically [23, 50, 58, 59]. Previous work generally assumes an initial set of primitives or part structure is given prior to learning. For example, Kim et al. [23] proposed an optimization to fit an initial set of box-shaped primitives to a class of 3D shapes and used them for correspondence and segmentation. Part structure is assumed by [59].

Others have learned shape templates with a neural network. In Zou et al. [60], a supervised RNN is trained to generate sets of primitives matching those produced by a heuristic fitting optimization. Sharma et al. [45] use reinforcement learning to decompose input shapes into a CSG parse tree. Like our approach, this approach does not require additional training data, but CSG trees are unsuitable for many template applications.

Tulsiani et al. [50] proposed a neural network that learned placements for a small number (3 to 6) of box primitives from image or shape inputs, without additional supervision. Our method builds on this approach, but greatly expands the number and detail of the shape elements, allowing for the precise shape associations required for correspondence and semantic segmentation applications.

Implicit Shape Representations: Decades ago, researchers in computer graphics proposed representing shapes with sets of local shape functions [41, 3]. The most common form is a summation of polynomial or Gaussian basis functions centered at arbitrary 3D positions, sometimes called *metaballs* [3], blobby models [34], or soft objects [56].

Property	Voxel	Octree	Point	Mesh	Deep	Ours
Interpret	+	+	+	+	-	+
Concise	-	+	+	+	-	+
Surface	+	+	-	+	-	+
Volume	+	+	-	-	+	+
Topology	+	+	-	-	+	+
Deform	-	-	+	+	-	+

Table 1. Comparison of desirable properties of various 3D representations, rated as suitable (+) or unsuitable (-). From top to bottom: is the representation interpretable to humans; concise in storage; capable of representing surfaces and volumes; allows topological changes; and supports smooth deformation. Structured implicit functions are suitable in all properties. "Deep" refers to methods that represent a volumetric function as a deep neural network [38, 46].

Other forms include convolution surfaces [4] and partition of unity implicits [36]. These representations support compact storage, efficient interior queries, arbitrary topology, and smooth blends between related shapes, properties that are particularly useful for our application of predicting template shapes.

Shape Representations for Learning: Recently, several deep network architectures have appeared that encode observations (color images, depth images, 3D shapes, etc.) into a latent vector space and decode latent vectors to 3D shapes. Our work follows this approach. We argue that our structured implicit representation is superior for template learning compared to decoding voxels [6, 54, 55], sparse-voxel octrees [49], points [11], meshes [15, 21, 52], box primitives [50], signed-distance function estimators [38], or indicator function estimators [32].

Table 1 compares the properties of these representations. Compared to points, implicit surfaces are superior because they provide a clearly-defined surface. Compared to meshes, implicit surfaces can continuously adapt to arbitrary topology. Structured implicit functions are most similar to voxel grids since both implicitly represent a surface. Unlike voxel grids, they provide a sparse representation of shape, though octree techniques can provide sparse representations of voxels. The major difference for our work is that our shape elements can be moved and transformed in a smooth way to, for example, track gradual changes in airplane wing shape across a shape collection. By contrast, two similar, but slightly transformed shapes will have entirely different voxel representations.

Techniques have recently been proposed to directly approximate volumetric functions such as signed-distance fields or indicator functions using deep neural networks [38, 32, 46]. Compared to these approaches, structured implicit functions are light weight, easily interpretable, and provide template geometry that can be modified or transformed by later processing.

3. Structured Implicit Shape Representation

We assume each input shape can be modeled as a watertight surface bounding an interior volume (real-world meshes usually must be processed to satisfy this assumption, see Sec. 4.2). We aim to represent this surface as the ℓ level set of a function $F(\mathbf{x}, \mathbf{\Theta})$, where $\mathbf{x}$ is a 3D position and $\mathbf{\Theta}$ is a vector of template parameters. In the structured implicit formulation, F is the sum of the contributions of a fixed number of shape elements with local influence, labeled $i \in [N]$, where N is their count. Each element is a function f_i defined by its parameter vector θ_i (making $\mathbf{\Theta}$ simply the concatenation of θ_i):

$$F(\mathbf{x}, \mathbf{\Theta}) = \sum_{i \in [N]} f_i(\mathbf{x}, \theta_i) \tag{1}$$

The specific version of shape elements we adopt are *scaled axis-aligned anisotropic 3D Gaussians*. Here, θ_i consists of a scale constant c_i, a geometric center $\mathbf{p}_i \in \mathcal{R}^3$, and per-axis radii $\mathbf{r}_i \in \mathcal{R}^3$.

$$f_i(\mathbf{x}, \theta_i) = c_i \exp\left(\sum_{d \in \{x,y,z\}} \frac{-(\mathbf{p}_{i,d} - \mathbf{x}_d)^2}{2\mathbf{r}_{i,d}^2} \right) \tag{2}$$

Intuitively, one can think of this representation as a set of squished or stretched 3D blobs. We found this set of parameters to be the minimum necessary to achieve good results. More sophisticated shape elements, such as full multivariate Gaussians, or even windowed quadric functions [36], would likely improve results, but we do not experiment with those here.

