Synthesizing 3D Shapes via Modeling Multi-View Depth Maps and Silhouettes with Deep Generative Networks

Amir Arsalan Soltani[1] Haibin Huang[2] Jiajun Wu[1] Tejas D. Kulkarni[3] Joshua B. Tenenbaum[1]
[1]Massachusetts Institute of Technology [2]University of Massachusetts, Amherst [3]Google DeepMind
{arsalans, jiajunwu, jbt}@mit.edu hbhuang@cs.umass.edu tejasdkulkarni@gmail.com

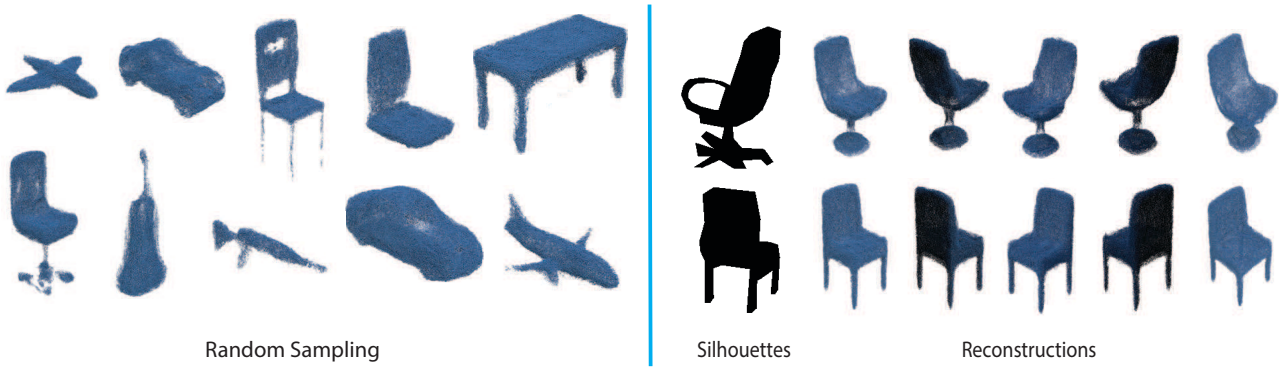

Figure 1: Left: Our method generates multi-view depth maps and silhouettes, and uses a rendering function to obtain the 3D shapes. Right: We can also extend our framework to reconstruct 3D shapes from single/multi-view depth maps or silhouettes.

Abstract

We study the problem of learning generative models of 3D shapes. Voxels or 3D parts have been widely used as the underlying representations to build complex 3D shapes; however, voxel-based representations suffer from high memory requirements, and parts-based models require a large collection of cached or richly parametrized parts. We take an alternative approach: learning a generative model over multi-view depth maps or their corresponding silhouettes, and using a deterministic rendering function to produce 3D shapes from these images. A multi-view representation of shapes enables generation of 3D models with fine details, as 2D depth maps and silhouettes can be modeled at a much higher resolution than 3D voxels. Moreover, our approach naturally brings the ability to recover the underlying 3D representation from depth maps of one or a few viewpoints. Experiments show that our framework can generate 3D shapes with variations and details. We also demonstrate that our model has out-of-sample generalization power for real-world tasks with occluded objects.

1. Introduction

What makes a good generative model of 3D shapes? We argue that the synthesized objects should be both realistic and novel. The generator should be able to capture the fine details of objects of different categories. It should also characterize the possible variations that shapes may have, generalizing beyond a fixed collection.

Traditional shape synthesis methods typically employ a template-based model, where objects and parts are from a large predefined repository. Shape synthesis is then essentially to memorize and recombine these parts [10, 3]. The template representation enables generating shapes of high-fidelity; however, it constrains the possibility of obtaining novel objects that are never observed before.

Recently, researchers explored sampling novel voxelized 3D shapes by modeling them via volumetric convolutional networks [28, 27, 6, 18]. Shapes obtained this way are highly varied; however, the use of voxel representation limits their resolutions due to the curse of dimensionality.

Here we consider an alternative approach: we first use deep generative networks to model and sample from the space of 2D images, specifically multi-view depth maps or silhouettes; we then employ a rendering function to synthe-

size 3D shapes from the sampled multi-view images. Compared to modeling 3D shapes via deep networks directly, our model brings us the unique advantages of generating highly varied shapes with much finer details, and simultaneously reconstructing 3D shapes from one or multiple views.

This is made possible because, first, a multi-view 2D representation still offers rich flexibility and allows generating detailed 3D shapes [17, 14]. Second, it is much easier to model the space of 2D images than the space of 3D shapes directly. In a lower dimensionality, models can characterize objects in higher resolution. In particular, we leverage the recently proposed variational autoencoder [12] to generate multi-view 2D image representations of resolution 224×224. In comparison, current volumetric networks that model 3D shapes directly currently only scale only up to $64 \times 64 \times$ [27].

Employing a 2D viewpoint-based representation naturally enables 3D reconstruction from one or a few depth maps or silhouettes. We evaluate our model on the NYU-D dataset as a real-world experiment, and validate that the model can generalize to recognize 3D shapes from 2D maps well and robustly. We also present analysis of the learned shape space through tasks like shape interpolation, and demonstrate applications including shape classification.

