Contour-based 3D Modeling through Joint Embedding of Shapes and Contours

Aobo Jin
University of Houston, USA
ajin4@uh.edu

Qiang Fu*
Beijing University of Post and Communications, China

Zhigang Deng
University of Houston, USA
zdeng4@uh.edu

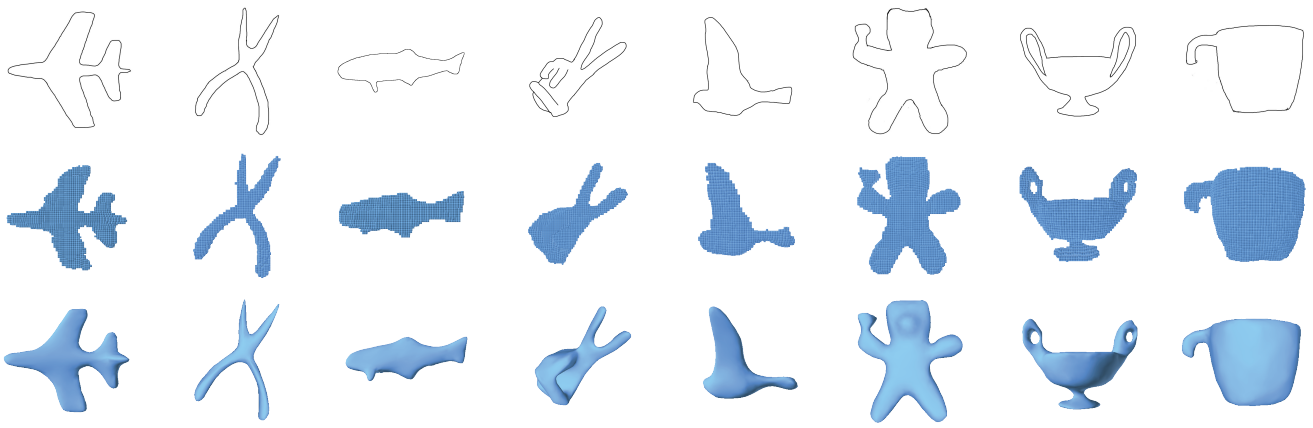

Figure 1: Gallery of modeling results. With a sketch contour input (top), our method can automatically generate its corresponding 3D model with voxel representation (middle) and then convert and refine it to its 3D mesh representation (bottom).

ABSTRACT

In this paper, we propose a novel space that jointly embeds both 2D occluding contours and 3D shapes via a variational autoencoder (VAE) and a volumetric autoencoder. Given a dataset of 3D shapes, we extract their occluding contours via projections from random views and use the occluding contours to train the VAE. Then, the obtained continuous embedding space, where each point is a latent vector that represents an occluding contour, can be used to measure the similarity between occluding contours. After that, the volumetric autoencoder is trained to first map 3D shapes onto the embedding space through a supervised learning process and then decode the merged latent vectors of three occluding contours (from three different views) of a 3D shape to its 3D voxel representation. We conduct various experiments and comparisons to demonstrate the usefulness and effectiveness of our method for sketch-based 3D modeling and shape manipulation applications.

*Most of Qiang Fu's involvement on this work was done while he worked at University of Houston as a PostDoc.

I3D '20, May 5–7, 2020, San Francisco, CA, USA

ACM ISBN 978-1-4503-7589-4/20/05…$15.00
https://doi.org/10.1145/3384382.3384518

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

deep learning, geometry modeling, sketch-based modeling

ACM Reference Format:
Aobo Jin, Qiang Fu, and Zhigang Deng. 2020. Contour-based 3D Modeling through Joint Embedding of Shapes and Contours. In *Symposium on Interactive 3D Graphics and Games (I3D '20), May 5–7, 2020, San Francisco, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3384382.3384518

1 INTRODUCTION

As an intuitive method of creating 3D models, sketch-based modeling has been widely studied during the past several decades. A variety of methods and systems have been developed to explore multiple conceptual options, without finalizing geometric details in the early design stage [Bae et al. 2008; Kara and Shimada 2007, 2008; Shao et al. 2013]. However, since at the early stage modelers may lack a clear mental image on the target shape, current methods that typically require user sketches with abundant lines, might decelerate the modeling effectiveness. For example, to model a 3D shape from a single view sketch, besides simple occluding contours, the modelers need to have extra information, such as supportive lines [Li et al. 2017] or lines extracted from images [Chen et al. 2013] to provide 3D surface curvature information and other details. Therefore, efficient shape generation solely based on simple occluding contours is still considered a wide open problem to date.

Inspired by the above problem, we present a new method by jointly embedding 2D occluding contours and 3D shapes into a space via a variational autoencoder (VAE) and a volumetric autoencoder. In the space, the similarities of both 3D shapes and 2D occluding contours can be measured by the distances of their corresponding latent vector representations. Specifically, we extract multiple occluding contours from a number of views for each 3D object in a 3D shape dataset by employing a well-known, non-photorealistic rendering algorithm [DeCarlo et al. 2003]. These occluding contours are then used as the training data to train a variational autoencoder, which can map an occluding contour to a point in the embedding space. The coordinates of each point are a latent vector that not only can measure the similarity of the occluding contours but also can be decoded to the original occluding contour representation. Moreover, when the viewpoint of a 3D shape are determined, the 3D shape also has its coordinates in the embedding space, based on its projected occluding contour.

We then consider the problem of mapping the geometry of 3D shapes to this embedding space. In other words, we expect the latent vectors in the embedding space can also represent 3D geometry, and can even be decoded to recover corresponding 3D models. Since the occluding contours of a 3D shape from three orthogonal views (i.e., front, top, and side views) can typically capture its geometry, we expect the vector encoded by the volumetric autoencoder could consist of the latent vectors of the three occluding contours of the 3D shape. To this end, we employ a supervised learning framework to train a volumetric autoencoder for the voxel representation of 3D shapes. Therefore, in the embedding space, for any group of latent vectors that respectively represent the occluding contours of the three different, orthogonal views of a 3D shape, we can concatenate them into a new vector and further decode it to the voxel representation of the underlying 3D shape. As shown in Figure 2, once we update an occluding contour to a new one by changing the associated fragment(s) in the vector, a new 3D shape with the new occluding contour can be generated through the volumetric autoencoder. This is mainly because the contours from the other two views provide some additional information to assist the changed contour to form its 3D shape.