Because all constants c_i are negative, we have that $f_i(\mathbf{x}, \theta_i) < 0$ and thus $F(\mathbf{x}, \mathbf{\Theta}) < 0$, $\forall \mathbf{x} \in \mathcal{R}^3$. Therefore we pick a negative isolevel ℓ and define the surface S to be its crossing:

$$S = \left\{ \mathbf{x} \in \mathcal{R}^3 : F(\mathbf{x}, \mathbf{\Theta}) = \ell \right\} \tag{3}$$

We set $\ell := -0.07$, which was chosen by grid search. The reason that the constants are negative rather than positive is to maintain the convention that function values inside the surface should be less than ℓ, while values outside the surface should be greater than ℓ. This leads to a convenient binary outside/inside test for points x:

$$F(\mathbf{x}, \mathbf{\Theta}) > \ell \tag{4}$$

For most experiments presented here, we use $N = 100$. Because each shape element has seven parameters, the total dimensionality of our representation is a fixed $7N = 700$ floating point values.

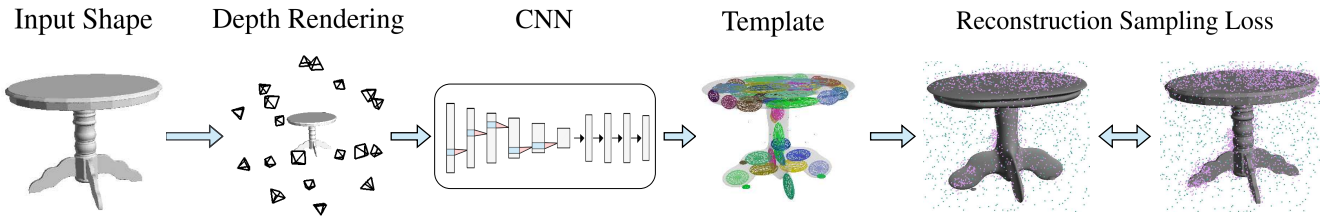

Figure 3. An overview of our method. The input to our system is a mesh. We render a stack of depth images around the mesh, and provide these as input to an early-fusion CNN. The output of the CNN is a vector with fixed dimensionality. This vector is interpreted as a shape template with parameters that define an implicit surface. Next, we sample points near the ground truth surface and also uniformly in space. A classification loss enforces that each sample point is correctly labeled as inside/outside by the surface reconstruction.

4. Template Learning

We propose a learning framework (Figure 3) to train a neural network to fit the shape template to data. The network's goal is to find the template parameters $\mathbf{\Theta}$ that best fit a 3D shape, where the loss penalizes the amount of predicted shape that is on the wrong side of the ground truth inside/outside border. We render multiple depth images of the mesh from fixed views to provide 3D input to the network. Our network has a feed-forward CNN architecture and predicts the entire parameter vector $\mathbf{\Theta}$ at once with a fully connected layer. During training, we choose sparse sample locations in 3D and evaluate our loss function at those locations with a classification loss. The details of this procedure are described in the rest of this section.

Note that although fitting consistency is vital to our applications, we do not directly enforce similar shapes to have similar template parameters; the network arrives at a smooth fitting function without intervention. We hypothesize that, as a matter of optimization, the smooth solution is "easier" for the network to learn, but analyzing the causes of this behavior is an engaging direction for future work.

4.1. Architecture

In order to learn the template, we first need to encode the input 3D shape. There are a variety of network architectures for encoding 3D shape; options include point networks [40], voxel encoders [31], or multi-view networks [48]. Because voxel encoders can be computationally expensive, and point cloud encoders discard surface information, we opt for a multi-view encoding network. Our network takes a stack of 20 depth images rendered from the vertices of a dodecahedron as input, as in [22]. The network contains 5 convolutional layers followed by 4 fully connected layers.

The final fully connected layer is linear and maps to the template parameter vector $\mathbf{\Theta}$, which in our experiments is usually 700-D. Even though we use an encoder/decoder style architecture, there is no heavy decoding stage: the code vector is our explicit representation. We experimented with alternative "decoding" architectures, such as an LSTM that predicts each shape element in succession. We found the LSTM architecture to perform better in some cases, but it took much longer to train, and was not able to scale easily to large numbers of shape elements.

4.2. Data Preparation

Before training, we must preprocess the input meshes to make them watertight. This step is important primarily because our loss function requires a ground truth inside/outside classification label.

In order to do the watertight conversion, we first convert the meshes to a 300^3 sparse voxel representation [35]. We flood fill the octree to determine inside/outside, then extract the isocontour of the volume to produce the watertight mesh. We generate 100,000 random samples uniformly in the bounding box of the mesh, and compute 0/1 inside/outside labels. We additionally compute 100,000 samples evenly distributed on the surface of the mesh.

We also render depth maps of the watertight meshes. For each mesh, we render 20 depth images at uniformly sampled viewing directions as input to the network. The (depth maps, labeled samples) pairs are the only data used for learning.