Our contributions are three-fold: first, we consider generative modeling of 3D shapes via multi-view 2D depth maps and silhouettes, and are able to generate highly varied shapes with fine details; second, we explore the learned shape space and its representation, and present results on both shape recognition and interpolation; third, we demonstrate that it is possible to extend our framework to real world silhouettes or depth maps, where objects may be cropped or occluded.

2. Related Work

There has been renewed interest in building generative models of 3D objects [27, 19, 8, 28]. Different approaches for modeling 3D shapes fall into three key categories: parts-based [8, 10, 1, 26], voxel-based [19, 27, 6, 28], and view based methods [18, 23]. Our proposed approach falls into the view based method category.

Modeling shapes via part-based models produces high resolution meshes. But it usually requires labeled 3D objects parts before training [10, 1], which are hard to obtain in many scenarios. It also lacks the creation ability as it generates shapes by retrieving and recombining database parts without further modifications. Huang *et al*. [8] built a generative model based on shapes structure and surfaces sample points distribution, but it is computationally expensive due to the need of dense point correspondence.

Deep generative models with voxel based representations relax the labeling requirement [28, 6, 19, 27]. In particular, Wu *et al*. [28] used deep Boltzmann machine to model and synthesize 3D shapes, Girdhar *et al*. [6] learned a joint embedding of 3D shapes and 2D images via autoencoders, and Wu *et al*. [27] modeled 3D shapes in a generative-adversarial manner. However, the higher dimension leads to a large number of elements in shape representation. This is one of the main limitations for scaling voxel based representations to more complex objects and scenes.

View based representations have demonstrated strong potential for both shape recognition and synthesis [23, 18, 5, 25, 16, 13, 29]. Tatarchenko *et al*. [25], Yan *et al*. [29], Choy *et al*. [4], and Park *et al*. [16] explored reconstructing 3D shapes from single or multi-view images, but their approach are mostly for reconstruction, not 3D shape synthesis. Kulkarni *et al*. [13] and Dosovitskiy *et al*. [5] explored generating shapes in multiple viewpoints, though they did not look into 3D shape synthesis, either.

Using viewpoints and deep generative models, we show in this paper that our approach is able to produce high resolution 3D shapes for tasks including generation, reconstruction, interpolation and out-of-sample generalization on a hold-out test set.

3. Approach

Our goal is to synthesize high-resolution, detailed 3D shapes; we also want to reconstruct the 3D shape given one or more view points but is also to sample new shapes randomly. At the core of our model is a variational auto-encoder, modeling the space of multi-view depth maps and silhouettes. We then reconstruct the 3D shapes from them.

Model Our model is primarily based on a variational auto-encoder [12]. Let us consider a dataset of depth or silhouette images $X = \{x_i\}_{i=1}^N$ drawn from an i.i.d distribution, where each x_i is a set containing depth or silhouette images from single or multiple views (up to 20 in our case). Assume that the data is drawn from some random process involving an unobserved random variable Z for the underlying 3D representation, and a class label C whenever specified.

The random variable Z is drawn from a prior distribution $p_{\theta*}(Z)$ and X is subsequently generated from some conditional distribution $p_{\theta*}(X|Z)$. The true model parameters $\theta*$ are unknown. We assume that the prior and likelihood are parametrized by differentiable functions $p_\theta(Z)$ and $p_\theta(X|Z, C)$. Following a variational gradient based algorithm proposed in [12], we are interested in estimating both Z and θ given raw data and labels whenever specified.

From Depth Maps to 3D Shapes In the final step, all depth maps are projected back to the 3D space to create the final rendering. We reconstruct 3D shapes from multi-view silhouettes and depth maps by first generating a 3D point cloud from each depth image with its corresponding camera setting. The union of these point clouds from all views can be seen as an initial estimation of the shape. We then refine it

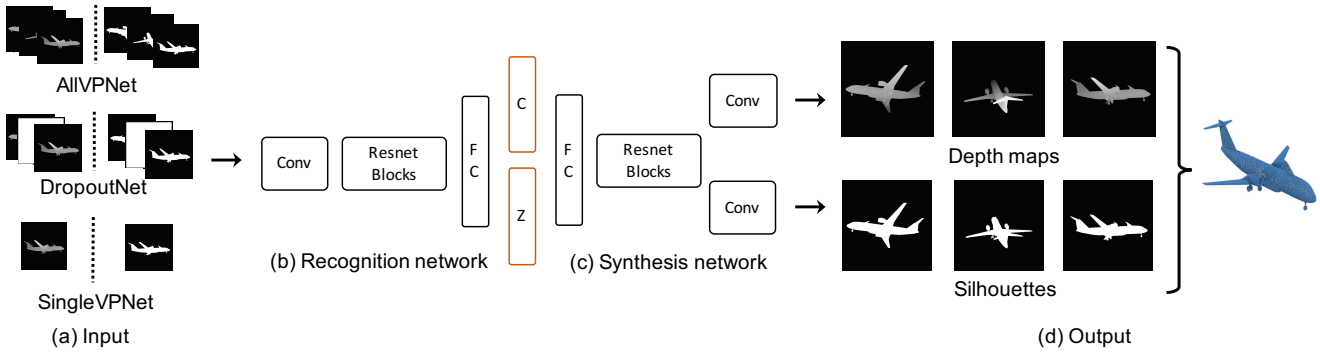

Figure 2: **Pipeline**: the input for our network can be depth maps or silhouettes. They will be fed to the network, which learns the latent variables $P(Z|S)$. The reconstructed depth maps and silhouettes are fused together to produce the final 3D point cloud. In the conditional network, we also fed in a one-hot vector.

by applying predicted silhouettes to filter out noise points. A point will be kept only if all of its multi-view 2D projections are valid in the silhouettes. Supervision on the camera angles or distance from the 3D shape centroid is not explicitly provided to the models. The only explicit supervision given is the class labels when training conditional models.