Based on the proposed embedding space, we demonstrate its selected applications on 3D modeling and shape manipulation. In the 3D modeling application, given a 2D occluding contour drawn manually or extracted from an image, the trained VAE is first used to encode it to a latent vector in the embedding space. We then search for its neighbors in the embedding space that represent the occluding contours with similar shapes. Since these neighbors are actually generated from 3D shapes, we can also find the other two orthogonal contour views of the neighbors as the missing contour views to supplement the input single occluding contour. To this end, just like sketch-based modeling, our method can generate a 3D shape with the vector consists of the three latent vectors that correspond to the input occluding contour and the other two supplemented contours. In the shape manipulation application, given a 3D model, we can perform shape manipulations on it using the similar occluding contours in the embedding space to replace certain contour views of the given model to generate 3D shape variations. The manipulation is only focused on the outline of a 3D shape instead of its surface details. Finally, we demonstrate the effectiveness of our method through various experiments and comparisons with the state-of-the-art, sketch-based 3D modeling methods [Li et al. 2018; Xie et al. 2013]. Figure 1 shows some results generated by our method.

In sum, our method makes the following two major contributions: 1) a new framework to form an embedding space of 3D shapes and 2D occluding contours via a variational autoencoder and a volumetric autoencoder; and 2) two selected applications of the embedding space for sketch-based 3D modeling and shape manipulations.

2 RELATED WORK

In this section we give a categorized review of the recent related literature, including the joint embedding of shapes and images, sketch-based 3D modeling, and learning-based shape synthesis, especially deep neural network based methods. Readers are referred to [Cook and Agah 2009; Ding and Liu 2016; Kazmi et al. 2014; Olsen et al. 2009] for comprehensive surveys on sketch-based modeling.

Joint embedding spaces. Joint embedding methods have been employed for classification and retrieval tasks. To retrieve 3D shapes by 2D images or retrieve 2D images by 3D shapes, Li et al. [2015] proposed a joint embedding space for images and 3D shapes, which can bridge the gap between 3D shapes and images. Instead of linking 2D and 3D spaces, Tasse et al. [2016] build semantic-based descriptors for 3D shapes, sketches, and images by embedding 3D shapes, sketches, and images into a word vector space to handle diverse retrieval tasks. It is difficult to directly extend their method for our work, since we need a continuous embedding space to handle shape generation and manipulation tasks. Embedding spaces can also be utilized to analyze arrangements and shapes of parts across models [Averkiou et al. 2014]. To easily present high-dimensional shapes in a low-dimensional space, embedding spaces have also been widely investigated for the shape modeling of both human bodies [Anguelov et al. 2005; Loper et al. 2015] and faces [Blanz et al. 1999]. In our work, the embedding space is constructed by the latent vectors of 2D contours. Then, we further learn a volumetric autoencoder to project 3D shapes onto the embedding space. This process is similar to the work of [Li et al. 2015], where a space with 3D shapes and their mapped 2D images is constructed. Therefore, we borrow the term of "joint embedding space" for our method.

Sketch-based modeling. Hand-drawn sketches, as an intuitive input, have been widely utilized for interactive 3D modeling [Fu et al. 2016; Gingold et al. 2009; Huang et al. 2017; Xie et al. 2013; Zeleznik et al. 2007]. Some existing methods [Igarashi et al. 2007; Nealen et al. 2007] have been focused on using simple lines to generate a 3D shape based on local geometric properties implied by lines. They typically also support editing operations such as cutting, growing, and blending. However, these methods can only generate simple models and are less effective for organic shapes with smoothly varying surfaces. Later, researchers introduced methods to generate free-form surfaces by exploiting geometric constraints in specific types of line drawings [Karpenko and Hughes 2006; Xu et al. 2014]. Fan et al. [2013] utilize shadow guidance to guide the sketching part and further search for a nearest shape from a dataset to combine a new 3D shape. Since multi-view sketches can convey depth information directly, River et al. [2010] introduced a two-view

sketch-based modeling system. Recently, Li et al. [2017] proposed a sketch-based 3D modeling method, where some support lines are employed to indicate the surface curvatures of 3D models and predict the curvature field. To generate occluded parts, multi-view interaction is supported in their method. Even though the above geometric methods achieve certain successes on the generation of 3D shapes, their user input still requires non-trivial prior knowledge, instead of drawing simple occluding contours. In addition, the concept of sketch-based modeling has also been extended for creating 3D facial expressions [Sucontphunt et al. 2008] and various man-made objects [Chen et al. 2013; Xu et al. 2016].

Learning-based shape synthesis. Learning-based shape synthesis methods aim to learn geometric structure from 3D datasets in order to extract the features of 3D shapes or even generate plausible 3D shapes. Several methods were designed to predict the depth image and norms from a single image [Eigen and Fergus 2015; Hoiem et al. 2005; Saxena et al. 2008; Wang et al. 2015] through supervised learning. Human face reconstruction through images has been well studied [Richardson et al. 2017; Tran et al. 2018; Winkler et al. 2018]. Gkioxari et al. [2019] proposed a method to predict the mesh from an image. More complex shape reconstruction tasks have attracted a lot of attentions in recent years. For example, Tatarchenko et al. [2016] utilize the encoder-decoder architecture of convolutional network (ConvNet) to predict multiple views of a given object in 2D space. Later, Lun et al. [2017] extend the idea and use the U-net architecture [Ronneberger et al. 2015] to generate 3D shapes with the front and side view sketches. However, the two views of sketches are still difficult to draw since there still exist some shared information between the two views such as the height of a standing character. A single-view sketch-based modeling method [Li et al. 2018] was also proposed to predict the curvature field with the ConvNet and U-Net architecture. This method requires support lines for surface curvatures and needs multi-view interactions to generate full 3D shapes. By contrast, our work only requires a single-view occluding contour to generate or manipulate a 3D shape, which can greatly reduces the difficulty and manual effort of sketch-based modeling.

3 APPROACH OVERVIEW

As illustrated in Figure 2, our method constructs a space that jointly embeds 3D shapes and 2D occluding contours. Our embedding space is not limited to any object categories or particular view directions, where a point represented by a latent vector can represent a sketch extracted from an object category with an arbitrary projection direction. And, our method can even generate a similar one through the trained VAE. That is, the latent vectors of three view projections (i.e., front, top, and side views) of a certain 3D object can be merged together to form a new vector that can be further used to generate the voxel representation of the 3D shape through a trained volumetric autoencoder.

