




Semantics-Guided Latent Space Exploration for Shape Generation

Tansin Jahan[†]  Yanran Guan[†]  Oliver van Kaick 

School of Computer Science, Carleton University, Canada

Abstract

We introduce an approach to incorporate user guidance into shape generation approaches based on deep networks. Generative networks such as autoencoders and generative adversarial networks are trained to encode shapes into latent vectors, effectively learning a latent shape space that can be sampled for generating new shapes. Our main idea is to enable users to explore the shape space with the use of high-level semantic keywords. Specifically, the user inputs a set of keywords that describe the general attributes of the shape to be generated, e.g., “four legs” for a chair. Then, our method maps the keywords to a subspace of the latent space, where the subspace captures the shapes possessing the specified attributes. The user then explores only this subspace to search for shapes that satisfy the design goal, in a process similar to using a parametric shape model. Our exploratory approach allows users to model shapes at a high level without the need for advanced artistic skills, in contrast to existing methods that allow to guide the generation with sketching or partial modeling of a shape. Our technical contribution to enable this exploration-based approach is the introduction of a label regression neural network coupled with shape encoder/decoder networks. The label regression network takes the user-provided keywords and maps them to distributions in the latent space. We show that our method allows users to explore the shape space and generate a variety of shapes with selected high-level attributes.

CCS Concepts

• **Computing methodologies** → **Shape modeling**;

1. Introduction

Creating digital models of 3D shapes is a challenging task for novice users, since traditional modeling tools involve the creation of shapes at a low level, where all the fine details of the surfaces have to be explicitly modeled. Thus, over the last twenty years, computer graphics research has also developed alternative approaches to facilitate modeling of shapes for non-expert users. One line of research proposed the use of parametric models of shapes. With these models, the user can synthesize new shapes, such as human faces [BV99] or bodies [ASK*05], by manipulating a set of sliders that specify the parameters of a shape model. Thus, the shape does not have to be explicitly modeled by the user and can be synthesized at a high level. However, conventional parametric shape models are only applicable to shapes that can be described by a template, since a one-to-one mapping between all the shapes in a collection is required by these methods.

Recently, there has been great interest in using deep neural networks for synthesizing shapes, such as variational autoencoders (VAEs) [BLRW16] or generative adversarial networks (GANs) [WZX*16], which learn statistical models from collections

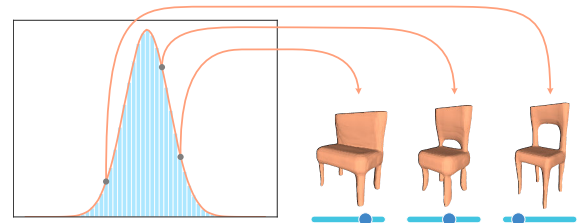


Figure 1: Our approach for semantics-guided shape generation: after specifying keywords that constrain the attributes of the generated shapes (“straight square back” and “four straight short legs” in this example), the user can manipulate a set of sliders to explore the subspace of shapes with these attributes. The subspace is modeled as a set of mixtures of distributions [Bis94], one for each dimension of a latent representation of the shapes. The example shows three shapes generated by navigating through one specific dimension of the shape space.

of shapes. In these approaches, the networks learn to encode high-dimensional shapes into low-dimensional latent vectors and then decode the latent vectors back into shapes, so that a shape can be generated by simply sampling a latent vector and providing it to the decoder. Thus, the space spanned by the latent vectors can be seen

[†] Both authors contributed equally to this work.

as a type of parametric shape space. These approaches have sparked great interest since the shape spaces can be learned with weaker constraints, requiring only a consistent alignment of the shapes in the collection rather than a mapping to a template.

There have been several efforts investigating the best shape representation to be used in conjunction with deep networks, such as voxel grids [WZX*16, BLRW16], octrees [WLG*17], atlases of parameterizations [GFK*18], implicit fields [CZ19, PFS*19], point clouds [ADMG18], and graphs of parts [LXC*17, NW17]. However, less effort has been spent on how to effectively control these generative models for enabling users to create a shape according to a pre-defined set of goals or an intended design.

Although the shape spaces learned by the networks are low-dimensional when compared to the shapes themselves, to allow the networks to learn effective shape models, the latent vectors still need to possess a large number of dimensions, e.g., 128 dimensions or more. Thus, manually exploring the shape space spanned by the latent vectors is impractical, since many dimensions have to be considered and not all latent vectors correspond to meaningful shapes. As a result, in the literature, most of the synthesis methods have been evaluated by randomly sampling latent vectors or interpolating these random vectors [WZX*16]. One work proposes a “snapping” mechanism akin to sketching approaches, where the user creates a partial model by aggregating cubic blocks with an interface similar to Minecraft, and the method then “snaps” the shape to the manifold defined by the latent space, updating the user-modeled shape to the closest matching shape in the latent space [LYF17]. Other works introduce approaches that convert 2D sketches drawn by a user into 3D objects, either with the use of procedural modeling [HKYM17] or deformable models [SBS19]. These are valuable tools for aiding modeling with neural networks, but they still require sufficient artistic skills from the user.

In this paper, we introduce a method to facilitate the navigation of the shape spaces learned with deep networks by incorporating semantic information into the process. Our key idea is to let the user explore the shape space according to high-level semantic keywords, where the keywords characterize the attributes that the intended shape should possess, e.g., “four legs” and “square back” for a chair. Specifically, given the user input, we map the attributes to a subspace of the latent space, where the subspace captures the shapes possessing the attributes. The user then explores only the subspace to search for shapes that satisfy the intended design (Figure 1), similarly to using a parametric shape model. To enable an efficient exploration, we map the keywords to distributions of the latent dimensions, such that the user is only required to explore dimensions with significant variance (such as the distribution shown in Figure 1). Thus, our method enables an exploratory modeling approach, where the user can model shapes at a high level and is not required to possess artistic skills as required by sketching or partial modeling approaches.

To enable this exploration-based approach, our technical contribution is the introduction of a method composed of a label regression neural network coupled with shape encoder/decoder networks. The label regression network (LRN) has a custom architecture that takes the user-provided keywords and maps them to multiple distributions in the latent space. These distributions capture the multi-

ple modes that characterize the shapes related to the selected keywords. The network is trained with the latent representation of a collection of shapes paired with semantic keywords. Moreover, for encoding/decoding shapes to/from latent vectors, we use networks introduced in previous work and couple them with our LRN to provide a complete framework for exploration. Specifically, we use a generalized autoencoder (GAE) [WHWW14, GJvK20] for encoding shapes, and a feed-forward network with fully-connected layers based on an implicit representation [CZ19] for decoding shapes. Note that our framework does not allow to synthesize entirely novel shapes, but is able to interpolate and thus combine attributes of the training shapes, which is the goal of the exploratory approach.

We apply our method to three collections of shapes and show with a qualitative evaluation that this solution enables users to explore the latent space with less effort. We also present a quantitative evaluation of the quality of the generated shapes based on inception scores, and an analysis of the method to demonstrate that the learned models are sound.

2. Related work

In this section, we briefly review traditional statistical shape models and then discuss the latest developments using deep networks.

Statistical and parametric shape models. Statistical shape models summarize an entire collection of shapes of the same class with a set of parameters [BSBW16]. To obtain a useful model, the number of parameters in the model should usually be much smaller than the dimensionality of the data, e.g., 20 parameters compared to thousands of vertices that compose the triangle meshes of the shapes. A popular parametric model in the literature is built by first representing all the shapes in a collection with a consistent triangle mesh, where all the vertices across the meshes are in correspondence. Then, by applying a statistical analysis method such as principal component analysis (PCA) to the vertex positions, the main modes of variation of the shapes can be discovered. A user can then explore the shape space by tweaking a set of sliders that modify the parameters, and visualize the corresponding shapes [BSBW16]. This type of approach has been used most notably for creating shape models of human faces [BV99] and bodies [ASK*05]. Nevertheless, these methods can also be applied to other types of shapes, e.g., Wang et al. [WLJ*18] create statistical models of botanical trees by defining a metric in the shape space and then analyzing the data with a geodesic form of PCA.