3.1. Networks

We use three main networks to do our experiments. The networks are called *AllVPNet*, *DropoutNet*, and *SingleVPNet*, depending on the input they receive. *AllVPNet* and *DropoutNet* take in either 20 depth maps or 20 silhouettes, rendered with fixed camera angles, as input. For *DropoutNet* we randomly zero-out (drop) between 15 to 18 of the view points of each sample before inputting them to the encoder of the model. The input to the *SingleVPNet* is a single depth or silhouette image chosen randomly from the 20 available views. For *SingleVPNet*, the identity of the views are unknown to the model.

All networks produce 20 view depth maps and silhouettes simultaneously in the output. For each network, we use either depth maps or silhouettes as input, resulting in 6 networks for our experiments. Each of those networks can be trained unsupervisedly or conditionally, resulting in 12 networks in total in all of our experiments. All networks are expected to learn the 3D representations of objects $P(Z|x)$ given all, multiple, or single view inputs.

3.2. Architecture

For the encoder, we use the B-type (projection) residual blocks [7] that project down the feature maps of the previous layer. Each residual block consists of 2 layers within itself. We use dilated convolutional modules [30] for both residual blocks and non-residual convolutional layers (for downsampling only). The first layer of the encoder uses a convolutional layer and the last layer regresses on the mean and log-variances of the prior distribution with two(three for conditional models) FC layers. For the conditional models, the last layer also regresses on the class scores with another FC layer.

The decoder uses the B-type residual blocks for the first 3 layers. However, instead of downsampling the feature maps, it implements the transposed convolutional operation and then switches to strided transposed convolutional layers [22] for the last 3 layers.

All convolutional layers in both the decoder and the encoder use 4×4 filters, except the modules within each residual block which use 3×3 as their filter size. The number of parameters grows significantly as B-Type residual blocks are used for most layers. Therefore, we use the factorized convolutional filters proposed in [24] for all layers in the encoder and decoder, except for the last 3 layers of the decoder. This way we can reduce the number of parameters by a factor of about 1.6. The maximum number of feature maps is not more than 74×8 for the layers in the middle. The first and last layer of both encoder and decoder have 74×4 channels. All layers except the fully connected layers use batch-normalization [9].

For unconditional models, we aim to minimize the following loss functions during training,

$$\mathcal{L}(x_i, \phi, \theta) = -D_{KL}(q_\phi(Z|x_i)||p_\theta(Z)) + \mathrm{E}_{q_\phi(Z|x_i)}[\log p_\theta(x_i|Z)], \quad (1)$$

and the loss function for the conditional model is

$$\mathcal{L}(X, \phi, \theta, C) = -D_{KL}(q_\phi(Z|x_i, c^{(i)})||p_\theta(Z)) + \mathrm{E}_{q_\phi(Z|x_i)}[\log p_\theta(x_i|Z, c_i)] - c_i^{\text{target}} \log(c_i^{\text{pred}}). \quad (2)$$

The first term computes the KullbackLeibler divergence of the approximating distribution to the Normal prior distribution. The second term denotes the reconstruction error, and the third term in the conditional model refers to the classification loss. Note that the reconstruction loss function is

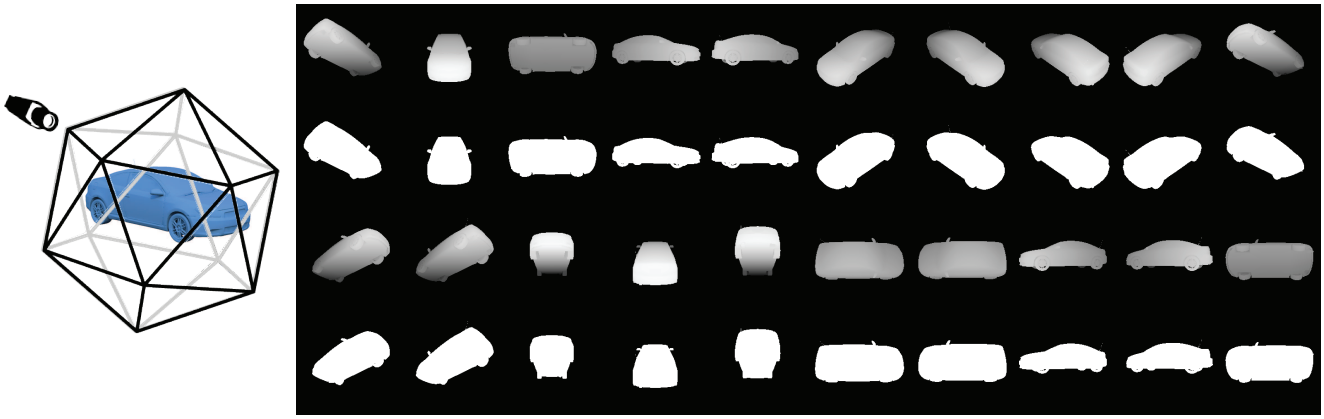

Figure 3: **Training data.** Left: we render a 3D model from ShapeNet [2] with 20 viewpoints. *First and third rows*: depth maps; *second and fourth rows*: silhouettes. *First and second rows*: Views 1 to 10; *Third and fourth rows*: Views 11 to 20

in fact composed of two loss functions: 1) The depth map reconstruction error, and 2) the silhouettes reconstruction error. For the KL divergence loss, we multiply its gradient values for means and variances by 75 to push the approximation distribution towards the prior and balance the sampling diversity.