To construct the joint embedding space, we use a two-step process to train a VAE and a volumetric autoencoder. At the first step, with the occluding contours extracted from a 3D model dataset as the training data, a VAE model is trained to encode the occluding contours to latent vectors that are distributed in a continuous embedding space (§4.2). Thanks to the multivariate Gaussian distribution in the continuous embedding space, any latent vector in the space can be decoded to an occluding contour, and the distance between latent vectors can also describe the similarity between the corresponding occluding contours. At the second step, to enable the latent vectors for the generation of 3D models, we employ supervised training on a volumetric autoencoder framework (§4.3). Specifically, since the three view projections of 3D models in our dataset have been known, we propose a loss function to minimize the distance between the combination of the latent vectors of the three view projections of a 3D model, with the encoder output of the volumetric autoencoder, and expect the output of the volumetric autoencoder to be similar to the voxel representation of the 3D model. In this way, the trained volumetric autoencoder is capable to generate corresponding 3D voxels given the occluding contours from the three views. Therefore, our joint embedding space can both represent and generate 2D occluding contours and 3D shapes, and it can be directly used for sketch-based modeling and shape manipulation (§5). Finally, we employ the marching cube algorithm [Lorensen and Cline 1987] and Laplacian smoothing [Field 1988] as post-processing steps to convert the 3D voxel representation to its corresponding polygon mesh model.

Data preprocessing. Our dataset consists of 1165 3D models collected from [Chen et al. 2009] and [Shilane et al. 2004]. We extended the data by uniformly generating 50 random viewpoints on a bounding sphere whose center is the same as the geometric center of the enclosed 3D shape (see an example in Figure 3 (left)). We obtained a total of 58,250 (1165*50) 3D projections at the end. After that, we employ an existing non-photorealistic rendering algorithm [DeCarlo et al. 2003] to generate occluding contours from the front, top, and side views (also called samples in this writing) for each 3D projection case, and record the three-views relation as the label (see Figure 3 (right)). We utilize three views of occluding contours instead of one to create embedding space since we intent to create the embedding space dense and the three views we select can cover enough information of a 3D model. In this way, we obtained a total of 174,750 (=58,250*3) samples and we then randomly split the samples to the VAE training data (90%) and the test data (10%). Figure 3 shows the process of collecting the occluding contours from the three views of a 3D model.

4 OUR METHOD

In this section, we first describe how to train a variational autoencoder network to encode the occluding contours to latent vectors, in order to construct an embedding space. Then, we detail how to jointly embed 3D shapes in the space by training a volumetric autoencoder, so that the latent vectors in the embedding space can be decoded to both 2D occluding contours and 3D shapes.

4.1 Variational Autoencoder

As a generative model, variational autoencoder (VAE) that is the key model to construct the embedding space in our work, has attracted a lot of attentions in recent years [Kingma and Welling 2013; Rezende et al. 2014; Tan et al. 2018]. VAE has a similar structure with the autoencoder (AE) [Kingma and Welling 2013] that consists

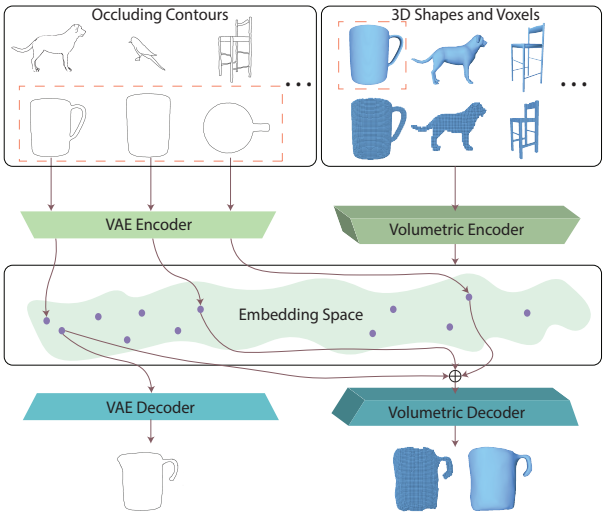

Figure 2: The architecture of our method, including a dataset that provides 3D shapes and their corresponding 2D occluding contours (top), a variational autoencoder and a volumetric autoencoder for the joint embedding of the 3D shapes and 2D contours (middle). The points in the embedding space are represented by latent vectors that can be decoded to 2D contours or 3D shapes, thus to support sketch-based 3D modeling and shape manipulation applications (bottom).

of an encoder $E_\theta(x)$ and a decoder $D_\phi(z)$. The main differences between VAE and AE is that, VAE approximates $E_\theta(x)$ as a posterior distribution $q(z|x)$, instead of using a deterministic function for $E_\theta(x)$ in AE [Kingma and Welling 2013]. $E_\theta(x)$ can encode input x to a latent vector z, while $D_\phi(z)$ can generate an output x' from the latent vector z. Particularly, VAE can generate new data x' by sampling z from a prior distribution $p_\phi(x|z)$. We train the encoding and decoding parameters θ and ϕ using stochastic gradient variational Bayes (SGVB) algorithm [Kingma and Welling 2013] as follows:

$$\theta^*, \phi^* = \arg\min_{\theta,\phi} \mathbb{E}_{z\sim E_\theta(x)}[-\log p_\phi(x|z)] + \mathbb{D}_{kl}(E_\theta(x)|p(z)), \quad (1)$$

where $\mathbb{D}_{kl}$ denotes the Kullback-Leibler divergence, which measures the difference between $E_\theta(x)$ and $p(z)$. In VAE, $p(z)$ is specified as a standard Normal distribution: $p(z) \sim N(0,1)$, and $E_\theta(x)$ is a multivariate Gaussian distribution: $E_\theta(x) \sim N(z_\mu, diag(z_\sigma))$.

The latent code z is sampled as:

$$z = z_\mu + \epsilon \odot z_\sigma, \epsilon \sim N(0,1), \quad (2)$$

where $\odot$ is an element-wise matrix multiplication operator. This expression of z is a re-parameterization trick [Kingma and Welling 2013] to make all the operations differentiable for back propagation.

4.2 Embedding Space Creation

To construct the joint embedding space, we train a VAE network using the 2D occluding contours extracted from our 3D shape dataset. We consider an occluding contour as a binary image and feed it to the VAE model at the training stage. Specifically, we reshape the input image to $128 \times 128 \times 1$ before feeding it to the VAE encoder,

Table 1: The VAE architecture used to create the embedding space. The last fully connected layer in the encoder is duplicated for z_μ and z_σ and uses Equation 2 to generate z as output. All convolutional layers are followed by batch normalization, leaky ReLU activation, and dropout layer, except the last layer of the decoder.