4.3. Loss

The goal of our loss function is only to measure deviation from the input shape; we assume that our representation will naturally create a smooth template due to its structure. In order to accurately reconstruct the surface, we employ three individual loss functions, described in detail in the following sections. L_U and L_S are classification losses ensuring that the volume around the ground truth shape is correctly classified as inside/outside. These losses were inspired by recent work on implicit function learning [32, 9]. L_C enforces that all of the shape elements contribute to the reconstruction. The total loss function is a weighted combination of the three losses:

$$L = w_U L_U + w_S L_S + L_C \tag{5}$$

L_C has no weight here because it contains two subclasses with different weights w_a and w_b.

As our losses compare the structured implicit value $F(\mathbf{x}, \mathbf{\Theta})$ to indicator function labels (0 inside, 1 outside), we formulate a soft classification boundary function to better facilitate gradient learning:

$$G(\mathbf{x}, \mathbf{\Theta}) = \text{Sigmoid}\left(\alpha(F(\mathbf{x}, \mathbf{\Theta}) - \ell)\right) \tag{6}$$

where α controls the sharpness of the boundary, and is set to 100 as determined by grid search.

4.3.1 Uniform Sample Loss L_U

If $F(\mathbf{x}, \mathbf{\Theta})$ correctly classifies every point in the volume according to the ground truth shape boundary, then it has perfectly reconstructed the ground truth. To measure the classification accuracy, we choose (x, y, z) coordinates uniformly at random in the bounding box of the ground truth mesh. We evaluate $F(\mathbf{x}, \mathbf{\Theta})$ at these locations, and apply a loss between the softened classification decision G, and the ground truth class label, which is 0 inside and 1 outside:

$$L_U(\mathbf{x}, \mathbf{\Theta}) = \begin{cases} \beta G(\mathbf{x}, \mathbf{\Theta})^2 & \mathbf{x} \text{ inside} \\ (1 - G(\mathbf{x}, \mathbf{\Theta}))^2 & \mathbf{x} \text{ outside} \end{cases} \tag{7}$$

At each training batch we randomly select 3,000 of the precomputed 100,000 points to evaluate the loss. β accounts for the inside/outside sample count differences.

4.3.2 Near Surface Sample Loss L_S

While the uniform sample loss is effective, it is problematic because it prioritizes surface reconstruction based on the fraction of the volume that is correct. The network can easily achieve 99%+ correct volume samples and still not visually match the observation. In particular, thin structures are unimportant to a volumetric loss but subjectively important to the reconstruction. To improve performance, we sample proportionally to surface area, not volume. We additionally want to ensure that the network is not biased to produce an offset surface, so the loss should be applied with similar weight on both the positive and negative side of the surface boundary.

In order to achieve these goals, we implemented the following algorithm. For each of the 100,000 surface samples, a ray is cast in each of the positive and negative normal directions away from the surface point. Because the mesh is watertight, at least one of the two samples must intersect the surface. The minimum of these two intersection distances is chosen, and truncated to some threshold. We sample a point along either normal direction with probability inversely proportional to the squared distance from the surface and proportional to the minimum intersection distance. The output samples roughly satisfy both of our goals: no thin structures are missed, regardless of their volume, and there is an equal sampling density on both sides of the surface.

This loss function, L_S, is identical to L_U (see Equation 7) except for the sample locations where it is applied. Note that L_S and L_U are not redundant with one another. Because L_S only contains samples very near the surface, it does not on its own enforce that the network keep free space clear of spurious shapes. We found it most effective to use a weighted combination of both losses, using L_S to do hard example mining, and L_U to ensure that free space around the shape remains clear.

4.3.3 Shape Element Center Losses L_C

One problem with the loss so far is that it is only concerned with the final composite function $F(\mathbf{x}, \mathbf{\Theta})$. If shape elements do not affect F, they also don't affect the loss. This "death" of shape elements can easily happen over time, since elements are randomly initialized and some are likely to be far from the ground truth surface. Their contribution to L_U and L_S is small, and there is no incentive for the network to use them. Our solution to this problem is to apply a third loss L_C, the center classification loss. This loss enforces that all predicted centers must lie on the inside of the predicted shape and within the ground truth bounding box:

$$L_C(\mathbf{x}, \mathbf{\Theta}) = \begin{cases} w_a G(\mathbf{x}, \mathbf{\Theta})^2 & \mathbf{x} \in B \\ w_b \sum_d \max(0, B_L - \mathbf{x}_d, B_U - \mathbf{x}_d)^2 & \mathbf{x} \notin B \end{cases} \tag{8}$$

Above, w_a and w_b are hyperparameters balancing the two cases, which are in different units. B is the axis aligned bounding box of the ground truth shape, which has a lower coordinate B_L and an upper coordinate B_U. It states that if the predicted center $\mathbf{x}$ is inside the ground truth bounding volume (where L_U will be applied, keeping free space empty), then $\mathbf{x}$ must also be inside the predicted surface. On the other hand, if $\mathbf{x}$ is outside the ground truth bounding boxing, then it should be directly encouraged to move inside the bounding volume because it can't be useful to the template from that distance.