3.3. Training

For the conditional models the Z vector is concatenated with a hot-vector C, which has a length corresponding to the number of categories in the data set. Then the $Z+C$-element vector is fed to the decoder as shown in 2. During training, we use the ground truth class labels, but during testing, we feed the predicted class labels by the model to the decoder, and do no use the ground-truth labels.

We use ADAM [11] for optimization. We use a learning rate around 5×10^{-6} with annealing. The reason for using small learning rate is due to using the raw pixel-wise reconstruction error values for both depth and silhouette images. We use ℓ_1 distance to compute the errors for the generated depth and silhouette images. The training starts with a a batch size of 4, increases by 2 every 20 epochs, but does not go beyond 8. We empirically noticed that small batch size and small initial learning rates help more in getting less average-looking 3D shapes and help lower losses.

4. Experiments

In this section, we show the results of our experiments for conditional and unconditional 3D shape generation, reconstruction, classification, and out of sample generalization.

4.1. Setup

We train all of our models on the ShapeNet Core [2] data set which consists of aligned 3D models. We use all data for all categories in ShapeNet Core and divide them into train (92.5%), validation (7.5%) each containing 37,892, 3,070 shapes (3D models) respectively.

For each 3D model in the data set, we setup multiple viewpoints to get the depth images as shown in Figure 3. The renderings are generated by placing 20 virtual cameras at 20 vertices of a dodecahedron enclosing the shape. All cameras point towards the centroid of the mesh. The centroid is calculated as average of the mesh face centers, weighted by the face areas. The silhouettes are obtained by binarizing the depth images. For all our experiments, we use rendering images of size 224×224. Training takes about two days.

We also test our *SingleVPNet* model, trained with silhouettes, on the extracted chair examples of the NYU-D [21] data set, and show that our model is capable of out-of-sample generalization. From the pixel-level labels of the NYU-D data set, we extract depth maps of chairs, and place them in the middle of a canvas of size 224×224. Most samples are occluded in addition to not having a fixed camera angle and view point, introducing a challenge to our model.

4.2. 3D Shape Generation

After our models have been trained, we use them for generating unconditional and conditional samples. Figure 4 highlights generated samples from our model. We also highlight some conditional samples in Figure 5. Similar to [5] and [19], the shape category prior is induced through a one-hot vector with cardinality equal to the number of classes when drawing samples.

4.3. Reconstruction

The model can reconstruct the 3D model when it is fed all views. However, an interesting use case is when we may not have access to all view points. In Figure 6, we show that our model can reconstruct the full 3D model given limited view points. To obtain these results we use DropoutNet for which, while training and testing, 15 to 18 views of the input