Layer	Type	Kernel	Stride	Output
enc.	conv2d	5×5	2×2	64×64×32
enc.	conv2d	5×5	2×2	32×32×64
enc.	conv2d	5×5	2×2	16×16×128
enc.	conv2d	5×5	2×2	8×8×256
enc.	fc	N/A	N/A	128
dec.	fc	N/A	N/A	16384
dec.	deconv2d	5×5	2×2	16×16×128
dec.	deconv2d	5×5	2×2	32×32×64
dec.	deconv2d	5×5	2×2	64×64×32
dec.	deconv2d	5×5	2×2	128×128×1

Figure 3: Examples of the projections for a 3D shape with random viewpoints with the same center (a). We extract three views of occluding contours (i.e., front, side, top views) for each projection (b).

and encode the input into an embedding space with 128-dimension parameters: z_μ and z_σ. Then, we sample z from z_μ and z_σ using the re-parameterization trick introduced in [Kingma and Welling 2013]. The latent vector can be used as the input of the decoder and reverted to a $128 \times 128 \times 1$ image. Table 1 shows the specifications of our VAE model.

We utilize two loss functions to learn the optimal weights of the VAE network. The first one is the reconstruction loss (denoted as $\mathcal{L}_{recon\_2D}$) between the output of the decoder and the input image, and the second is the KL divergence loss (denoted as $\mathcal{L}_{KL}$). Since our input is a binary image, we use binary cross entropy as the loss function for the reconstruction loss. In order to increase the impact of the reconstruction loss on the weights of the network, our final loss function is defined as follow:

$$\mathcal{L}_{VAE} = w_{recon}\mathcal{L}_{recon\_2D} + \mathcal{L}_{KL}. \quad (3)$$

The VAE encoder and decoder networks have similar architectures, with kernel size = 5, stride = 2, and padding = 1 for all the

convolution operations. After convolution operations, we add a batch normalization layer, leaky ReLU activation, and a dropout layer, except the last layer of the decoder. The *sigmoid* activation is used for the decoder output. We set w_{recon} to 100, keep the probability to 0.95 in the dropout layer, and set β to 0.2 in the leaky ReLU activation. We trained in total 50 epochs with the Adam solver, with the batch size = 64 and the learning rate = 0.5×10^{-3}.

4.3 Joint Embedding Space

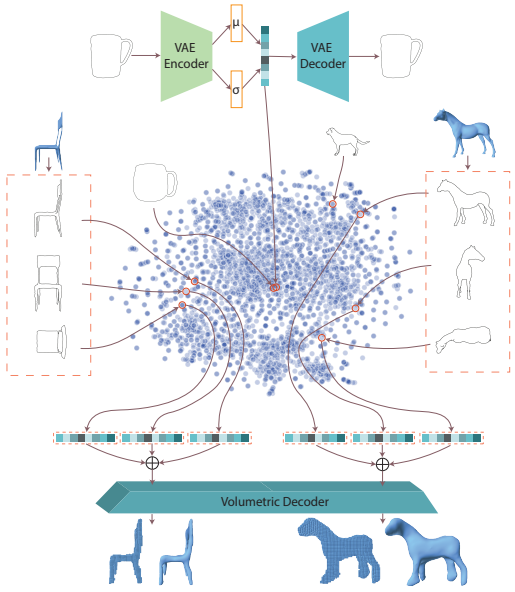

Figure 4: The 2D visualization of a portion of the embedding space. Each point in the space is a latent vector that is encoded by the trained VAE to represent a certain occluding contour (top). The latent vectors of the three view projections can be combined and decoded to a 3D voxel representation by a trained volumetric autoencoder (bottom).

As illustrated in Figure 4, the latent vectors of the 2D occluding contours form a space. 3D shapes can be projected with a certain viewpoint and then be embedded into the space. The distance between latent vectors in the embedding space represents the similarity between the corresponding occluding contours. Our VAE model can ensure this since the encoder actually extracts features (i.e., latent vectors) from input contours, and similar contours are close each other in the low dimensional embedding space. For example, the front view of a dog's occluding contour may be similar to the front view of a horse's occluding contour, and thus their corresponding latent vectors would have a small distance. However, this projection is a one-way mapping from 3D shapes to 2D occluding contours. We also want the latent vectors in the embedding space can be mapped (decoded) back to 3D shapes. To this end, we train a volumetric autoencoder to use the view-related latent vectors to reconstruct corresponding 3D shapes. We employ supervised training to force such a vector to be the same as the combination of the latent vectors of the projections of 3D shapes from the front, side, and top views. We choose to use the three orthogonal views since they are generally sufficient to characterize the geometry of 3D models.

Table 2: The autoencoder architecture used to map 3D shapes into the embedding space. All convolutional layers are followed by batch normalization, leaky ReLU activation, and dropout layer, except the last layer of the decoder.

Layer	Type	Kernel	Stride	Output
enc.	conv3d	5×5	2×2	32×32×32×32
enc.	conv3d	5×5	2×2	16×16×16×64
enc.	conv3d	5×5	2×2	8×8×8×128
enc.	fc	N/A	N/A	384
dec.	fc	N/A	N/A	65536
dec.	deconv3d	5×5	2×2	16×16×16×64
dec.	deconv3d	5×5	2×2	32×32×32×32
dec.	deconv3d	5×5	2×2	64×64×64×1

In order to feed 3D shapes to the network, we first voxelize 3D shapes to grids with $64 \times 64 \times 64$ resolution, where the value of a grid cell equals 1 if the cube of this grid is inside or on the boundary of the 3D shape; and 0 otherwise. The encoder then encodes the voxelized 3D shape to a 384-dimension (3×128) vector that can be decoded to the original 3D shape by the decoder. We expect the encoded vector to be a combination of the latent vectors of the projections of the 3D shapes from three orthogonal views in the order of the front view, the left view, and the top view in the embedding space. Specifically, we use 3D convolutional operations for convolution layers, and each convolution layer is followed by batch normalization layer, leaky ReLU activation, and dropout layer. The *sigmoid* activation is used in the decoder output to generate a grid with $64 \times 64 \times 64 \times 1$ resolution. The details of the volumetric autoencoder network are given in Table 2.