5. Experiments

We conduct experiments to demonstrate important properties of the shape template: it accurately fits a wide variety of shapes, fits similar shapes with similar templates, can be used to find 3D-to-3D and 2D-to-3D correspondences, and can be fit from RGB images alone. We train and test on ShapeNet Core V2 [8], using the dataset split defined by 3D-R^{2}N^2 [10]. We show results trained on both the full dataset (Sections 5.1, 5.3, 5.4) and trained per-class (Section 5.2). Identical hyperparameters were used to train all templates.

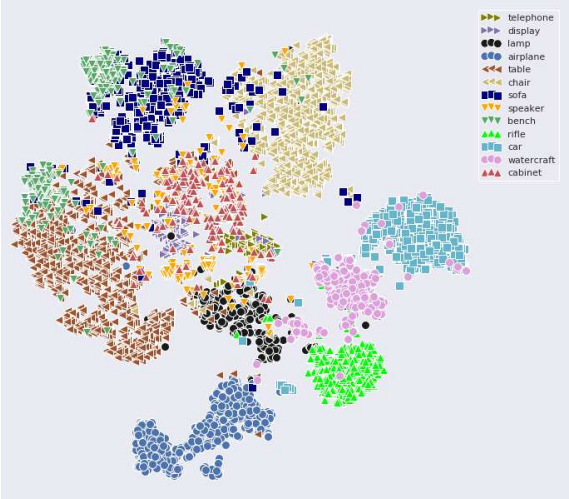

Figure 4. t-SNE visualization of template parameters on ShapeNet test set, colored by shape class labels. Note the clean clustering of most classes. Mixed clusters are also intuitive, e.g. mixing between tables, benches, and sofas.

5.1. Clustering by Template Parameters

A desirable property of a template fitting procedure is that similar shapes are fit with similar template parameters. Figure 4 shows a t-SNE [51] visualization of the template parameter vectors Θ for the ShapeNet test set, colored by ShapeNet class labels. Several classes of shapes (airplanes, rifles, cars) are neatly clustered by their template parameters. Other classes are mixed, but in intuitive ways: some benches look like tables, other benches look like sofas, and some sofas look like chairs. Cabinets, speakers, and displays are all essentially boxes, so they have similar template parameters.

5.2. Comparison to Volumetric Primitives

The closest alternative approach to ours is the volumetric primitives of Tulsiani, et al. [50]. We provide a detailed comparison between our template shapes and their shape abstractions using results generously provided by the authors. For this comparison we trained one fitting network per shape class, not one network for all classes, to match the procedure of [50]. Figure 5 shows representative results for examples from the ShapeNet training set, with 10, 25, and 100 shape elements (see supplemental material for the full set of results). In comparison to volumetric primitives (Figure 5 a), our templates (b-d) are more detailed, have higher consistency, and better reflect the structure of the input mesh (f).

5.3. Single-View RGB Prediction and Labeling

Figure 6 shows qualitative results demonstrating predictions from photographs of ShapeNet-style objects. To predict the template parameters from an RGB image, we apply

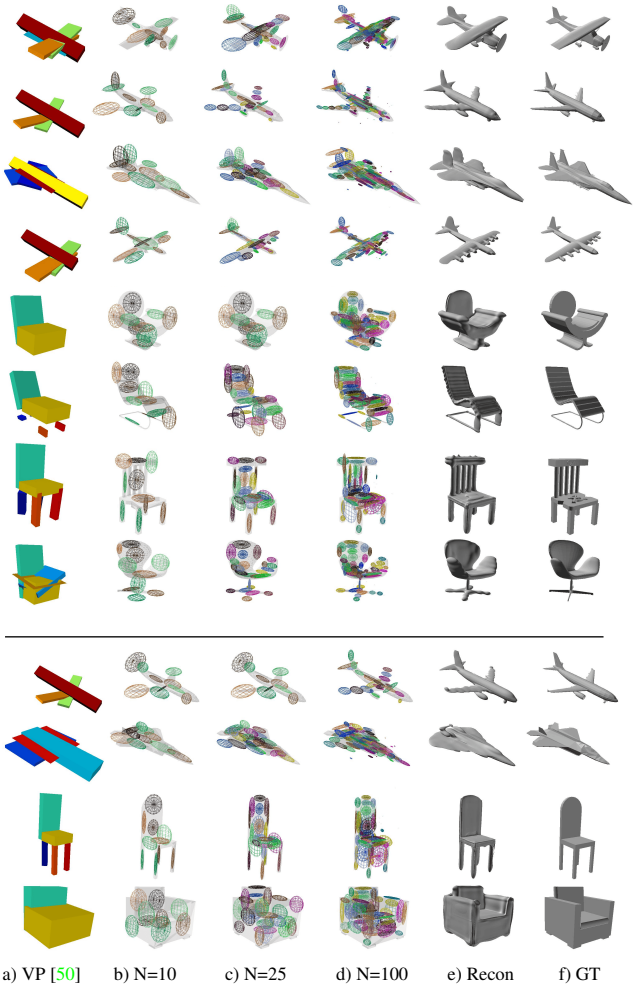

Figure 5. Comparison to Volumetric Primitives [50]: (a) volumetric primitives result; (b-d) templates computed with our method for 10, 25, and 100 elements; (e) surface reconstruction from the template in (d); (f) ground truth surface mesh. Shapes above the line come from our training set, while the shapes below the line are from our test set.

a similar technique to CNN purification [28] or network distillation [16] and train a second network that regresses from RGB to the template parameters already found through our 3D-to-3D training scheme. The training data for this network is synthetic OpenGL renderings of the ShapeNet training set, with camera angles chosen randomly from a band around the equator of the shape.