Similarly to statistical models, parametric shape models allow to generate variations of a base shape by manipulating a set of parameters, although the parameters are explicitly defined during modeling. Based on this representation, Schulz et al. [SSB*17] perform retrieval on parametric shape collections by describing the parametric shapes as manifolds in a descriptor space. Talton et al. [TGY*09] introduce an interface to enable exploration of parametric models by learning a probability map over the design space with kernel density estimation.

Modeling aided by keywords. More closely related to our work, Streuber et al. [SQRH*16] introduce a method to generate human bodies with user-defined ratings of textual shape attributes, e.g.,

the user quantifies how “muscular” or “curvy” the body should be. The main building block of the approach is the learning of a function that relates the textual attribute ratings to parameters of a human shape model. Chaudhuri et al. [CKGF13] learn an association between semantic attributes given by users and shape parts with an optimization approach. The association is then used in an exploratory modeling interface. Yumer et al. [YCHK15] allow to continuously deform shapes by manipulating handles associated to semantic keywords such as “ergonomic” or “elegant”. Achlioptas et al. [AFH*19] investigate different natural language models to produce or evaluate textual descriptions reflecting the structure of shapes. Chen et al. [CCS*18] introduce a method for generating 3D shapes from textual descriptions by learning a joint text and shape embedding space.

Our method shares similarities with these works, such as the use of keyword attributes to describe the characteristics of the shapes to be created. However, our method is designed specifically to be used with recent deep networks that learn latent shape spaces, and allows further exploration of the latent space.

Object generation with deep networks. In the last few years, deep neural networks have shown great success in performing generative tasks, being used for example to generate images [RAY*16], indoor scenes [WSCR18], and terrains [GDG*17]. Our focus in this paper is on the generation of man-made objects, and a variety of deep learning approaches for creating this type of data representation have also been introduced. Wu et al. [WZX*16] introduced a GAN that learns a latent space of object shapes based on a voxel grid representation of the shapes. The latent space can then be sampled to generate new objects. Similarly, Brock et al. [BLRW16] use VAEs and voxel grids to learn a shape space that can be used for object generation, while Dai et al. [DQN17] use an encoder-predictor network to perform shape completion, which can be seen as a constrained type of synthesis.

Moreover, since a voxelization is not the ideal parameterization for 2D surfaces embedded in 3D, other approaches have explored alternative shape representations in conjunction with neural networks. Groueix et al. [GFK*18] explicitly store the shape topology by encoding the shape as an atlas of local parameterizations. Park et al. [PFS*19] and Chen and Zhang [CZ19] encode shapes as implicit functions which are able to generate much smoother objects. Park et al. [PFS*19] make use of signed distance functions, which can be reconstructed with existing methods, while Chen and Zhang [CZ19] encode shapes as inside/outside binary indicator functions. Furthermore, Chen et al. [CTZ20] use an implicit encoding to generate shapes with a recursive subdivision of space into convex sets. For shapes that can be represented as graphs of parts, Li et al. [LXC*17] introduce an approach that encodes shapes as a hierarchy of parts, while Nash and Williams [NW17] introduce a method that models both part geometry and connectivity. Both methods can be used for shape generation. Wu et al [WZX*20] generates shapes by sequential assembly of shape parts.

All of the methods discussed above can be used to generate shapes based on input latent vectors or example representations of shapes. The latter can be achieved in encoder-decoder networks by training the encoder with a different data representation than the

decoder, e.g., encoding an image into a latent vector and then decoding it into a voxel grid. However, directly controlling the generation to achieve a desired design is challenging, as in the case when the user knows only the general attributes that the designed shape should possess. We see our work as a way of complementing the existing approaches by enabling the user to explore the latent spaces and more closely control the shape generation.

Interactive modeling with deep networks. A few deep learning approaches have been proposed to control the generation of shapes more closely. Guérin et al. [GDG*17] introduce a terrain authoring system, where the user can draw a 2D sketch of the topographic features of a terrain, and a GAN then synthesizes a corresponding elevation model. For man-made shapes, Huang et al. [HKYM17] introduce an approach where a user can draw a 2D sketch of the desired shape, and the system then generates a corresponding 3D object. This is accomplished by learning a neural network that maps sketches to parameters of a procedural modeling program, which then synthesizes the shape based on an algorithm. Smirnov et al. [SBS19] map sketches to deformable parametric templates. Zekun et al. [ZAESB20] introduce a method where shapes represented with a set of primitive geometries can be manipulated to guide the generation of fine-grained meshes. This is accomplished by learning a joint embedding of coarse and fine-grained shape geometries. Liu et al. [LYF17] allows to create partial shapes with a Minecraft-type interface, which are then projected onto the latent manifold learned by a GAN, providing a corresponding shape in the latent space. Sung et al. [SSK*17] introduce the ComplementMe system where a user can design a man-made shape in an interactive manner. Given a partially constructed shape, the system suggests additional parts that can be added to the shape, according to a graph-based shape model learned by neural networks.

In this paper, we also introduce an approach that allows users to control the generation of shapes in a more interactive manner. In contrast to the works discussed above, our method allows the generation to be performed as a mix of a guided and exploratory process. The user first provides keywords that describe the high-level characteristics of the shapes. The system then allows the user to explore the subspace of shapes that possess these characteristics. This is advantageous in that the user may have a general idea of the intended design, but may not have necessarily determined all the fine details that the final shape should possess, or may not have sufficient artistic skills to manually model or draw the fine details. Differently from simply learning an embedding of shapes [LSQ*15], the use of deep generative networks allow us to interpolate shapes in the collection, leading to novel shapes. We explain our method in more detail as follows.

3. Semantics-guided shape generation

Our solution to semantics-guided shape generation consists of three deep networks: (i) a label regression network (LRN) that learns a mapping from keywords to the latent space of shapes, (ii) a shape encoder network (SEN) that learns a latent manifold from a set of shapes, and (iii) a shape decoder network (SDN) that reconstructs distance fields of shapes from samples of the latent manifold. The use of the three networks in our workflow is illustrated in Figure 2. We describe the three networks in the following subsections.

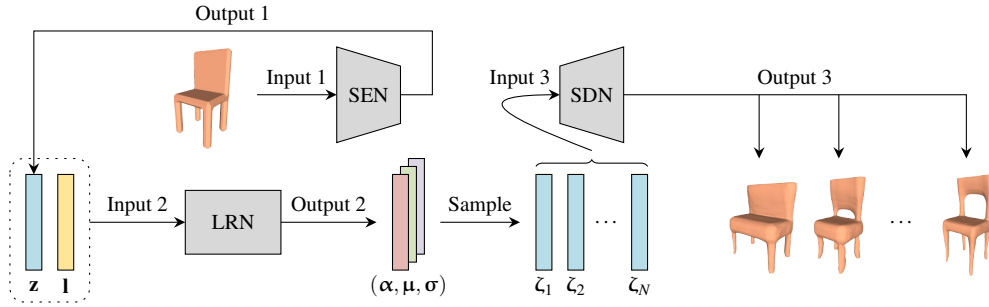


Figure 2: Our semantics-guided shape generation involves three different deep networks. During the training phase, we train the shape encoder network (SEN) to learn a latent manifold from a set of shapes represented by volumes. Then, for each shape, we use its learned latent representation \mathbf{z} and its user-provided labels \mathbf{I} to train the label regression network (LRN), which learns to map a set of labels into a set of mixture models (α, μ, σ) that describe distributions of latent vectors. After training, given only a set of requested labels \mathbf{I} , we predict a set of mixture models with the LRN. Then, we can sample latent vectors ζ from the mixture models and use the shape decoder network (SDN) to reconstruct occupancy fields of shapes from the vectors.

3.1. Label regression network (LRN)

The LRN takes as input user-provided keywords that describe a target shape. The network then maps the keywords to the latent space learned by the SEN. Thus, the output encoding of the latent space used by the LRN is tied to the representation learned by the SEN. Our SEN maps shapes to latent vectors of dimension n , while the SDN maps latent vectors back to shapes. The SEN and SDN are described in the following sections.