We implement two loss functions to train this network. The first one is a reconstruction loss $\mathcal{L}_{recon\_3D}$ to calculate the difference between the input 3D shape and the generated 3D shape. We use binary cross entropy to measure the difference as a normal autoencoder. The second loss function, which leads to the main difference between our method and a normal autoencoder is to measure the difference of the three-view latent vectors y generated by the volumetric encoder with the VAE encoder output $\hat{y}$, given three-views occluding contours of one 3D model, defined as follows:

$$\mathcal{L}_{view} = ||y - \hat{y}||_2. \tag{4}$$

Similar to the KL loss in our VAE model, the $\mathcal{L}_{view}$ is introduced to embed 3D shapes into the embedding space by minimizing the distance between its encoder output y and the VAE encoder outputs for three-views occluding contours $\hat{y}$ in the embedding space. The total loss of the network is expressed as follows:

$$\mathcal{L}_{AE} = \mathcal{L}_{recon\_3D} + \mathcal{L}_{view}, \tag{5}$$

The convolutional layers of the encoder and the decoder of the volumetric autoencoder have a similar architecture with the aforementioned VAE network, but use 3D convolution operations. We use the *sigmoid* activation for the output of the decoder. For superparameters, we set the keep probability to 0.95 for dropout

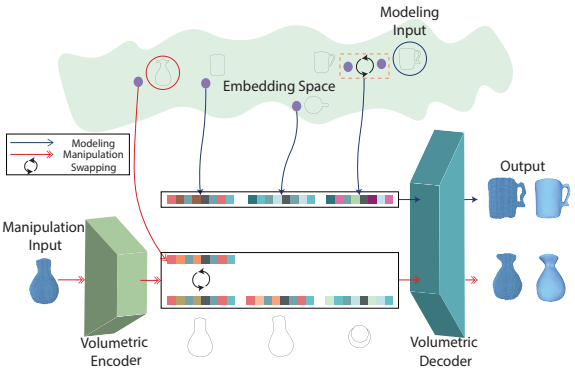

Figure 5: Workflows of the two selected applications of our method. For 3D modeling (blue arrows), the input occluding contour (in blue circle) is swapped with a similar one in our dataset and then is combined with the contours from the other two views to form a vector, which is decoded back to a new 3D shape. For shape manipulation (red arrows), the input 3D shape is first encoded to a vector, and then a similar occluding contour in the embedding space is swapped with the corresponding part in the vector.

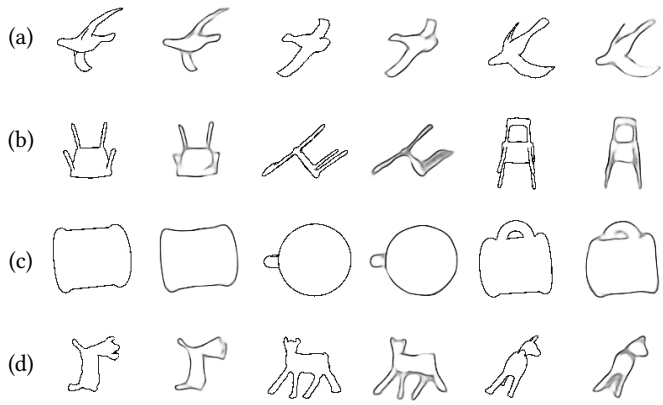

Figure 6: Comparisons between contours generated by the non-photorealistic rendering method given a 3D model (columns 1, 3 and 5) and by the VAE decoder given three latent vectors generated from the volumetric encoder (columns 2, 4 and 6). Front, left, and top views are compared from left to right. The categories are birds, chairs, cups, and fourlegs from (a) to (d), respectively.

layers, and 0.2 for leaky ReLU activation. We trained 80 epochs with the Adam solver, with the batch size = 64 and the learning rate = 0.5×10^{-3}.

5 SELECTED APPLICATIONS

In this work, we show two selected applications of our method. The first is to transform a hand-drawn or image-extracted sketch to a 3D model. The second is to edit or manipulate existing 3D models to generate shape variations.

3D modeling based on a single 2D sketch. . For this application, we first feed a user sketch to the encoder of the VAE network to generate its latent vector. Then, we search and identify an occluding contour sample in the embedding space with the minimum latent vector distance to the given sketch. Since the identified occluding contour is the projection of a specific 3D shape in our dataset, the latent vectors of the projections from the other two views are also known due to the pre-specified labels (i.e., the two views of occluding contours in our dataset). Therefore, the latent vector of the input sketch and the latent vectors of the other two views of the most similar occluding contour are fed to the trained volumetric autoencoder and generate a 3D voxel representation. In this pipeline, the only input is a single 2D sketch, the latent vectors of the other two views, required by the volumetric autoencoder, can be simply acquired by feeding the two views of the occluding contours of the searched 3D shape in our dataset to the VAE encoder or directly feeding the 3D shape to the encoder of the volumetric autoencoder to obtain the latent vectors. Figure 5 illustrates this process via an example (blue arrows).

3D model manipulation. . In this application, we first encode an input 3D voxel to its latent vectors of three views via the volumetric encoder, the three views of the latent vector have to follow the order of the front view, the left view, and the top view since we trained the volumetric autoencoder with this view order. Then, we use the latent vector of its front view projection to search for several similar occluding contour samples in the embedding space. These occluding contours provide possible deformation variations of the input 3D model. In this way, we can obtain shape variations of the input 3D model through the trained volumetric autoencoder: replacing its front view projection with a similar occluding contour in the embedding space. Moreover, users can also manually edit the front view projection of the input 3D model by removing or adding line segments to change its 2D shape, which can indirectly manipulate the resulting 3D shape. This application focus on manipulate outline of a given 3D model, details of the 3D model cannot be handled with this method. Figure 5 illustrates this process with an example (with red arrows). Note that this application cannot be done at the level of latent vectors due to the uncertainty of the relation between 3D surfaces and each dimension of the latent vectors.

6 RESULTS

In this section we show our experimental results. We also demonstrate the effectiveness of our method for sketch-based 3D modeling through comparisons with state-of-the-art, sketch-based 3D modeling methods.

As shown in the top of Figure 4, in the joint embedding space (visualized in 2D), we can observe the positions of the projections of the same 3D shape from different views, and the positions of the projections of different 3D shapes from the same view. We can see that the closer two occluding contours are in the space, the more similar shapes they have. This shows that the latent vectors that represent 2D occluding contours in the space can also describe their shape similarity. In the bottom of figure 4, we show that the latent vectors of three view projections of a 3D shape can be put

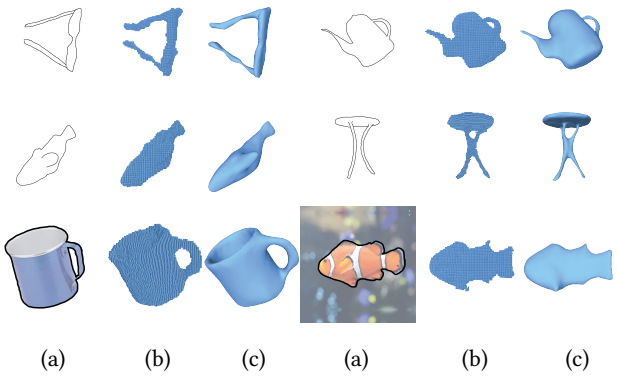

Figure 7: Sketch-based 3D modeling results by our approach. Our method can use either hand-drawn sketches (top two rows) or occluding contours extracted from images (the last row) of column (a) to generate 3D voxels representations (b) and polygon meshes (c).