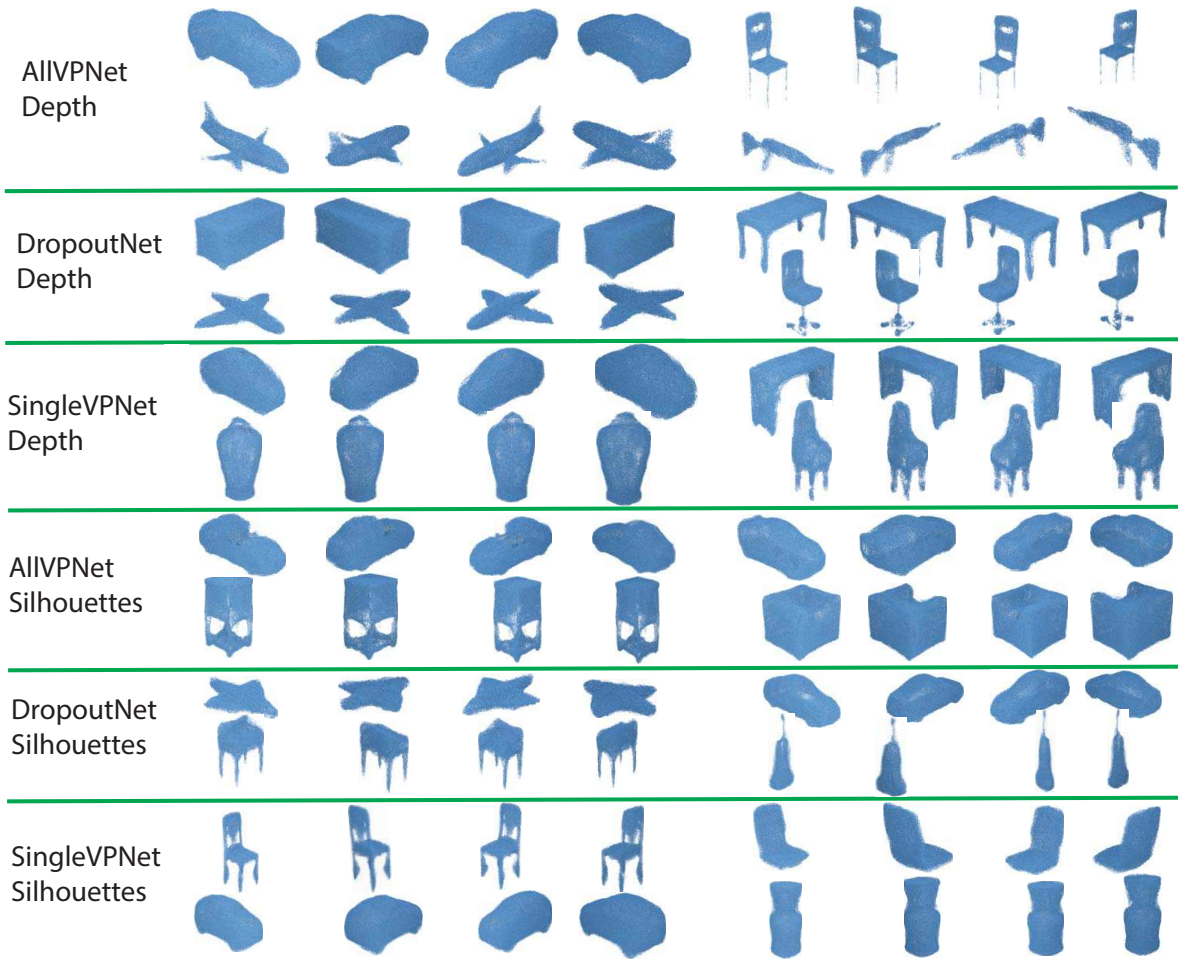

Figure 4: **Unconditional Sampling**: After training our model on training samples on all ShapeNet categories, we draw unconditional samples from the prior $P(Z)$, and feed them to the decoder to get the 3D models. The figure shows models produced by our AllVPNet, DropoutNet, and SingleVPNet, trained with depth maps or silhouettes.

are randomly zeroed-out.

We also test the ability of our pipeline in obtaining the underlying 3D representation and reconstructing the full 3D model given a single view point. Note that the model has not been fed any extra information about the underlying class or the angle of the input view point. We also do not clamp the hidden variables to ease the inference computation as in [28]. The input view point may not be very informative (*e.g.*, looking at a sofa handle orthogonally), making it both harder for the model to infer the underlying 3D representation and more challenging to reconstruct the 3D object perfectly due to ambiguities, quantitatively shown in Table 4. As presented in Figure 6, our model is capable of reconstructing back the underlying 3D model with acceptable quality given only one view point.

To show the out-of-sample generalization of our model, we show that it can recover the 3D models for the chairs in the NYU-D data set [21], without fine-tuning. The NYU-D dataset introduces another challenge, as the camera position is no longer pre-selected. Therefore, we can test how well the model has learned about different angles implicitly. We show the results in Figure 8.

	AllVP		Dropout		SingleVP	
	Depth	Sil.	Depth	Sil.	Depth	Sil.
Uncond.	84.0	83.6	78.5	77.4	70.7	66.8
Cond.	83.9	83.5	78.5	78.1	72.4	67.9

Table 1: We compute IoU of reconstructions in 3D voxels, on our test set of ShapeNet [2], for our models trained on silhouettes and depth maps conditionally and unconditionally

For a quantitative evaluation, we compare our reconstructions results in intersection over union (IoU) in Table 1 using 32^3 voxels, and in mean ℓ_1 reconstruction losses in Table 3.