A first design for the LRN would be to simply train it to map keywords to latent vectors. That would allow to map a set of keywords to a single point in the shape space, but would not allow the users to explore the shape variability. Thus, our chosen design for the network is to map a set of keywords to a mixture of h Gaussian distributions of latent vectors, which can be seen as forming a probabilistic subspace of the shape space. In our work, we use $h = 5$. The user can then sample different vectors from the mixture to explore the subspace, or methodically change the entries of the latent vectors according to the distributions.

Input and output. The input to the LRN is a set of keywords represented as a binary vector $\mathbf{I} \in \mathbb{B}^m$, where the entry l_i associated to the i -th keyword in the dictionary is set to 1 or 0 depending on whether the keyword is selected or not. The output of the network is a mixture model of latent vectors conditioned on the input labels, represented as three vectors α , μ , and σ , where α is a h -dimensional vector storing the mixing coefficient for each distribution, and μ and σ are two vectors with dimension $h \times n$ storing the mean and standard deviation that represent each Gaussian distribution in the mixture model. Specifically, for each dimension i of a hypothetical latent vector \mathbf{z} , the network outputs h pairs of mean and standard deviation $\{(\mu_{w_i}, \sigma_{w_i}) \mid w \in \{1, 2, \dots, h\}\}$, with each pair associated with a mixing coefficient α_w .

Network architecture. The architecture of our LRN is illustrated with the diagram in Figure 3. The regression is performed with a feed-forward deep network composed of three hidden layers, one input layer with m nodes (with one node per label), and three out-

put layers that provide the parameters of n mixture models (one mixture model per dimension of the latent space). All the layers are fully-connected, since all of the output distributions may be dependent on all of the input labels. Since the standard deviations σ_{w_i} should always be positive, for the output layer producing σ , we use modified exponential linear units (ELUs) as the activation functions to ensure the non-negativity of the outputs. We define the ELU for an input scalar x as:

$$\text{ELU}(x) = \begin{cases} x + 1, & \text{if } x \geq 0, \\ \exp(x), & \text{otherwise.} \end{cases} \quad (1)$$

Training and loss function. The LRN is trained with a set $\mathcal{T}_{\text{LRN}} = \{(\mathbf{I}^j, \mathbf{z}^j)\}$. Each sample $(\mathbf{I}^j, \mathbf{z}^j)$ corresponds to one training shape, containing the binary label vector \mathbf{I}^j that describes the attributes of the training shape and the latent vector $\mathbf{z}^j \in \mathbb{R}^n$ produced for the shape by the SEN. We describe our dataset in Section 4. The training objective of the regression is to produce, for each entry i of each sample \mathbf{z}^j , a mixture model that maximizes the sampling probability of z_i^j . The mixture model is composed of h Gaussian distributions and its probability density is represented by a linear combination of Gaussian kernel functions [Bis94]:

$$P(z_i^j \mid \mathbf{I}^j) = \sum_{w=1}^h \alpha_w \phi_{w_i}(z_i^j \mid \mathbf{I}^j), \quad (2)$$

where h is the number of components in the mixture, α_w is the mixing coefficient, and $\phi_{w_i}(z_i^j \mid \mathbf{I}^j)$ is the probability density function (PDF) of the target z_i^j for the w -th kernel:

$$\phi_{w_i}(z_i^j \mid \mathbf{I}^j) = \frac{1}{\sigma_{w_i} \sqrt{2\pi}} \exp\left(-\frac{(z_i^j - \mu_{w_i})^2}{2\sigma_{w_i}^2}\right), \quad (3)$$

where μ_{w_i} and σ_{w_i} represent the mean and standard deviation of the w -th kernel for each entry i . Thus, the loss function of the network is defined as minimizing the negative logarithm of the PDFs of the

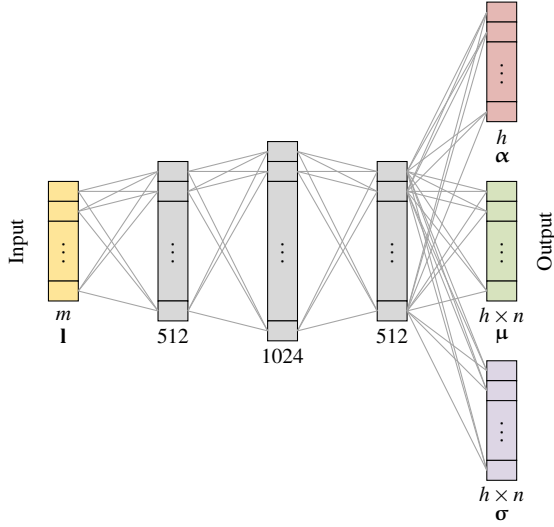


Figure 3: The architecture of our feed-forward deep network for label regression, mapping m input labels into n mixture models, with each model having h components. The numbers denote the dimensions of the input/output and intermediate representations.

kernels, which for one entry i of one sample j is denoted as:

$$\mathcal{L}_{\text{LRN}} = -\log \left(P(z_i^j | I^j) \right). \quad (4)$$

3.2. Shape encoder network (SEN)

We use a 3D generalized autoencoder (3D-GAE) [GJvK20] to encode shapes with our approach, where shapes are represented as volumes partitioned into voxels. The reason for using a 3D-GAE is that this network is able to learn a manifold latent space from data relations explicitly provided during training. Specifically, by providing a measure of shape similarity to the network, the 3D-GAE is able to learn a latent space where the proximity of latent vectors corresponds to the similarity of their corresponding training shapes. Moreover, we use a volumetric representation for encoding shapes since a volume is simpler to process by convolutional neural networks than other representations. Nevertheless, our method can use any encoder network that maps some representation of shapes into a latent space. Specifically, as discussed in Section 3.1, the training data \mathcal{T}_{LRN} for the label regression is composed of pairs (I^j, z^j) of label and latent vectors. Thus, to generate the training data, we require a network that can encode shapes into a latent representation. This requirement rules out the use of GANs commonly used for shape synthesis, but other encoder networks such as autoencoders (AEs), VAEs, and VAE-GANs [LSLW16] can be used.

Input and output. The SEN is composed of two networks: an encoder and a decoder. The encoder takes as input a shape represented as a set of $32 \times 32 \times 32$ voxels, where a voxel can be defined as either occupied (1) or empty (0). The encoder outputs a latent representation $\mathbf{z} \in \mathbb{R}^n$ of the shape. The decoder then translates a latent vector back into a shape represented as a volume. Note that, we require the encoder and decoder for training the network. However,

once the network has learned its objective, we discard the decoder and use mainly the encoder to create the training data for the LRN. Moreover, given that the last layer of the decoder uses a sigmoid activation function, each output voxel contains a real value in the range $[0, 1]$. Thus, we transform each voxel into a binary value according to a threshold of 0.5, that is, the voxel becomes 0 if the real value is below 0.5, or 1 otherwise.

Network architecture. We use a symmetric architecture where the encoder and decoder have the same number of layers. The encoder uses three layers for three levels of downsampling, while the decoder performs upsampling also at three different levels. Specifically, the input is downsampled from $32 \times 32 \times 32$ to $4 \times 4 \times 4$ using three convolutional layers followed by a batch normalization layer that helps the network to learn features independently from the output of previous layers. We use filters of size $4 \times 4 \times 4$ and stride 2 for all the convolutional layers. The output layer of the encoder, which generates the latent vector \mathbf{z} , is a fully-connected layer with a rectified linear unit (ReLU). We experimented with different values for the size of the latent vectors and found that 128 dimensions provide the best encodings with our architecture. The decoder follows the inverse of this architecture for performing upsampling, with the exception that the output layer uses a sigmoid activation function.