Because the template is consistent, we can go further than overall 3D shape prediction and predict correspondence between pixels in the image and the influence regions of individual shape elements (Figure 6, right). Each element tends to produce a particular part of each shape: the i^{th} element might produce the tail fin of an airplane, while the j^{th} might produce the wingtip. Because of this consistency,

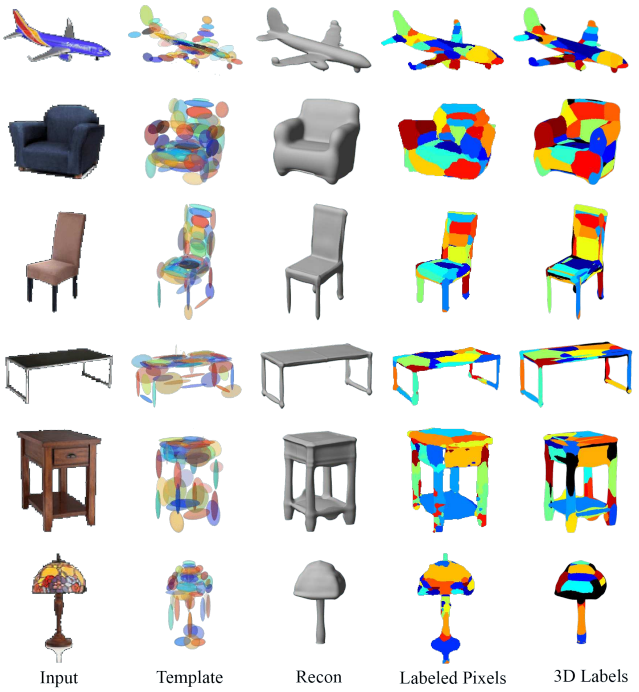

Figure 6. Template fitting and labeling from photographs. From left to right: input image with background removed, fit template, corresponding isosurface, image pixels labeled by the highest-value shape element, corresponding 3D regions labeled by the highest-value element. Regions in 3D not found by the image labeling network are black. The labeling performs well for easily oriented shapes (top rows), and worse for shapes with rotational symmetries (bottom rows). Note that the labeling is based entirely on the template, without additional region or part labels.

a semantic segmentation network [44] can be trained to label pixels by the index of the shape element with maximum weight at that pixel. The result is a segmentation of the image into 3D regions, without additional region or part labels. One limitation of this approach is that the template learning does not take into account object symmetry. Shapes with natural orientations, such as airplanes and chairs, are successful, while shapes without fronts and backs, such as the lamp and nightstand, confuse the network.

Similar techniques have been used for human body pose prediction [53, 5] using hand-made templates, but to our knowledge, we are the first to use a learned template.

5.4. Shape Correspondence

The learned template is consistent across shapes of the same class, meaning that the same elements will influence equivalent shape parts (e.g. airplane wings). This property can be exploited to find correspondences between different shapes. We present one automatic approach to achieve that. First, we use our network to compute the template configuration $\mathbf{\Theta}$ of each shape we want to correspond. Then,

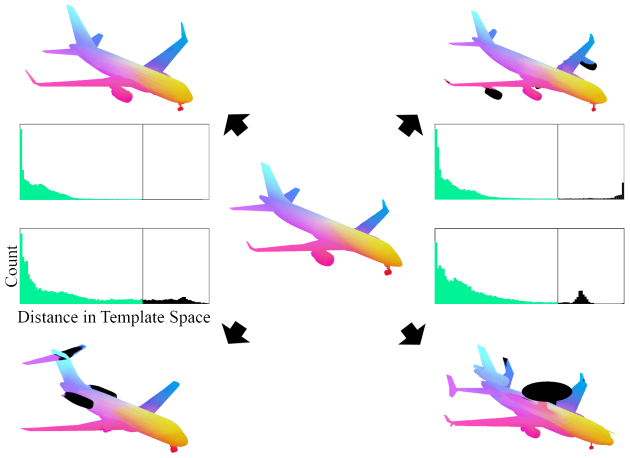

Figure 7. Transferring per-vertex colors from source airplane (center) to target airplanes (corners). Vertices are corresponded to their nearest neighbor in template space. Matching colors indicate corresponding vertices, while black regions have no corresponding vertices in the source. The histograms plot the proportion of nearest-neighbor distances that produce good matches (green) and outliers (black, distance > 0.65). Outliers include extra wing and tail engines, landing gear, and a radar dome, all missing on the source airplane. Correspondences were computed for resampled ShapeNet meshes from the training set of the multi-class network.

for each vertex $\mathbf{v}$, we compute its template coordinates. The template coordinates consist of three numbers for each shape element. Those are computed by subtracting the element's center from the vertex position, dividing each coordinate by the corresponding element radius (improving correspondence between elongated and squashed elements), then scaling that vector to be of length $F(\mathbf{v}, \mathbf{\Theta})$. The direction of each per-element vector helps geometrically localize the vertex, while its length denotes the influence of that element. Finally, the cosine distance between template coordinates can be used to find the closest target vertex for each source vertex, as visualized in Figure 7.