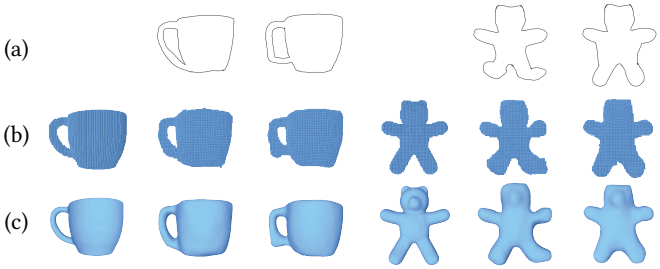

Figure 8: Shape manipulation results by our approach. We manually modified the occluding contours (a) of the given models to generate shape variations. In each case, the first 3D voxel representation and its corresponding polygon mesh are used as the reference, (b) and (c) are the generated voxel representations and their corresponding smoothed polygon meshes, respectively.

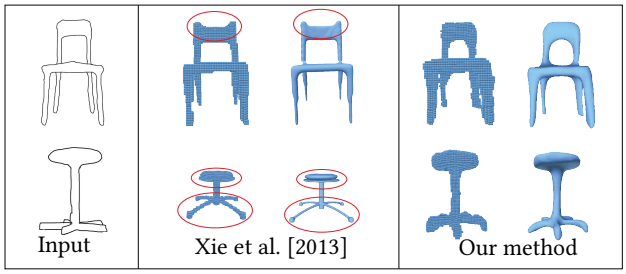

Figure 9: Comparisons between our method and the method by Xie et al. [2013].

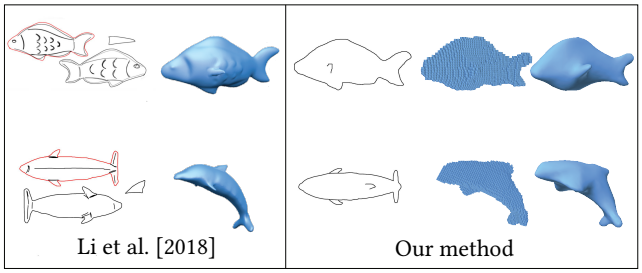

Figure 10: Comparisons with Li et al. [2018]. We show the required inputs of Li et al. [2018] and our method, as well as the corresponding modeling results. Note that the results by Li et al. [2018] are directly taken from the original authors' paper.

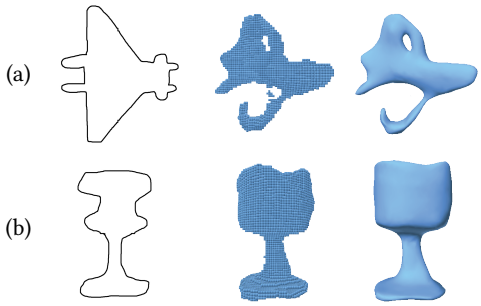

Figure 11: Failure cases by our method. (a) the input sketch is signficantly different from any shapes in the training dataset, and (b) the three orthogonal views of the input contain overlapping geometry.

together and then be decoded to a 3D voxel representation through the trained volumetric autoencoder. Since the embedding space is independent of object categories, our method is suitable for both man-made and natural objects.

Since the quality of the embedding space directly impacts the results by our method, we calculated the mean square error (MSE) for randomly selected 12 categories (i.e., airplane, ant, bird, chair, cup, fish, four-leg, hand, plier, table, teddy, and vase): the average MSE is 0.019. Figure 6 visualizes the comparisons between the contours generated by the non-photorealistic rendering method and generated by using the VAE decoder on three latent vectors obtained from the volumetric encoder. The examples shown in this figure were randomly selected from our experiments. As shown in this figure, the contours generated by our approach are reasonably close to the ground-truth.

To evaluate the quality of the volumetric autoencoder, we also calculated the mean IoU (Intersection over union), precision, and recall for 12 different categories with the voxel representations of the 3D shapes in our dataset as the ground truth, shown in Table 3. In each category, we randomly selected 900 samples to do the calculations. As shown in this table, our method performed reasonably well on the three quantitative measures, in particular, the precision and recall.

We also quantitatively compared our method with two baseline sketch-based modeling methods: Igarashi et al. [2007] and Nealen

Table 3: The IoU, precision, and recall of the decoded results by the volumetric autoencoder for 12 different categories, with the voxel representations of the 3D shapes in our dataset as the ground truth.

Category	IoU	Precision	Recall
Airplane	0.76	0.90	0.83
Ant	0.70	0.89	0.77
Bird	0.76	0.89	0.83
Chair	0.68	0.76	0.86
Cup	0.85	0.93	0.91
Fish	0.85	0.94	0.89
Four-leg	0.79	0.92	0.85
Hand	0.78	0.91	0.84
Plier	0.76	0.89	0.83
Table	0.76	0.89	0.83
Teddy	0.88	0.95	0.93
Vase	0.90	0.96	0.93
Average	0.79	0.90	0.86

et al. [2007]. Since it is intuitive to measure the reconstruction accuracy using IoU in 3D space, we voxelized the resulting meshes by Igarashi et al. [2007] and Nealen et al. [2007] so that we can calculate the IoU through the voxel expression with the original 3D models. We randomly selected three sketches from the categories of cup, airplane, and teddy. With the selected sketches, we generated 3D models using the method by Igarashi et al. [2007], the method by Nealen et al. [2007], and our method, respectively. The comparison results are shown in Table 4. We can clearly see that our method achieved higher IoU reconstruction accuracies.

We also compared our method with a state of the art, deep learning approach by Delanoy et al. [2018]. The comparison results are also shown in Table 4. We directly ran our method on the same chair and vase categories that were used by Delanoy et al. [2018]. We selected 10% of samples in each category and calculated the mean IoUs. Even though our training dataset only contains 57 different chairs and 36 different vases, compared to 540 different chairs and 270 different vases in the training dataset of Delanoy et al. [2018], our method can still obtain slightly higher IoUs than the method by Delanoy et al. [2018].

Figure 7 shows some 3D modeling results by our methods through hand-drawn sketches and image-extracted 2D occluding contours. The 3D voxel representation is the direct output from our model, which is further converted to 3D polygon meshes by the well-known marching cubes algorithm [Lorensen and Cline 1987]. The 3D models generated by our approach are well matched with the sketches from the given view point. Note that all the input sketches and the generated models are unseen during training process. Figure 8 shows the generated 3D shape variations by our method.