Training and loss function. The 3D-GAE is trained with the GAE loss [WHWW14], where the reconstruction $\hat{\mathbf{x}}^j$ of each input shape \mathbf{x}^j is compared to a set of shapes $\Omega_{\mathbf{x}^j}$, which consists of the k -nearest neighbors of \mathbf{x}^j given by the Chamfer distance [FSG17]. In our work, we use $k = 10$. The 3D-GAE loss combines the GAE loss with the reconstruction error of each batch of shapes \mathcal{B} , so that the model can better converge to a global optimum [GJvK20]:

$$\mathcal{L}_{\text{SEN}} = \|\mathcal{B} - \hat{\mathcal{B}}\|^2 + \sum_{\mathbf{x}^j \in \mathcal{B}} \sum_{\mathbf{x}^{j'} \in \Omega_{\mathbf{x}^j}} s_{j,j'} \|\mathbf{x}^{j'} - \hat{\mathbf{x}}^j\|^2, \quad (5)$$

where $s_{j,j'}$ is the weight given by Laplacian eigenmaps [BN01]:

$$s_{j,j'} = \exp \left(-\frac{\|\mathbf{x}^j - \mathbf{x}^{j'}\|^2}{t} \right), \quad (6)$$

and t is an empirical tuning parameter. In our work, we set $t = 200$.

The training set \mathcal{T}_{SEN} is composed of voxelized triangle meshes. To voxelize a model, we subdivide its triangles until each edge is smaller than the sides of a voxel, and then denote as occupied all the voxels that contain vertices of the subdivided mesh.

3.3. Shape decoder network (SDN)

With the latent manifold learned by the SEN, instead of also using a volumetric decoder to reconstruct latent vectors into shapes, we use the network introduced by Chen and Zhang [CZ19] to decode latent vectors into an implicit representation of shapes, which leads to reconstructed shapes with a higher visual quality. Specifically, the input latent vector ζ is reconstructed into an occupancy field \mathcal{F} , where $\mathcal{F}(\mathbf{p}) = 0$ for points outside the shape and $\mathcal{F}(\mathbf{p}) = 1$ for points inside the shape.

Input and output. The input to the decoder is an $(n + 3)$ -dimensional vector, composed of the n -dimensional vector ζ sampled from the latent manifold concatenated with a 3D point \mathbf{p} . The implicit decoder aims to find a parameterization $f(\zeta, \mathbf{p})$ that maps each input vector to the value $\mathcal{F}(\mathbf{p})$. Thus, the output of the network is a value in the range $[0, 1]$ that indicates whether the point \mathbf{p} is inside or outside the shape encoded by ζ .

Network architecture. The SDN follows a feed-forward architecture [CZ19], with five fully-connected hidden layers for five levels of downsampling that have respectively 2048, 1024, 512, 256, and 128 units. Each of the first four hidden layers is skip-connected with the input vector and all the five layers use leaky ReLU as the activation function. The output layer is also fully-connected with the previous hidden layer, but uses a sigmoid activation function.

Training and loss function. The SDN simply uses the mean squared error as the loss function. For training the network, for each input shape \mathcal{S} , we first pre-compute a set of points \mathcal{P} uniformly sampled from \mathcal{S} and the ground-truth occupancy field $\mathcal{F}_{\mathcal{S}}$. Then, for each point $\mathbf{p} \in \mathcal{P}$, we minimize the average squared difference between the ground-truth occupancy $\mathcal{F}_{\mathcal{S}}(\mathbf{p})$ and the predicted value $f(\zeta, \mathbf{p})$:

$$\mathcal{L}_{\text{SDN}} = \frac{1}{|\mathcal{P}|} \sum_{\mathbf{p} \in \mathcal{P}} \|\mathcal{F}_{\mathcal{S}}(\mathbf{p}) - f(\zeta, \mathbf{p})\|^2. \quad (7)$$

The loss is summed for all training shapes. We use the latent vectors learned for the training shapes by the SEN to train the SDN, with each latent vector \mathbf{z}^j paired with a set of points \mathcal{P}^j sampled from its corresponding shape. Thus, the training set can be denoted as $\mathcal{T}_{\text{SDN}} = \{(\mathbf{z}^j, \mathcal{P}^j)\}$.

3.4. Exploration of the shape space

Once the three networks are trained, as explained above, the user can employ them to perform an exploration of the latent space and generate new shapes. The user first inputs a set of labels \mathbf{I} which are mapped with the LRN to three vectors (α, μ, σ) representing the predicted mixture model. The mixture model is then used to create a latent vector ζ according to further user input. Specifically, our system first ignores the insignificant Gaussian components, i.e., components associated with low mixing coefficients α . Then, for a Gaussian component w and each dimension i of the latent space, our system exposes the dimension to the user only if σ_{wi} is significant (larger than a threshold ϵ). If σ_{wi} is not significant, we set $\zeta_i = \mu_{wi}$. We show the exposed dimensions to the user sorted by the magnitude of the standard deviations. Finally, the user can explore the exposed dimensions by varying a set of sliders that navigate around the space $(\mu_{wi} - \sigma_{wi}, \mu_{wi} + \sigma_{wi})$ to define each ζ_i . For any ζ_i defined with this process, the decoder network generates a corresponding shape. This results in an interactive experience where the user can analyze the shape generated by the new parameters selected. As we will discuss in Section 4, we observed experimentally that the number of dimensions with large σ is relatively small, and thus it is feasible for the user to manipulate them in an exploratory manner.

4. Results and evaluation

We first describe the set up of our experiments and then present qualitative results, followed by an overall evaluation of our method. Our implementation is available at <https://github.com/IsaacGuan/SGSG>.

4.1. Experimental setup

Shape dataset. We present results for our method on a set of 400 chairs from the COSEG dataset [WAvK*12] and 100 tables and lamps from the auto-aligned ModelNet40 [WSK*15, SB15]. We use attributes of the original shapes to derive the labels for training the LRN, described as follows.

Label dataset. We assign labels to each shape of the dataset. The labels describe a variety of visual attributes of the shapes, especially properties of their individual parts. To create the labels, we define a set of shape features and their admissible values, which are then transformed into binary labels. For example, the property “leg length” of chairs admits the values “short” and “long”, which are then transformed into the binary labels “leg length short” and “leg length long”. More details of our label dataset are described in Appendix A.

Optimizer and hyper-parameters for learning. We trained all the networks with the Adam optimizer. We split the datasets into training batches of 10 instances each for the LRN and SEN. As the hidden layers of the LRN are fully-connected layers that require extensive training, we set the learning rate for the LRN to 3×10^{-4} and train the network for 3000 epochs. For the SEN, we set the learning rate to 1×10^{-3} and train the network for 200 epochs. And for the SDN, we use a learning rate of 5×10^{-5} and adopt the progressive training scheme suggested by Chen and Zhang [CZ19], where we first train the SDN for 50 epochs with occupancy fields of shapes sampled from $16 \times 16 \times 16$ grids, then 50 epochs with occupancy fields sampled from $32 \times 32 \times 32$ grids, and finally 100 epochs with occupancy fields sampled from $64 \times 64 \times 64$ grids. Also, we pre-train the SDN for chairs and fine-tune it with lamps and tables.

Learning objectives. To prevent deep networks from overfitting, it is important to train the networks with datasets that are large enough and for an adequate number of epochs. In our setting, we would argue that overfitting is less of a concern, since we are not using our networks to predict the encoding for unknown test shapes. The main requirement in terms of generalization is that the network should be able to perform a meaningful interpolation between the latent vectors of the training shapes. We verify this requirement by monitoring the training loss and with a visual inspection of the results. After training, the average losses for the LRN, SEN, and SDN are respectively around 0.5, 3, and 0.03. Regarding the scalability of our method, to augment a given dataset with more shapes, it is possible to train the networks starting from the current set of neuron weights to reduce training time.

Timing and machine configuration. Our deep networks were trained with an NVIDIA GeForce RTX 2080 Ti GPU with 11 GB

of memory and CUDA version 10.0. The prediction performed by the LRN takes 1.1 ms, while the generation of a shape, including the implicit field prediction with the SDN and mesh creation with marching cubes, takes 0.25 s. Thus, once the networks are trained, the method allows for real-time exploration of the shape subspaces.