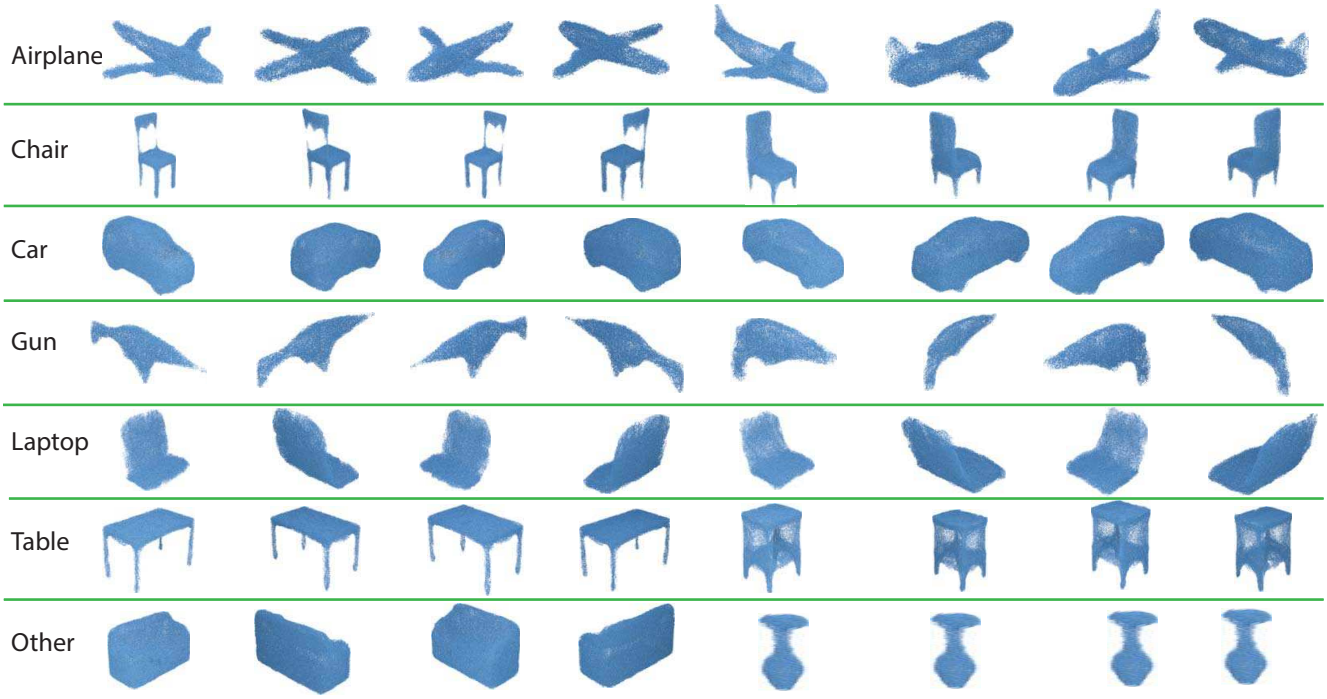

Figure 5: **Conditional Sampling**: we show random samples generated by drawing samples given a class label.

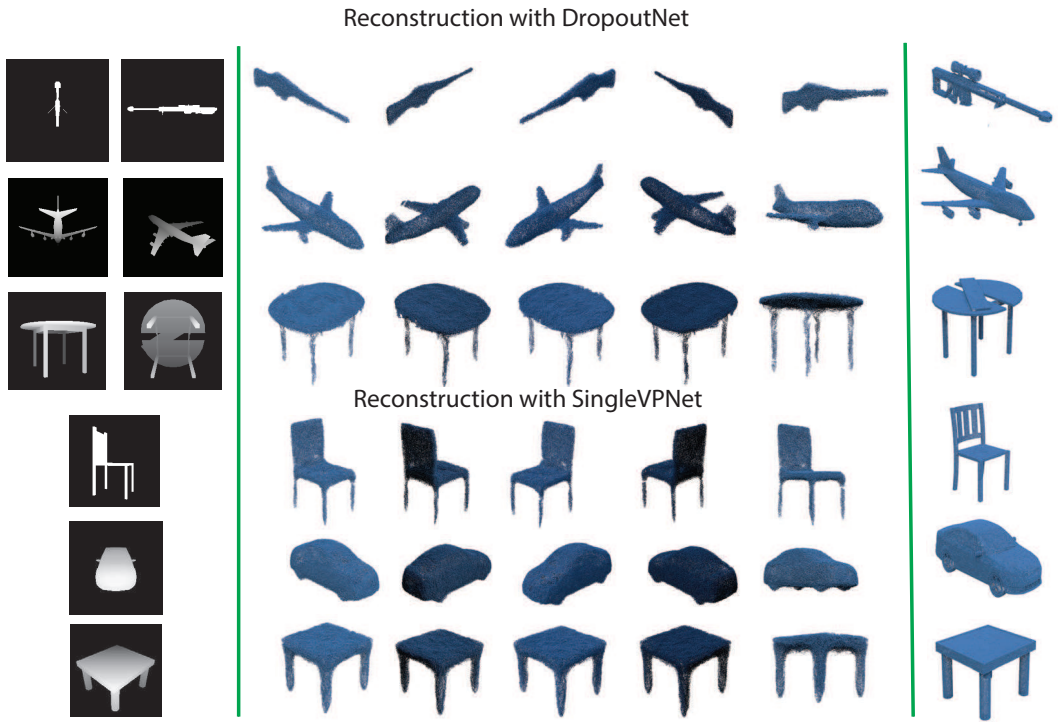

Figure 6: **3D Reconstruction from limited views**: our DropoutNet and SingleVPNet can reconstruct samples from either depth maps or silhouettes of a few or one viewpoint(s). **Top-half**: from left to right, the two input to DropoutNet, the inferred 3D reconstructions shown in different views, the ground truth model. **Bottom-half**: from left to right, the input to SingleVPNet, the inferred 3D reconstructions shown in different views, the ground truth model.

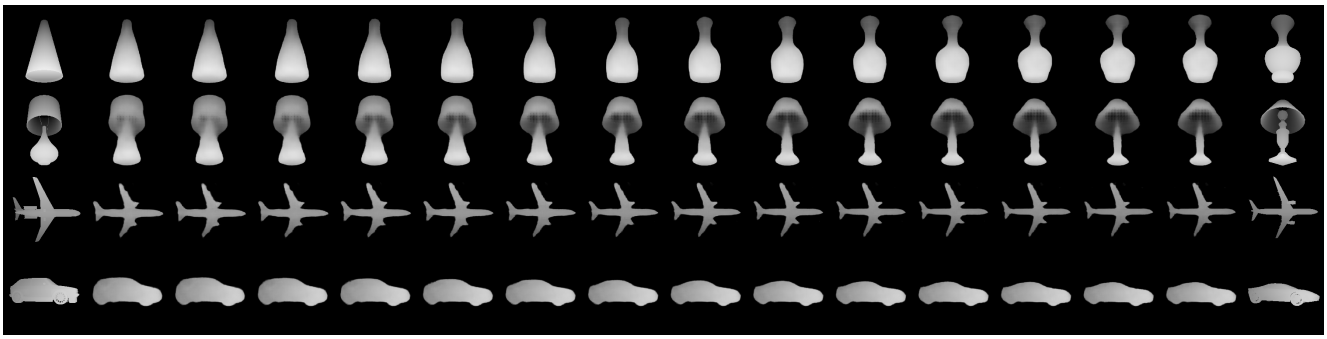

Figure 7: **Interpolation**: the learned model can also be used for interpolation. We use linear interpolation between two models' latent vectors, and feed them into the decoder network to synthesis new models views. In each row, the left most and the right most columns show the original models. We choose different view point for each row. We can see the images smoothly transform from left to right.