5.5. Human Scans

The method generalizes beyond the synthetic objects in the ShapeNet [8] dataset. In Figure 8 we show fits to BodyShapes [39] meshes from the CAESAR dataset [43]. This dataset contains fits to real scans of approximately 3,000 humans. We split the data into train (85%), validation (5%), and test (10%) splits and show results from the test split. Please note the consistency of the template fits.

5.6. RGB Single View 3D Reconstruction

While exact shape reconstruction is not the focus of our work, we compared the reconstruction accuracy of the template surface with the output of 3D-R^{2}N^2 [10], Point Set Generation Network [11], and Pixel2Mesh [52]. The inputs are single RGB images of unknown camera orienta-

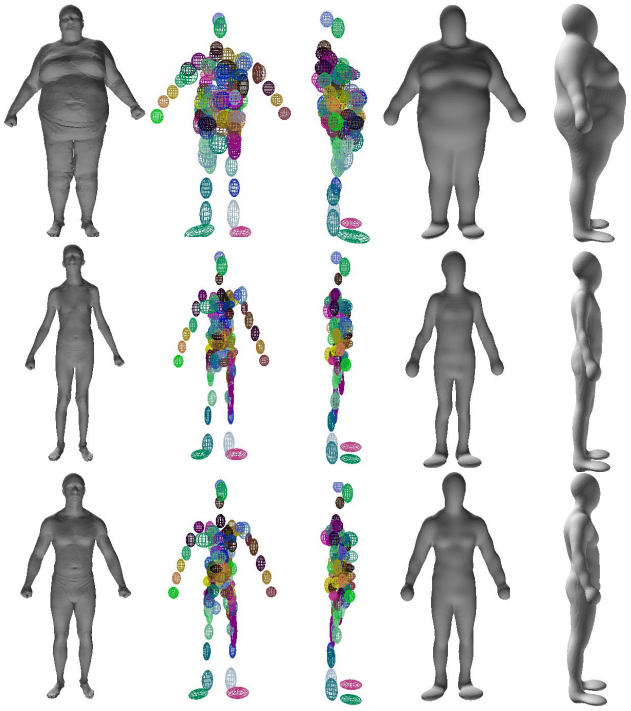

Figure 8. Results on the MPII BodyShapes [39] meshes from the CAESAR dataset [43]. We demonstrate correspondence on real scans of humans, indicating the method's ability to generalize beyond ShapeNet [8].

Threshold	τ				2τ			
Category	R2N2	PSG	P2M	Our	R2N2	PSG	P2M	Our
plane	41	68	**71**	69	63	81	81	**86**
bench	34	49	58	**62**	49	69	72	**82**
cabinet	50	40	**60**	40	65	67	**77**	64
car	38	51	**68**	47	55	78	**84**	70
chair	40	42	**54**	40	55	64	**70**	64
monitor	34	40	**51**	42	48	64	**67**	65
lamp	32	41	**48**	32	44	59	**62**	52
speaker	45	32	**49**	29	58	57	**66**	50
firearm	28	70	**73**	72	47	83	83	**88**
sofa	40	37	**52**	42	53	63	**70**	**70**
table	44	53	**66**	40	59	73	**79**	61
cellphone	42	56	**70**	56	61	80	**83**	79
watercraft	37	51	**55**	49	52	71	70	**75**
mean	39	49	**60**	48	55	70	**74**	70

Table 2. F-score (%) on the test split of ShapeNet from [10], with $\tau = 10^{-4}$ as in [52]. Higher numbers are better. R2N2 is 3D-R^{2}N^2 [10], PSG is Point-Set Generation Network [11], and P2M is Pixel2Mesh [52].

tion, so we use the distillation approach from Section 5.3. The train/test split is from 3D-R^{2}N^2. Our shape representation has only 700 degrees of freedom, compared with $32^3 = 32768$ DoF for the 3D-R^{2}N^2 grid, $1024 * 3 = 3072$ DoF for PSG's points, and $2466 * 3 = 7398$ DoF for the Pixel2Mesh vertices. Despite having many fewer degrees of freedom, the template surface reconstruction accuracy is similar to competing approaches (Table 2).

a) Template fit b) Reconstruction c) Input mesh

Figure 9. Shapes with angled parts, sharp creases, and thin structures are difficult for our method to learn.