4.2. Qualitative results

Table 1 shows example shapes generated with our approach. We manually created these results with our exploration method, to simulate the scenario where a user would explore the spaces constrained by the specified labels. Each shape was created in less than a minute as can be seen in the supplementary video. Specifically, to create these shapes, the user first selects a set of labels, which are shown on the left column of the figure. The LRN then predicts the mixtures of Gaussians for all the entries of the latent vectors. The center column shows a summary of the predicted distributions. Specifically, we take the standard deviation (σ) of the Gaussian distribution with most significant α in the mixture predicted for each dimension, and show the sorted σ 's of all dimensions in the plots. The right column shows examples of shapes that the user obtained by varying the entries of the latent vectors with significant standard deviation. We see how the generated shapes show variations in their structure and overall geometry. While the user was able to generate a variety of shapes, all of the shapes in a row still possess the attributes specified by the input labels. Moreover, we see in the σ plots that not all the dimensions of the latent vectors need to be considered in the exploration. We also see that some of the selected labels provide richer subspaces with more variance (as seen by the shape of the σ plots), while other labels constrain more the variation that can be found. Note also that the selected labels include configurations of labels that do not exist in the training data.

We further examine the individual examples. In Table 1(a), (b), and (f), the user provided a set of labels that constrain more the seats and legs of chairs than their backs. Thus, the exploration of the distributions of these three examples provides shapes that mainly vary in the type of back. We observe that the backs of the generated chairs vary from either straight to round and can also have holes or not. The width of the seats in (b) also varies since the provided label did not constrain the size of the seat. In (c) and (e), the user provided many constraints for the back of the chairs and was able to explore different types of legs, generating, e.g., beam-type legs, three/four roller legs, and four straight legs. In (d), the chairs are constrained by the labels to a more fixed structure, except for the presence or not of holes on the backs.

Furthermore, for the lamps in Table 1(h), the shapes are constrained to have a bell shade and pipe-type body. Thus, the variation in the shapes can be seen among the bases. We observe round and square bases and also the absence of a base among the generated lamps. We also observe a few geometric variations in the bell shades of these lamps. A slightly different label set is provided for (g), where most of the constraints are on the body and base of the lamps rather than on the shades. As a result, we can see rectangular, round, and bell shades present among the generated shapes. For the tables in (j), the variation among the number and connection of legs are expected as the labels constrain mainly the table top. Therefore, we see tables with one and four legs, and straight or roller-type legs.

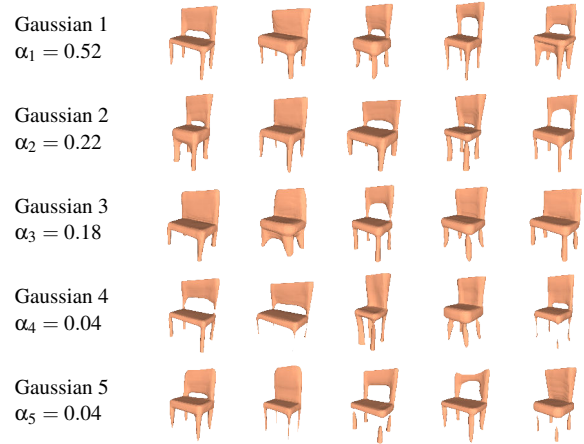


Figure 4: Shapes generated from different Gaussian distributions in one mixture model, where the mixture model is derived from the label set of Table 1(b). Note how the Gaussians with lowest alpha also generate the less meaningful shapes.

We also see a limitation of the approach in the example shown in (e). Although the user selected the label “vertical ladder” to constrain the back of the chairs, the learned implicit model is not fine enough to capture the multiple holes on the back, generating mainly one or no holes.

Table 1 shows results for the most significant Gaussians in the mixtures predicted for each example. In Figure 4, we show one exploration session where the user also considered other Gaussians in the mixture models. We see that the first three Gaussians provide an interesting range of variations of chairs with the given set of labels, although no Gaussian is responsible for a specific shape feature, e.g., there is variation in back holes in all of the first three Gaussians. This shows that the features in the latent space are not distributed according to a single Gaussian and thus using multiple distributions is more effective in capturing the label space. Moreover, the last two Gaussians which have low α values also provide less meaningful shapes as expected, showing that they can be ignored in the exploration.

We also performed an analysis of the space covered by the Gaussians to verify that the Gaussians in the mixtures capture different modes. Specifically, we measure the distance between the means of Gaussians with significant standard deviation, and compare these distances to the average pairwise distances between training shapes in the latent space. We find that, although there is overlap among the Gaussians, each Gaussian covers different regions of the latent space, as the distances between means are larger than 1% of the pairwise distances between shapes. Thus, a single distribution does not capture the same amount of detail as a mixture model.

4.3. Significance of standard deviation

One important question related to the exploration process is how to determine what standard deviation values are significant and will

	Labels	Distributions	Generated shapes							
(a)	Back: size — full fill — solid Seat: shape — square Leg: number — four length — long type — roller									
(b)	Back: size — full side view — straight Seat: shape — square Leg: number — four length — short type — straight									
(c)	Back: size — full fill — solid side view — straight front view — curved Seat: shape — circular Leg: length — short									
(d)	Back: size — half side view — straight front view — square Seat: shape — square Leg: number — four length — short type — straight									
(e)	Back: size — full fill — vertical ladder fill — hole(s) side view — bent front view — square Seat: shape — circular Leg: length — short									
(f)	Back: side view — straight Seat: shape — square Leg: number — four length — long type — straight									
(g)	Shade: top view — hole fitting — empty Body: length — medium type — pipe structure — straight Base: shape — round connection — untangled									
(h)	Shade: front view — bell fitting — empty Body: type — pipe Base: connection — untangled									
(i)	Top: type — single Leg: number — four type — straight Side: connection — open									
(j)	Top: type — single shape — round Leg: length — medium									

Table 1: Examples of results obtained with our method. Left: input labels selected by the user. Center: a plot of the sorted standard deviations (σ 's) for the most significant Gaussian in the mixture model predicted for each latent dimension. Right: shapes generated by creating different latent vectors according to the distributions. Note how the generated shapes possess the attributes described by the selected labels while revealing different variations in shape and sometimes structure.

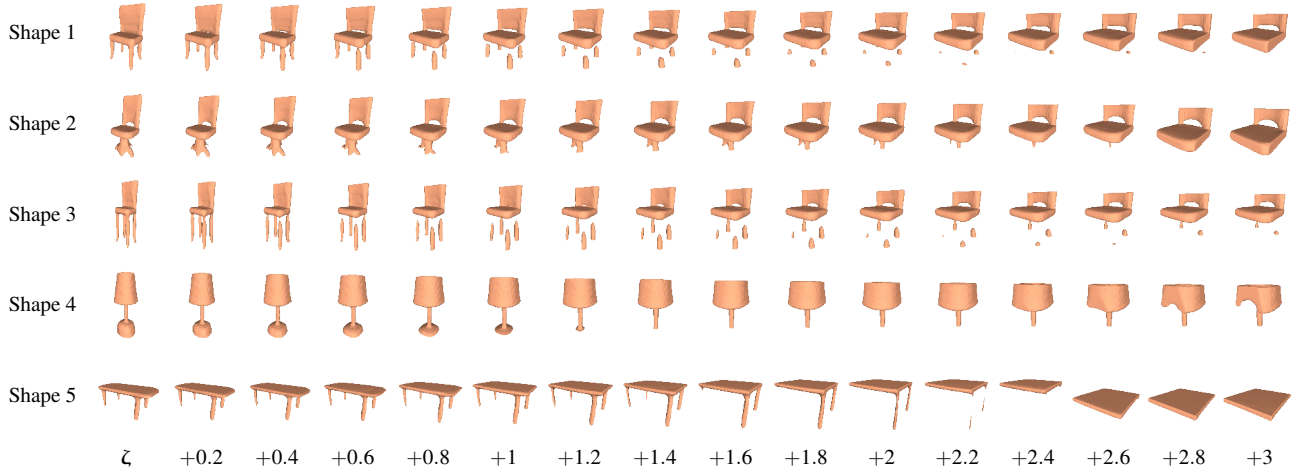


Figure 5: Visual examples of the changes that occur in a generated shape when adding perturbations to an initial latent vector.