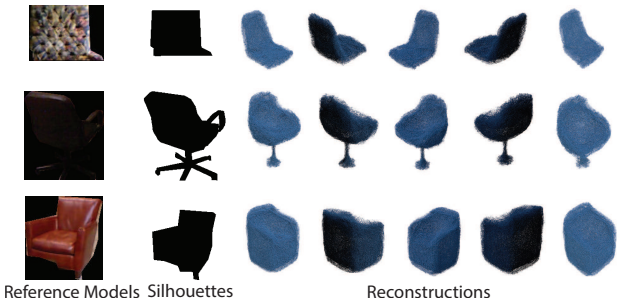

Figure 8: **3D Reconstruction from real images**: we show how our SingleVP model can reconstruct 3D models from a single-view silhouette of the NYU-D [21] data set. From left to right: raw RGB images, input silhouettes, the inferred 3D reconstructions in different views.

Models	Representation	Accuracy (%)
DeepPano [20]	panorama	78%
3D ShapeNet [28]	voxel	77%
VoxNet [15]	voxel	83%
MVCNN [23]	multi-view	90%
AllVPNet	multi-view	$82.1\% \pm 0.1$
DropoutNet	multi-view	$74.2\% \pm 0.2$
SingleVPNet	single-view	$65.3\% \pm 0.3$

Table 2: The classification accuracy on the test set of ModelNet40 [28]. Trained on depth maps, our framework achieves comparable results with other supervised learning methods like VoxNet [15]. MVCNN [23] also employed a multi-view representation, but they used ImageNet-pretrained networks and RGB images, and achieved better performance.

4.4. Classification

With the conditional models, we get class predictions and may compute classification accuracies. Although our goal is not to train a classifier, we evaluate our frameworks on shape classification, with depth maps or silhouettes as input. In Table 3, we summarize the classification accuracy results obtained after training our networks on all ShapeNet categories [2].

Network	Training Set		Test Set		Acc. (%)
	Depth	Sil.	Depth	Sil.	
AllVP	0.014	0.016	0.016	0.019	89.1 ± 0.1
Dropout	0.022	0.026	0.023	0.028	85.5 ± 0.1
SingleVP	0.028	0.036	0.029	0.036	82.7 ± 0.1
AllVP	0.015	0.016	0.017	0.019	87.6 ± 0.1
Dropout	0.022	0.026	0.023	0.028	84.9 ± 0.1
SingleVP	0.032	0.039	0.033	0.030	80.0 ± 0.2

Table 3: Classification accuracy and mean reconstruction errors in ℓ_1 norm, averaged over all 3D shapes, all 20 views, and all pixels for conditional models, on our test set of ShapeNet [2]. The errors appear to be small due to large empty background. Networks on the top were trained with depth and networks on bottom with silhouettes

We further evaluated our framework on the standard shape classification benchmark ModelNet40 [28]. As shown in Table 2, our framework, though with depth maps as input, achieves comparable performance with other supervised learning methods [15, 28]. MVCNN [23] also employed a multi-view representation, but they used ImageNet-pretrained networks and RGB images, and achieved better performance.

4.5. Analysis

Interpolation To see how well our models have learned the underlying manifold of the data, we do linear interpolation in the latent space between two randomly-selected samples from the same category, and reconstruct 3D models. We show results obtained through the *SingleVPNet* model in Figure 7. Our framework is able to interpolate between

Viewpoint	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Depth Err.	.031	.035	.036	.036	.038	.040	.040	.028	.029	.038	.038	.028	.029	.029	.029	.030	.029	.030	.030	.030
Sil. Err	.038	.043	.046	.046	.047	.050	.049	.035	.035	.047	.047	.035	.035	.036	.037	.037	.036	.037	.037	.037
Accuracy	82	82	81	80	79	79	79	84	83	78	78	85	83	85	84	84	84	84	84	83
IoU	74.2	70.4	69.9	70.1	70.0	68.3	68.4	75.0	75.4	68.1	68.4	75.0	75.4	74.7	74.6	74.5	74.7	74.3	74.4	75.5

Table 4: ℓ_1 reconstruction loss, classification accuracy (%), and IoU on our test set of ShapeNet [2], using *SingleVPNet* for each view. IoU and classification numbers drop more for view points with more ambiguity, where most parts of the objects are not visible. Still, our model is able to infer consistent representations for most views