In Figure 9, we compare our method with the method by Xie et al. [2013]. The latter employs sketch-based object retrieval to select suitable parts from a dataset to assemble man-made objects. We show two cases with the same inputs, and also provide the voxel representations of the results by Xie et al. [2013]. In order to make a fair comparison, the input sketches in this comparison did not have related 3D models in the dataset that was used by both our method and Xie et al. [2013]. As shown in this figure, we can see both the two methods can create plausible 3D shapes for the given sketches. Meanwhile, we admit that the 3D model quality by such pure assembly-based methods like Xie et al. [2013] is better than that of our method. However, Xie et al. [2013] synthesizes 3D models based on part assembly, while our method can directly generate 3D models, not requiring a pre-created, man-made part depository. As highlighted in the red circled regions, certain parts of the chairs synthesized by Xie et al. [2013] are visibly different from the counterparts of the input sketches due to the limited coverage of the pre-created part repository. In particular, the synthesized chair by Xie et al. [2013] in the bottom of Figure 9 has 4 legs, while the input user sketch actually has 5 legs. By contrast, our method does not have such a limitation and can produce correct results.

Table 4: IoU comparisons between our method and two baseline methods: Igarashi et al. [2007] and Nealen et al. [2007], and between our method and a state of the art, deep learning method by Delanoy et al. [2018]. The larger the IoU, the better the reconstruction accuracy.

Method	Cup	Airplane	Teddy	Chair	Vase
Igarashi et al. [2007]	0.20	0.15	0.71	-	-
Nealen et al. [2007]	0.22	0.10	0.69	-	-
Delanoy et al. [2018]	-	-	-	0.38	0.57
Our method	**0.88**	**0.72**	**0.89**	**0.41**	**0.61**

In Figure 10, we also compare our method with the sketch-based 3D modeling method by Li et al. [2018] that directly generates 3D shapes based on sketch inputs. The work of [Li et al. 2018] shares a similar goal as our work. However, the work of [Li et al. 2018] requires more detailed (often non-trivial), semantic sketch inputs to indicate surface curvature information including the front and back views of the same 3D shape and additional information such as boundary (i.e., red lines) to assist the fusion of two surfaces (i.e., the front and back meshes) together in post-processing. By contrast, our method is substantially easier to use even for novice users thanks to the simple requirement on the input.

7 DISCUSSION AND CONCLUSION

In this paper we present a new method that jointly embeds 3D shapes and 2D occluding contours into a continuous space, in order to support applications including sketch-based 3D modeling and 3D shape manipulation. Specifically, we train a variational autoencoder to encode occluding contours to latent vectors, which can represent the occluding contours and measure their similarity in the space. We also train a volumetric autoencoder that uses a vector consists of the latent vectors of the three orthogonal view projections of a 3D model, to generate its 3D voxel representation. To evaluate the effectiveness of our method, we conducted many experiments and also demonstrated the selected applications of our approach for sketch-based 3D modeling and shape manipulation, as well as the comparisons with state-of-the-art, sketch-based 3D modeling methods.

Limitations. Our current method still has several limitations. First, our method only focuses on the generation of coarse 3D shapes. Since our method only can generate 3D models with coarse surface details, certain small-scale surface details cannot be generated. Hence, the 3D models by our method typically need further polishing or post-processing, or be used as the starting point to guide 3D shape deformations or assembly. Besides, due to the potential noise in the generated voxel representation, the results might need filtering, smoothing, or refinements based on the symmetry of the shape or other geometric constraints. Second, our method can only generate 3D models whose occluding contours are similar to those of the shapes in the dataset. If the input sketch is significantly different from any shapes in the dataset, our method may fail to produce desired results (one example is shown in Figure 11(a)). Third, if the three orthogonal views of the input contain overlapping geometry, our method could fail to handle such cases (an example is shown in Figure 11(b)).

We plan to improve our method to handle more meticulous surface details of 3D models. We are also interested in training a model that can directly encode the representation of the sketches of a 3D model in different views, without obtaining the 2D projections through non-photorealistic rendering. We believe that, not limited to the demonstrated sketch-based 3D modeling and shape manipulation applications, the introduced joint embedding space can find its potential use for other geometric processing and shape modeling applications.

ACKNOWLEDGMENTS

This work is in part supported by NSF IIS-1524782. Qiang Fu is in part supported by an Open Project supported by the Virtual Reality Systems and Technologies National Key Lab in China.

REFERENCES

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

Melinos Averkiou, Vladimir G Kim, Youyi Zheng, and Niloy J Mitra. 2014. Shapesynth: Parameterizing model collections for coupled shape exploration and synthesis. *Computer Graphics Forum* 33, 2 (2014), 125–134.

Seok-Hyung Bae, Ravin Balakrishnan, and Karan Singh. 2008. ILoveSketch: as-natural-as-possible sketching system for creating 3d curve models. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*. ACM, 151–160.

Volker Blanz, Thomas Vetter, et al. 1999. A morphable model for the synthesis of 3D faces.. In *proc. of ACM Siggraph'99*, Vol. 99. 187–194.

Tao Chen, Zhe Zhu, Ariel Shamir, Shi-Min Hu, and Daniel Cohen-Or. 2013. 3-sweep: Extracting editable objects from a single photo. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 195.

Xiaobai Chen, Aleksey Golovinskiy, and Thomas Funkhouser. 2009. A Benchmark for 3D Mesh Segmentation. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 28, 3 (Aug. 2009).

Matthew T Cook and Arvin Agah. 2009. A survey of sketch-based 3-D modeling techniques. *Interacting with computers* 21, 3 (2009), 201–211.

Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. 2003. Suggestive contours for conveying shape. *ACM Transactions on Graphics (TOG)* 22, 3 (2003), 848–855.

Johanna Delanoy, Mathieu Aubry, Phillip Isola, Alexei Efros, and Adrien Bousseau. 2018. 3D Sketching using Multi-View Deep Volumetric Prediction. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1, 21 (may 2018).

Chao Ding and Ligang Liu. 2016. A survey of sketch based modeling systems. *Frontiers of Computer Science* 10, 6 (01 Dec 2016), 985–999.

David Eigen and Rob Fergus. 2015. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision*. 2650–2658.

Lubin Fan, Ruimin Wang, Linlin Xu, Jiansong Deng, and Ligang Liu. 2013. Modeling by drawing with shadow guidance. *Computer Graphics Forum* 32, 7 (2013), 157–166.

David A Field. 1988. Laplacian smoothing and Delaunay triangulations. *Communications in applied numerical methods* 4, 6 (1988), 709–712.