5.7. Limitations

Our method has several limitations apparent in Figure 9, which exhibits several failure cases. First, since our representation comprises of a small number of axis-aligned functions, it has limited ability to represent detailed, sharp, or angled structures (e.g., creases or corners). Second, since it learns to classify sides of a surface boundary, it struggles to reconstruct razor thin structures. Finally, since it uses a fixed number of shape elements (e.g., 100), it does not produce a template with 1-to-1 mapping to semantic shape components. We believe these limitations could be addressed with alternative (higher-order, non axis-aligned) local functions, distance-based loss functions, supervised training, and/or network architecture search.

6. Conclusion

This paper investigates using structured implicit functions to learn a template for a diverse collection of 3D shapes. We find that an encoder-decoder network trained to generate shape elements learns a template that maps detailed surface geometry consistently across related shapes in a collection with large shape variations. Applications for the learned template include shape clustering, exploration, abstraction, correspondence, interpolation, and image segmentation. Topics for future work include learning to generate higher-order and/or learned shape elements, deriving semantically meaningful shape elements via supervised learning, and using structured implicit functions for other applications such as 3D reconstruction.

7. Acknowledgements

We acknowledge ShapeNet [8], 3D-R^{2}N^2 [10], MPII BodyShapes [39], and Stanford Online Products [47] for providing training data for our method. We also thank the authors of Volumetric Primitives [50] for providing extended results from their method for our comparisons. We thank Avneesh Sud for helpful discussions and comments.

References

[1] Nikita Araslanov, Seongyong Koo, Juergen Gall, and Sven Behnke. Efficient single-view 3d co-segmentation using shape similarity and spatial part relations. In *German Conference on Pattern Recognition*, pages 297–308. Springer, 2016. 2

[2] Irving Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115, 1987. 2

[3] James F Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics (TOG)*, 1(3):235–256, 1982. 1, 2

[4] Jules Bloomenthal and Ken Shoemake. Convolution surfaces. *SIGGRAPH 1991*, 25(4):251–256, 1991. 1, 3

[5] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J. Black. Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image. In *European Conference on Computer Vision (ECCV)*, Lecture Notes in Computer Science. Springer International Publishing, Oct. 2016. 1, 7

[6] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Generative and discriminative voxel modeling with convolutional neural networks. *arXiv:1608.04236*, 2016. 3

[7] Roberto Brunelli. *Template Matching Techniques in Computer Vision: Theory and Practice*. John Wiley & Sons, 2009. 2

[8] 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, Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 1, 5, 7, 8

[9] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5939–5948, 2019. 4

[10] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European Conference on Computer Vision (ECCV)*, 2016. 5, 7, 8

[11] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 605–613, 2017. 3, 7, 8

[12] Noa Fish, Melinos Averkiou, Oliver Van Kaick, Olga Sorkine-Hornung, Daniel Cohen-Or, and Niloy J Mitra. Meta-representation of shape families. *ACM Transactions on Graphics (TOG)*, 33(4):34, 2014. 2

[13] Vignesh Ganapathi-Subramanian, Olga Diamanti, Soeren Pirk, Chengcheng Tang, Matthias Nießner, and Leonidas Guibas. Parsing geometry using structure-aware shape templates. In *2018 International Conference on 3D Vision (3DV)*, pages 672–681. IEEE, 2018. 1, 2

[14] Aleksey Golovinskiy and Thomas Funkhouser. Consistent segmentation of 3d models. *Computers & Graphics*, 33(3):262–269, 2009. 2

[15] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 216–224, 2018. 3

[16] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. 6

[17] Ruizhen Hu, Manolis Savva, and Oliver van Kaick. Functionality representations and applications for shape analysis. In *Computer Graphics Forum*, volume 37, pages 603–624. Wiley Online Library, 2018. 2

[18] Ruizhen Hu, Oliver van Kaick, Youyi Zheng, and Manolis Savva. Siggraph asia 2016: course notes directions in shape analysis towards functionality. In *SIGGRAPH Asia 2016 Courses*, page 8. ACM, 2016. 2

[19] Adrien Kaiser, Jose Alonso Ybanez Zepeda, and Tamy Boubekeur. A survey of simple geometric primitives detection methods for captured 3d data. In *Computer Graphics Forum*. Wiley Online Library, 2018. 2

[20] Evangelos Kalogerakis, Aaron Hertzmann, and Karan Singh. Learning 3d mesh segmentation and labeling. *ACM Transactions on Graphics (TOG)*, 29(4):102, 2010. 2

[21] Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *European Conference on Computer Vision (ECCV)*, pages 371–386, 2018. 3

[22] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5010–5019, 2018. 4

[23] Vladimir G Kim, Wilmot Li, Niloy J Mitra, Siddhartha Chaudhuri, Stephen DiVerdi, and Thomas Funkhouser. Learning part-based templates from large collections of 3d shapes. *ACM Transactions on Graphics (TOG)*, 32(4):70, 2013. 2

[24] Hamid Laga, Yulan Guo, Hedi Tabia, Robert B Fisher, and Mohammed Bennamoun. *3D Shape Analysis: Fundamentals, Theory, and Applications*. John Wiley & Sons, 2018. 2