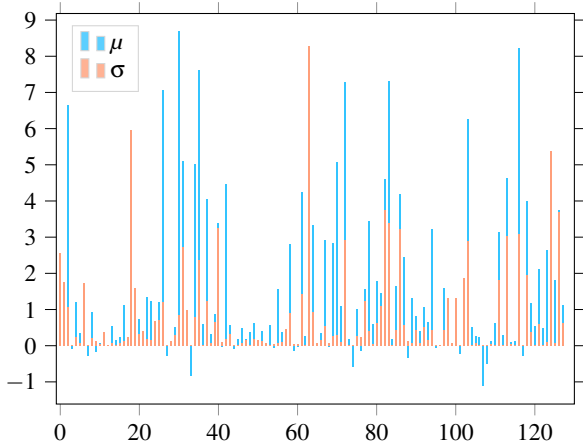


Figure 6: Distributions generated by the LRN for an input set of labels, represented with means (μ 's) and standard deviations (σ 's) of the most significant Gaussian in each mixture model. Each entry of a latent vector is one point along the x -axis. Note how there are entries with non-zero mean that can have low variance.

lead to variations in the generated shapes. To determine this threshold, we performed an experimental analysis of the sensitivity of the generated shapes to changes in the entries of the latent vectors. In this experiment, we start with a latent vector ζ derived from the mean of the most significant distribution predicted for a set of labels, and generate a shape corresponding to ζ . Next, we add perturbation vectors with increasing magnitude to ζ , and visually inspect the changes to the shape. Figure 5 shows a few shapes generated from the perturbed vectors, where we see that a magnitude of less than 0.2 does not lead to noticeable changes in the results. At the same time, a magnitude of more than 1 makes many of the shapes deviate significantly from the shape manifold. Thus, we select 0.2 as a threshold on σ for determining the dimensions to be explored

Method	Chair	Lamp	Table
3D-AE	2.98 ± 0.41	1.33 ± 0.31	2.19 ± 0.28
3D-VAE	2.92 ± 0.38	1.19 ± 0.25	1.78 ± 0.32
3D-GAE	3.09 ± 0.36	1.57 ± 0.24	2.45 ± 0.35
Ours	5.25 ± 0.56	2.81 ± 0.45	4.76 ± 0.56

Table 2: Inception scores of shapes generated by our method compared with 3D-AE, 3D-VAE, and 3D-GAE.

by the user. Moreover, when presenting the distributions to the user, we sort the dimensions by the magnitude of the σ values so that the user can give priority to dimensions with more variance.

Finally, in Figure 6, we show an example of the most significant Gaussian predicted for a set of labels, to provide evidence that the idea of facilitating the exploration according to the standard deviation values is sound. We see that, when σ values are low, the corresponding mean values can still be larger than zero, implying that low standard deviations do not only happen when the latent dimensions degenerate to constant zero distributions, but also appear for latent dimensions that are important in the shape encoding.

4.4. Evaluation and comparisons to other methods

We quantitatively evaluate the quality of the shapes generated by our method using the inception score and compare to other approaches in Table 2. Several works have used inception scores to evaluate the quality of the output of deep networks. We use the scoring method of Xie et al. [XZG*18] based on the pre-trained model of Qi et al. [QSN*16]. From our generated shapes, we manually selected 100 shapes of each class and compute the score, simulating the scenario where a user selects the best products of an exploration process. For reference, we compare our score with a volumetric autoencoder (3D-AE), a volumetric variational autoencoder (3D-VAE), and the 3D-GAE, which produce volumes as output. We compare our method to these networks according to the scores

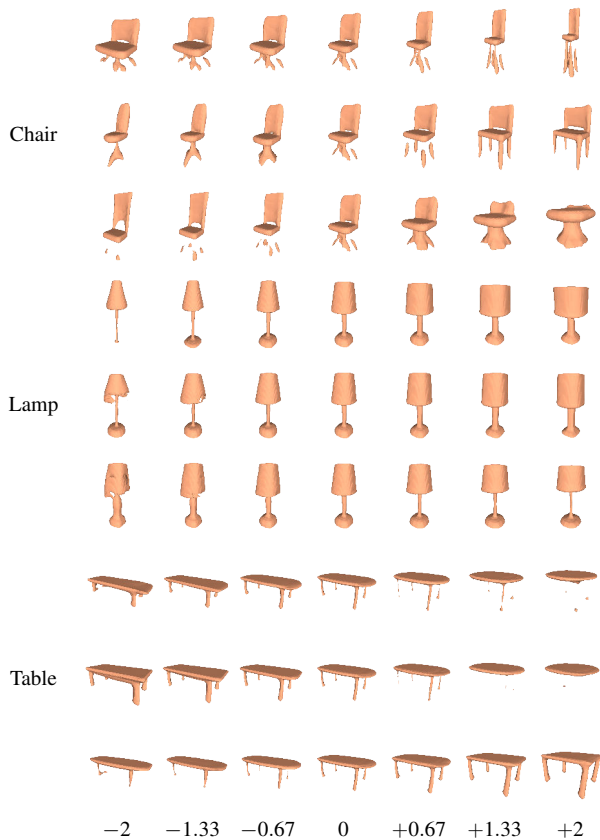


Figure 7: Results of moving along the three largest principal components of the latent spaces learned from chairs, lamps, and tables.

reported by Guan et al. [GJK20]. We observe that, the inception scores of manually selected shapes generated with the implicit representation are higher than those of shapes randomly sampled from the 3D-GAE based on a volumetric representation. This is of course expected due to the manual exploration involved, but serves as a confirmation that the generated shapes are of high quality.

We also compare our method to a shape space exploration method based on PCA. Similarly to GANSpace [HHL20], we apply PCA to the latent space to discover interpretable network controls. Specifically, we first compute the principal component decomposition of the latent vectors learned from the training shapes, where the decomposition preserves 95% of the original distribution. Then, we compute the mean and standard deviation on each principal axis. Finally, we add perturbations ranging from -2 to $+2$ times of the standard deviation to the mean on each principal axis and bring the edited vectors back to the original latent space using the inverse transform of the decomposition. Note that when applying a perturbation to a certain component, all other components are fixed to the mean. Figure 7 shows shapes reconstructed from the latent vectors created through this process, where we show results where the three largest components of each category of shapes were edited. We see that each principal component simultaneously tends to control three or more semantic features of a shape. Thus,

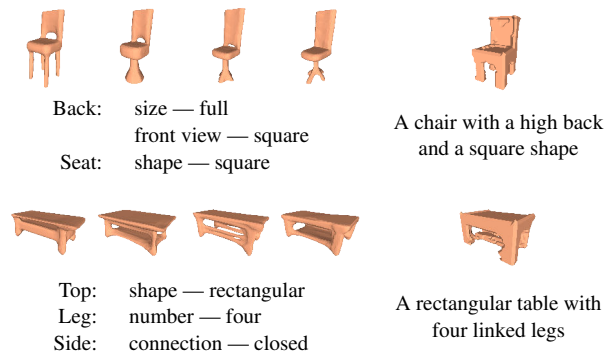


Figure 8: Comparison of shapes generated with our method (left) to results of Text2Shape (right). The labels and textual descriptions input to the methods are shown on the bottom of the shapes.

although the principal component decomposition can reduce the dimensionality of the latent space for exploration purposes, it does not allow to specify semantic constraints for shape generation, as several features are mixed together in each principal component. As a contrast, our method allows a more efficient exploration of the latent space by constraining the shape semantics.