Qiang Fu, Xiaowu Chen, Xiaoyu Su, and Hongbo Fu. 2016. Natural lines inspired 3D shape re-design. *Graphical Models* 85 (2016), 1–10.

Justin Johnson Georgia Gkioxari, Jitendra Malik. 2019. Mesh R-CNN. *ICCV* (2019).

Yotam Gingold, Takeo Igarashi, and Denis Zorin. 2009. Structured annotations for 2D-to-3D modeling. *ACM Transactions on Graphics (TOG)* 28, 5 (2009), 148.

Derek Hoiem, Alexei A Efros, and Martial Hebert. 2005. Geometric context from a single image. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, Vol. 1. IEEE, 654–661.

Haibin Huang, Evangelos Kalogerakis, Ersin Yumer, and Radomir Mech. 2017. Shape Synthesis from Sketches via Procedural Models and Convolutional Networks. *IEEE transactions on visualization and computer graphics* 23, 8 (2017).

Takeo Igarashi, Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. 2007. Teddy: a sketching interface for 3D freeform design. In *Acm siggraph 2007 courses*. ACM, 21.

Levent Burak Kara and Kenji Shimada. 2007. Sketch-based 3d-shape creation for industrial styling design. *IEEE Computer Graphics and Applications* 27, 1 (2007), 60–71.

Levent Burak Kara and Kenji Shimada. 2008. Supporting early styling design of automobiles using sketch-based 3d shape construction. *Computer-Aided Design and Applications* 5, 6 (2008), 867–876.

Olga A Karpenko and John F Hughes. 2006. SmoothSketch: 3D free-form shapes from complex sketches. *ACM Transactions on Graphics (TOG)* 25, 3 (2006), 589–598.

I. K. Kazmi, L. You, and J. J. Zhang. 2014. A Survey of Sketch Based Modeling Systems. In *2014 11th International Conference on Computer Graphics, Imaging and Visualization*. 27–36.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

Changjian Li, Hao Pan, Yang Liu, Xin Tong, Alla Sheffer, and Wenping Wang. 2017. BendSketch: modeling freeform surfaces through 2D sketching. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 125.

Changjian Li, Hao Pan, Yang Liu, Xin Tong, Alla Sheffer, and Wenping Wang. 2018. Robust Flow-guided Neural Prediction for Sketch-based Freeform Surface Modeling. *ACM Trans. Graph.* 37, 6, Article 238 (Dec. 2018), 12 pages.

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

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. 2015. SMPL: A skinned multi-person linear model. *ACM transactions on graphics (TOG)* 34, 6 (2015), 248.

William E Lorensen and Harvey E Cline. 1987. Marching cubes: A high resolution 3D surface construction algorithm. In *ACM siggraph computer graphics*, Vol. 21. ACM, 163–169.

Zhaoliang Lun, Matheus Gadelha, Evangelos Kalogerakis, Subhransu Maji, and Rui Wang. 2017. 3d shape reconstruction from sketches via multi-view convolutional networks. In *2017 International Conference on 3D Vision (3DV)*. IEEE, 67–77.

Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. 2007. FiberMesh: designing freeform surfaces with 3D curves. *ACM transactions on graphics (TOG)* 26, 3 (2007), 41.

Luke Olsen, Faramarz F Samavati, Mario Costa Sousa, and Joaquim A Jorge. 2009. Sketch-based modeling: A survey. *Computers & Graphics* 33, 1 (2009), 85–103.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082* (2014).

Elad Richardson, Matan Sela, Roy Or-El, and Ron Kimmel. 2017. Learning Detailed Face Reconstruction From a Single Image. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Alec Rivers, Frédo Durand, and Takeo Igarashi. 2010. 3D Modeling with Silhouettes. *ACM Trans. Graph.* 29, 4 (2010), 109:1–109:8.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.

Ashutosh Saxena, Min Sun, and Andrew Y Ng. 2008. Make3d: Learning 3d scene structure from a single still image. *IEEE transactions on pattern analysis and machine intelligence* 31, 5 (2008), 824–840.

Tianjia Shao, Wilmot Li, Kun Zhou, Weiwei Xu, Baining Guo, and Niloy J. Mitra. 2013. Interpreting Concept Sketches. *ACM Transactions on Graphics* 32, 4 (2013), 10.

Philip Shilane, Patrick Min, Michael Kazhdan, and Thomas Funkhouser. 2004. The princeton shape benchmark. In *Proceedings Shape Modeling Applications, 2004.* IEEE, 167–178.

Tanasai Sucontphunt, Zhenyao Mo, Ulrich Neumann, and Zhigang Deng. 2008. Interactive 3D facial expression posing through 2D portrait manipulation. In *Proceedings of graphics interface 2008*. Canadian Information Processing Society, 177–184.

Qingyang Tan, Lin Gao, Yu-Kun Lai, and Shihong Xia. 2018. Variational autoencoders for deforming 3d mesh models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5841–5850.

Flora Ponjou Tasse and Neil Dodgson. 2016. Shape2vec: semantic-based descriptors for 3d shapes, sketches and images. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 208.

Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. 2016. Multi-view 3d models from single images with a convolutional network. In *European Conference on Computer Vision*. Springer, 322–337.

A. T. Tran, T. Hassner, I. Masi, E. Paz, Y. Nirkin, and G. Medioni. 2018. Extreme 3D Face Reconstruction: Seeing Through Occlusions. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Fang Wang, Le Kang, and Yi Li. 2015. Sketch-based 3d shape retrieval using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1875–1883.

Rouven Winkler, Chengchao Qu, Sascha Voth, and Jürgen Beyerer. 2018. 3D Face Reconstruction from Low-Resolution Images with Convolutional Neural Networks. In *Proceedings of the 2018 the 2nd International Conference on Video and Image Processing*. ACM, 83–88.

Xiaohua Xie, Kai Xu, Niloy J Mitra, Daniel Cohen-Or, Wenyong Gong, Qi Su, and Baoquan Chen. 2013. Sketch-to-design: Context-based part assembly. *Computer Graphics Forum* 32, 8 (2013), 233–245.

Baoxuan Xu, William Chang, Alla Sheffer, Adrien Bousseau, James McCrae, and Karan Singh. 2014. True2Form: 3D curve networks from 2D sketches via selective regularization. *ACM Transactions on Graphics* 33, 4 (2014).

Mingliang Xu, Mingyuan Li, Weiwei Xu, Zhigang Deng, Yin Yang, and Kun Zhou. 2016. Interactive mechanism modeling from multi-view images. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 236.

Robert C Zeleznik, Kenneth P Herndon, and John F Hughes. 2007. SKETCH: An interface for sketching 3D scenes. In *ACM SIGGRAPH 2007 courses*. ACM, 19.