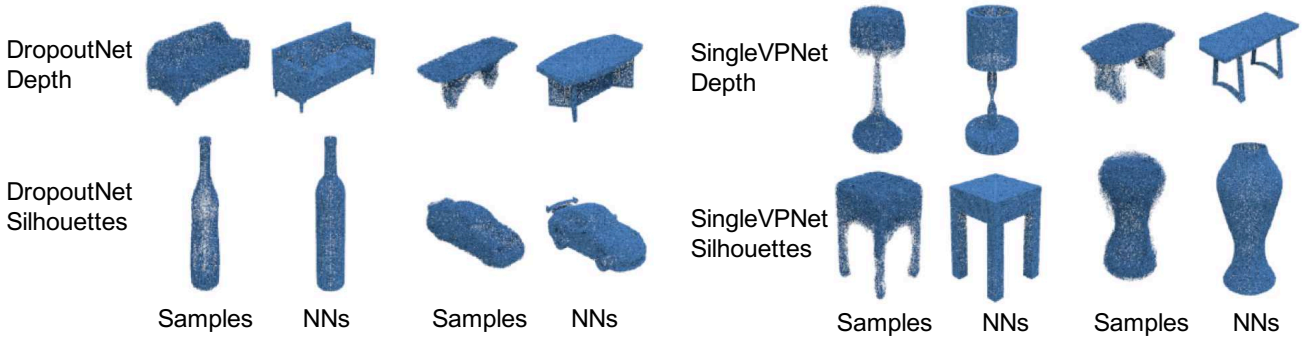

Figure 9: **Nearest neighbor examples to generated 3D models**: our model produces 3D models that are not identical to the closest training set samples.

objects and obtain smooth transitions in the space of depth maps.

Representation Consistency Our SingleVPNet takes a single image as input. In Table 4, we show quantitative evaluations when the inputs are from certain viewpoints, in ℓ_1 reconstructions, classfication accuracy, and intersection over union (IoU). We see that our model is robust to different views, and achieves consistent results.

Nearest Neighbors To verify whether the models are just memorizing training data, we show nearest neighbor results in Figure 9. We see that the generated shapes are different from its nearest neighbor in the training set.

5. Conclusion

We have explored a new paradigm for 3D shape generative modeling. Instead of modeling voxelized 3D objects directly, we instead employ deep generative models for multi-view depth maps and silhouettes and then rendering 3D objects from these 2D images. Our framework is able to generate 3D shapes that are both novel and with details. We have also demonstrated that it has applications in 3D shape reconstruction and recognition.

Acknowledgements This work is supported by ONR MURI N00014-16-1-2007, the Center for Brain, Minds and Machines (NSF STC award CCF-1231216), and grants from the Toyota Research Institute and Adobe Research. We thank Ilker Yildirim, Danilo J. Rezende, and Evangelos Kalogerakis for helpful suggestions and advice.

References

[1] M. Averkiou, V. G. Kim, Y. Zheng, and N. J. Mitra. Shapesynth: Parameterizing model collections for coupled shape exploration and synthesis. *CGF*, 33(2):125–134, 2014. 2

[2] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 4, 5, 7, 8

[3] S. Chaudhuri, E. Kalogerakis, L. Guibas, and V. Koltun. Probabilistic reasoning for assembly-based 3d modeling. In *SIGGRAPH*, 2011. 1

[4] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *ECCV*, 2016. 2

[5] A. Dosovitskiy, J. T. Springenberg, M. Tatarchenko, and T. Brox. Learning to generate chairs, tables and cars with convolutional networks. *IEEE TPAMI*, 39(4):692–705, 2017. 2, 4

[6] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta. Learning a predictable and generative vector representation for objects. In *ECCV*, 2016. 1, 2

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2015. 3

[8] H. Huang, E. Kalogerakis, and B. Marlin. Analysis and synthesis of 3D shape families via deep-learned generative models of surfaces. *CGF*, 34(5):25–38, 2015. 2

[9] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 3

[10] E. Kalogerakis, S. Chaudhuri, D. Koller, and V. Koltun. A probabilistic model for component-based shape synthesis. *ACM TOG*, 31(4):55, 2012. 1, 2

[11] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014. 4

[12] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *ICLR*, 2014. 2

[13] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum. Deep convolutional inverse graphics network. In *NIPS*, 2015. 2

[14] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. *IJCV*, 38(3):199–218, 2000. 2

[15] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IROS*, 2015. 7

[16] E. Park, J. Yang, E. Yumer, D. Ceylan, and A. C. Berg. Transformation-grounded image generation network for novel 3d view synthesis. In *CVPR*, 2017. 2

[17] M. Potmesil. Generating octree models of 3d objects from their silhouettes in a sequence of images. *CVGIP*, 40(1):1–29, 1987. 2

[18] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *CVPR*, 2016. 1, 2

[19] D. J. Rezende, S. Eslami, S. Mohamed, P. Battaglia, M. Jaderberg, and N. Heess. Unsupervised Learning of 3D Structure from Images. In *NIPS*, 2016. 2, 4

[20] B. Shi, S. Bai, Z. Zhou, and X. Bai. Deeppano: Deep panoramic representation for 3-d shape recognition. *IEEE SPL*, 22(12):2339–2343, 2015. 7

[21] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 4, 5, 7

[22] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014. 3

[23] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *ICCV*, 2015. 2, 7

[24] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *arXiv preprint arXiv:1512.00567*, 2015. 3

[25] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Multi-view 3d models from single images with a convolutional network. In *ECCV*, 2016. 2

[26] J. Wu, T. Xue, J. J. Lim, Y. Tian, J. B. Tenenbaum, A. Torralba, and W. T. Freeman. Single image 3d interpreter network. In *ECCV*, 2016. 2

[27] J. Wu, C. Zhang, T. Xue, W. T. Freeman, and J. B. Tenenbaum. Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling. In *NIPS*, 2016. 1, 2

[28] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015. 1, 2, 5, 7

[29] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In *NIPS*, 2016. 2

[30] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016. 3