Furthermore, we compare our method to Text2Shape [CCS*18], which allows users to synthesize shapes from textual descriptions based on learning a joint embedding of text and shapes. Figure 8 compares some of our generated shapes to Text2Shape results. We see that Text2Shape allows to generate one shape based on a textual description containing keywords similar to our labels. However, Text2Shape does not allow to further explore the space around the generated shape and obtain additional results, as enabled by our method. Moreover, the Text2Shape architecture is based on a volumetric shape representation which provides lower-quality meshes.

5. Conclusion, limitations, and future work

We introduced a method to facilitate the exploration of latent spaces for the generation of shapes with deep neural networks. We demonstrated that a mapping of semantic labels to distributions in the latent space enables users to explore subspaces of shapes constrained by the labels, effectively allowing the user to generate a variety of shapes with the specified attributes. This is made possible by the combination of a label regression network that learns distributions of latent vectors conditioned on the labels, and a generative network which translates sampled latent vectors into 3D shapes.

Although we demonstrated that the mappings learned by these two networks are sound and lead to a meaningful exploration and results, our work has certain limitations. First, our method requires a dataset of shapes labeled with keywords describing the attributes of the shapes. We manually assigned keywords to three datasets for our study. We show results with chairs and lamps since these are some of the categories of man-made shapes with the most structural and geometric variability. However, a crowdsourcing effort could provide the labeling for additional categories and larger datasets.

Evaluating the method on additional data could demonstrate the general applicability of the method.

Regarding the technical components of the method, currently the LRN maps a set of labels to a mixture of Gaussians in the latent space. It would be valuable to explore if other types of distributions or statistical models could be used to learn such a mapping. This would enable to capture modes in the latent space with different shapes and characteristics. In addition, when generating a shape from a given latent vector, we could perform a “snapping” of the vector onto the latent space [LXC*17], to obtain a shape that is part of the learned manifold, possibly improving the visual quality of the generated shape.

Acknowledgments

We thank the anonymous reviewers for their valuable comments. This work was supported by NSERC (2015-05407).

References

- [ADMG18] ACHLIOPTAS P., DIAMANTI O., MITLIAGKAS I., GUIBAS L.: Learning representations and generative models for 3D point clouds. In *Proc. Int. Conf. on Machine Learning* (2018), pp. 40–49. 2
- [AFH*19] ACHLIOPTAS P., FAN J., HAWKINS R., GOODMAN N., GUIBAS L.: ShapeGlot: Learning language for shape differentiation. In *Proc. IEEE Int. Conf. on Computer Vision* (2019), pp. 8937–8946. 3
- [ASK*05] ANGUELOV D., SRINIVASAN P., KOLLER D., THRUN S., RODGERS J., DAVIS J.: SCAPE: Shape completion and animation of people. *ACM Trans. on Graphics* 24, 3 (2005), 408–416. 1, 2
- [Bis94] BISHOP C. M.: *Mixture Density Networks*. Tech. Rep. NCRG/94/004, Aston University, 1994. 1, 4
- [BLRW16] BROCK A., LIM T., RITCHIE J. M., WESTON N.: Generative and discriminative voxel modeling with convolutional neural networks. *CoRR abs/1608.04236* (2016). 1, 2, 3
- [BN01] BELKIN M., NIYOGI P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems* (2001), pp. 585–591. 5
- [BSBW16] BRUNTON A., SALAZAR A., BOLKART T., WUHRER S.: Statistical shape spaces for 3D data: A review. In *Handbook of Pattern Recognition and Computer Vision*. World Scientific, 2016, pp. 217–238. 2
- [BV99] BLANZ V., VETTER T.: A morphable model for the synthesis of 3D faces. In *Proc. SIGGRAPH* (1999), pp. 187–194. 1, 2
- [CCS*18] CHEN K., CHOY C. B., SAVVA M., CHANG A. X., FUNKHOUSER T., SAVARESE S.: Text2Shape: Generating shapes from natural language by learning joint embeddings. In *Proc. Asian Conf. on Computer Vision* (2018), pp. 100–116. 3, 10
- [CKGF13] CHAUDHURI S., KALOGERAKIS E., GIGUERE S., FUNKHOUSER T.: AttribIt: Content creation with semantic attributes. In *Proc. ACM Symp. on User Interface Software and Technology* (2013), pp. 193–202. 3
- [CTZ20] CHEN Z., TAGLIASACCHI A., ZHANG H.: BSP-NET: Generating compact meshes via binary space partitioning. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* (2020), pp. 42–51. 3
- [CZ19] CHEN Z., ZHANG H.: Learning implicit fields for generative shape modeling. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* (2019), pp. 5932–5941. 2, 3, 5, 6
- [DQN17] DAI A., QI C. R., NIESSNER M.: Shape completion using 3D-encoder-predictor CNNs and shape synthesis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* (2017), pp. 6545–6554. 3
- [FSG17] FAN H., SU H., GUIBAS L.: A point set generation network for 3D object reconstruction from a single image. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* (2017), pp. 2463–2471. 5
- [GDG*17] GUÉRIN É., DIGNE J., GALIN E., PEYTAVIE A., WOLF C., BENES B., MARTINEZ B.: Interactive example-based terrain authoring with conditional generative adversarial networks. *ACM Trans. on Graphics* 36, 6 (2017), 228:1–228:13. 3
- [GFK*18] GROUEIX T., FISHER M., KIM V. G., RUSSELL B. C., AUBRY M.: A papier-mâché approach to learning 3D surface generation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* (2018), pp. 216–224. 2, 3
- [GJvK20] GUAN Y., JAHAN T., VAN KAICK O.: Generalized autoencoder for volumetric shape generation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition Workshops* (2020), pp. 1082–1088. 2, 5, 10
- [HHLP20] HÄRKÖNEN E., HERTZMANN A., LEHTINEN J., PARIS S.: GANSpace: Discovering interpretable GAN controls. In *Advances in Neural Information Processing Systems* (2020), pp. 9841–9850. 10
- [HKYM17] HUANG H., KALOGERAKIS E., YUMER E., MECH R.: Shape synthesis from sketches via procedural models and convolutional networks. *IEEE Trans. on Visualization and Computer Graphics* 23, 8 (2017), 2003–2013. 2, 3
- [LSLW16] LARSEN A. B. L., SØNDERBY S. K., LAROCHELLE H., WINTHER O.: Autoencoding beyond pixels using a learned similarity metric. In *Proc. Int. Conf. on Machine Learning* (2016), pp. 1558–1566. 5
- [LSQ*15] LI Y., SU H., QI C. R., FISH N., COHEN-OR D., GUIBAS L. J.: Joint embeddings of shapes and images via CNN image purification. *ACM Trans. on Graphics* 34, 6 (2015), 234:1–234:12. 3
- [LXC*17] LI J., XU K., CHAUDHURI S., YUMER E., ZHANG H., GUIBAS L.: GRASS: Generative recursive autoencoders for shape structures. *ACM Trans. on Graphics* 36, 4 (2017), 52:1–52:14. 2, 3, 11
- [LYF17] LIU J., YU F., FUNKHOUSER T.: Interactive 3D modeling with a generative adversarial network. In *Proc. Int. Conf. on 3D Vision* (2017), pp. 126–134. 2, 3
- [NW17] NASH C., WILLIAMS C. K. I.: The shape variational autoencoder: A deep generative model of part-segmented 3D objects. *Computer Graphics Forum* 36, 5 (2017), 1–12. 2, 3
- [PFS*19] PARK J. J., FLORENCE P., STRAUB J., NEWCOMBE R., LOVEGROVE S.: DeepSDF: Learning continuous signed distance functions for shape representation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* (2019), pp. 165–174. 2, 3
- [QSN*16] QI C. R., SU H., NIESSNER M., DAI A., YAN M., GUIBAS L. J.: Volumetric and multi-view CNNs for object classification on 3D data. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* (2016), pp. 5648–5656. 9
- [RAY*16] REED S., AKATA Z., YAN X., LOGESWARAN L., SCHIELE B., LEE H.: Generative adversarial text to image synthesis. In *Proc. Int. Conf. on Machine Learning* (2016), pp. 1060–1069. 3
- [SB15] SEDAGHAT N., BROX T.: Unsupervised generation of a view-point annotated car dataset from videos. In *Proc. IEEE Int. Conf. on Computer Vision* (2015), pp. 1314–1322. 6
- [SBS19] SMIRNOV D., BESSMELTSEV M., SOLOMON J.: Learning manifold patch-based representations of man-made shapes. *CoRR abs/1906.12337* (2019). 2, 3
- [SQRH*16] STREUBER S., QUIROS-RAMIREZ M. A., HILL M. Q., HAHN C. A., ZUFFI S., O’TOOLE A., BLACK M. J.: Body Talk: Crowdshaping realistic 3D avatars with words. *ACM Trans. on Graphics* 35, 4 (2016), 54:1–54:14. 2
- [SSB*17] SCHULZ A., SHAMIR A., BARAN I., LEVIN D. I. W., SITTHI-AMORN P., MATUSIK W.: Retrieval on parametric shape collections. *ACM Trans. on Graphics* 36, 1 (2017), 11:1–11:14. 2