[25] Vincent Léon, Vincent Itier, Nicolas Bonneel, Guillaume Lavoué, and Jean-Philippe Vandeborre. Semantic correspondence across 3d models for example-based modeling. In *Eurographics Workshop on 3D Object Retrieval 2017 (3DOR 2017)*, 2017. 2

[26] Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. Grass: Generative recursive autoencoders for shape structures. *ACM Transactions on Graphics (TOG)*, 36(4):52, 2017. 2

[27] Lingxiao Li, Minhyuk Sung, Anastasia Dubrovina, Li Yi, and Leonidas J Guibas. Supervised fitting of geometric primitives to 3d point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2652–2660, 2019. 2

[28] Yangyan Li, Hao Su, Charles Ruizhongtai Qi, Noa Fish, Daniel Cohen-Or, and Leonidas J. Guibas. Joint embeddings

of shapes and images via cnn image purification. *ACM Trans. Graph.*, 34(6):234:1–234:12, Oct. 2015. 6

[29] Yangyan Li, Xiaokun Wu, Yiorgos Chrysathou, Andrei Sharf, Daniel Cohen-Or, and Niloy J Mitra. Globfit: Consistently fitting primitives by discovering global relations. *ACM Transactions on Graphics (TOG)*, 30(4):52, 2011. 2

[30] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH 1987, pages 163–169, New York, NY, USA, 1987. ACM. 2

[31] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015. 4

[32] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3, 4

[33] Niloy J Mitra, Michael Wand, Hao Zhang, Daniel Cohen-Or, Vladimir Kim, and Qi-Xing Huang. Structure-aware shape processing. In *SIGGRAPH 2014 Courses*, page 13. ACM, 2014. 2

[34] Shigeru Muraki. Volumetric shape description of range data using blobby model. *SIGGRAPH 1991*, 25(4):227–235, 1991. 2

[35] Ken Museth. Vdb: High-resolution sparse volumes with dynamic topology. *ACM Trans. Graph.*, 32(3):27:1–27:22, July 2013. 4

[36] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. *Multi-level partition of unity implicits*, volume 22. ACM, 2003. 1, 3

[37] Maks Ovsjanikov, Wilmot Li, Leonidas Guibas, and Niloy J Mitra. Exploration of continuous variability in collections of 3d shapes. *ACM Transactions on Graphics (TOG)*, 30(4):33, 2011. 2

[38] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 165–174, 2019. 3

[39] Leonid Pishchulin, Stefanie Wuhrer, Thomas Helten, Christian Theobalt, and Bernt Schiele. Building statistical shape spaces for 3d human modeling. *Pattern Recognition*, 2017. 7, 8

[40] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017. 4

[41] Antonio Ricci. A constructive geometry for computer graphics. *The Computer Journal*, 16(2):157–160, 1973. 2

[42] Lawrence Roberts. *Machine Perception of Three-Dimensional Solids*. 01 1963. 1, 2

[43] Kathleen M Robinette, Hans Daanen, and Eric Paquet. The caesar project: a 3-d surface anthropometry survey. In *Second International Conference on 3-D Digital Imaging and Modeling (Cat. No. PR00062)*, pages 380–386. IEEE, 1999. 7, 8

[44] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015. 7

[45] Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. Csgnet: Neural shape parser for constructive solid geometry. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5515–5523, 2018. 2

[46] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. Deepvoxels: Learning persistent 3d feature embeddings. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2437–2446, 2019. 3

[47] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 8

[48] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 4

[49] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2088–2096, 2017. 3

[50] Shubham Tulsiani, Hao Su, Leonidas J Guibas, Alexei A Efros, and Jitendra Malik. Learning shape abstractions by assembling volumetric primitives. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2635–2643, 2017. 2, 3, 6, 8

[51] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008. 1, 6

[52] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *European Conference on Computer Vision (ECCV)*, pages 52–67, 2018. 3, 7, 8

[53] Lingyu Wei, Qixing Huang, Duygu Ceylan, Etienne Vouga, and Hao Li. Dense human body correspondences using convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 7

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

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

[56] Geoff Wyvill, Craig McPheeters, and Brian Wyvill. Data structure for soft objects. In *Advanced Computer Graphics*, pages 113–128. Springer, 1986. 1, 2

[57] Kai Xu, Vladimir G Kim, Qixing Huang, and Evangelos Kalogerakis. Data-driven shape analysis and processing. In *Computer Graphics Forum*, volume 36, pages 101–132. Wiley Online Library, 2017. 2

[58] Mehmet Ersin Yumer and Levent Burak Kara. Co-abstraction of shape collections. *ACM Transactions on Graphics (TOG)*, 31(6):166, 2012. 2

[59] Youyi Zheng, Daniel Cohen-Or, Melinos Averkiou, and Niloy J Mitra. Recurring part arrangements in shape collections. In *Computer Graphics Forum*, volume 33, pages 115–124. Wiley Online Library, 2014. 2

[60] Chuhang Zou, Ersin Yumer, Jimei Yang, Duygu Ceylan, and Derek Hoiem. 3d-prnn: Generating shape primitives with recurrent neural networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 900–909, 2017. 2