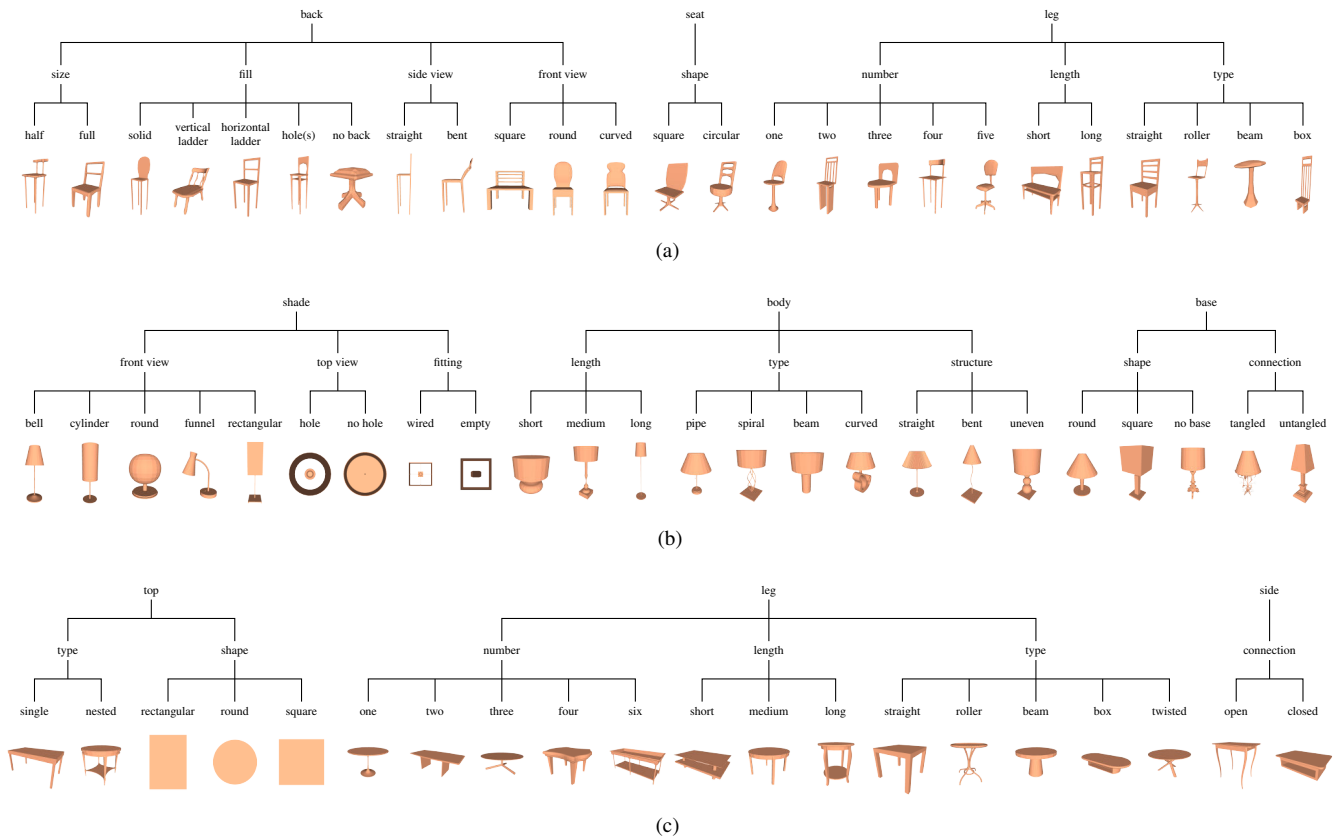


Figure 9: All the labels that we use to describe the visual attributes of (a) chairs, (b) lamps, and (c) tables, along with an example shape for each attribute.

[SSK*17] SUNG M., SU H., KIM V. G., CHAUDHURI S., GUIBAS L.: ComplementMe: Weakly-supervised component suggestions for 3D modeling. *ACM Trans. on Graphics* 36, 6 (2017), 226:1–226:12. 3

[TGY*09] TALTON J. O., GIBSON D., YANG L., HANRAHAN P., KOLTUN V.: Exploratory modeling with collaborative design spaces. *ACM Trans. on Graphics* 28, 5 (2009), 167:1–167:10. 2

[WAVK*12] WANG Y., ASAFI S., VAN KAICK O., ZHANG H., COHENOR D., CHEN B.: Active co-analysis of a set of shapes. *ACM Trans. on Graphics* 31, 6 (2012), 165:1–165:10. 6

[WHWW14] WANG W., HUANG Y., WANG Y., WANG L.: Generalized autoencoder: A neural network framework for dimensionality reduction. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition Workshops* (2014), pp. 496–503. 2, 5

[WLG*17] WANG P.-S., LIU Y., GUO Y.-X., SUN C.-Y., TONG X.: O-CNN: Octree-based convolutional neural networks for 3D shape analysis. *ACM Trans. on Graphics* 36, 4 (2017), 72:1–72:11. 2

[WLJ*18] WANG G., LAGA H., JIA J., XIE N., TABIA H.: Statistical modeling of the 3D geometry and topology of botanical trees. *Computer Graphics Forum* 37, 5 (2018), 185–198. 2

[WSCR18] WANG K., SAVVA M., CHANG A. X., RITCHIE D.: Deep convolutional priors for indoor scene synthesis. *ACM Trans. on Graphics* 37, 4 (2018), 70:1–70:14. 3

[WSK*15] WU Z., SONG S., KHOSLA A., YU F., ZHANG L., TANG X., XIAO J.: 3D ShapeNets: A deep representation for volumetric shapes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* (2015), pp. 1912–1920. 6

[WZX*16] WU J., ZHANG C., XUE T., FREEMAN W. T., TENENBAUM J. B.: Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In *Advances in Neural Information Processing Systems* (2016), pp. 82–90. 1, 2, 3

[WZX*20] WU R., ZHUANG Y., XU K., ZHANG H., CHEN B.: PQ-NET: A generative part Seq2Seq network for 3D shapes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* (2020), pp. 826–835. 3

[XZG*18] XIE J., ZHENG Z., GAO R., WANG W., ZHU S.-C., WU Y. N.: Learning descriptor networks for 3D shape synthesis and analysis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* (2018), pp. 8629–8638. 9

[YCHK15] YUMER M. E., CHAUDHURI S., HODGINS J. K., KARA L. B.: Semantic shape editing using deformation handles. *ACM Trans. on Graphics* 34, 4 (2015), 86:1–86:12. 3

[ZAESB20] ZEKUN H., AVERBUCH-ELOR H., SNAVELY N., BELONGIE S.: DualSDF: Semantic shape manipulation using a two-level representation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* (2020), pp. 7628–7638. 3

Appendix A: Label dataset

In total, we have 25 labels for the set of chairs, 24 labels for lamps, and 20 labels for tables, which are summarized in Figure 9. We manually selected the labels according to the visual properties of the shapes to evaluate our method.