Learning Descriptor Networks for 3D Shape Synthesis and Analysis

[1] Jianwen Xie*, [2] Zilong Zheng*, [2] Ruiqi Gao,
[3] Wenguan Wang, [2] Song-Chun Zhu, [2] Ying Nian Wu

[1] Hikvision Research Institute, USA
[2] University of California, Los Angeles, USA
[3] Beijing Institute of Technology, China
*(equal contribution)

June 21, 2018

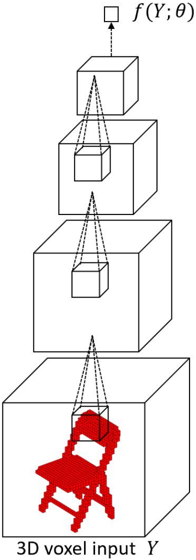
Introduction

Introduction

1. With the introduction of **large 3D CAD dataset**, some interesting attempts have been made on 3D object recognition and synthesis.
2. We have witnessed impressive progress on developing **3D discriminator** (for classification) and **3D generator** (for synthesis), however, there has not been much work in modeling 3D data based on energy-based models (descriptive model).
3. The focus of the paper is to develop **3D deep convolutional energy-based model** (3D Descriptor Net) for 3D voxelized data. (Alternative to 3D GAN)

Model and learning algorithm

Probability density



The model is a probability density distribution defined on volumetric data Y:

$$p(Y;\theta) = \frac{1}{Z(\theta)} \exp\left[f(Y;\theta)\right] p_0(Y),$$

where $f(Y;\theta)$ is a 3D bottom-up ConvNet structure, $Z(\theta)$ is the normalizing constant,

$$Z(\theta) = \int \exp\left[f(Y;\theta)\right] p_0(Y) dY$$

and $p_0(Y)$ is the reference distribution such as Gaussian white noise,

$$p_0(Y) \propto \exp\left[-\frac{\|Y\|^2}{2s^2}\right]$$

Energy-based form

The 3D Descriptor net can be written as the form of energy-based model:

$$p(Y;\theta) = \frac{1}{Z(\theta)} \exp\left[-\mathcal{E}(Y;\theta)\right],$$

where the **energy function** is:

$$\mathcal{E}(Y;\theta) = \frac{\|Y\|^2}{2s^2} - f(Y;\theta).$$

Term $\frac{\|Y\|^2}{2s^2}$ is from Gaussian distribution.

Learning by maximum likelihood estimation

[1] Jianwen Xie*, [2] Zilong Zheng*, [2] Ruiqi Gao, [3] Wenguan Wang, [2] Song-Chun Zhu, [2] Ying Nian Wu

Suppose we observe training examples $\{Y_i, i = 1, ..., n\}$.

The maximum likelihood estimation (MLE) seeks to maximize the log-likelihood function:

$$L_p(\theta) = \frac{1}{n} \sum_{i=1}^{n} \log p(Y_i; \theta).$$

The gradient of the log-likelihood is:

$$L_p'(\theta) \;\; = \;\; \mathrm{E}_\theta\left[\frac{\partial}{\partial\theta}\mathcal{E}(Y;\theta)\right] - \frac{1}{n}\sum_{i=1}^{n}\frac{\partial}{\partial\theta}\mathcal{E}(Y_i;\theta),$$

where E_θ denotes expectation with respect to $p(Y;\theta)$. The expectation term is due to $\frac{\partial}{\partial\theta}\log Z(\theta) = \mathrm{E}_\theta\left[\frac{\partial}{\partial\theta}\mathcal{E}(X;\theta)\right]$, which is analytically intractable. (MCMC needed!)

Sampling by Langevin dynamics

We use MCMC (e.g., Langevin dynamics) to sample $\{\tilde{Y}_i, i = 1, ..., \tilde{n}\}$ from $p(Y;\theta) \propto \exp[-\mathcal{E}(Y;\theta)]$.

One step of Langevin revision:

$$Y_{\tau+1} = \underbrace{Y_\tau - \frac{\delta^2}{2}\frac{\partial}{\partial Y}\mathcal{E}(Y_\tau;\theta)}_{\text{find } Y \text{ to minimize } \mathcal{E} \text{ via gradient descent}} + \underbrace{\mathrm{N}(0, \delta^2 I_D)}_{\text{brownian motion}}$$

MCMC approximation of the gradient:

$$\begin{aligned} L'_p(\theta) &= \mathrm{E}_\theta\left[\frac{\partial}{\partial\theta}\mathcal{E}(Y;\theta)\right] - \frac{1}{n}\sum_{i=1}^{n}\frac{\partial}{\partial\theta}\mathcal{E}(Y_i;\theta) \\ &\approx \underbrace{\frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}}\frac{\partial}{\partial\theta}\mathcal{E}(\tilde{Y}_i;\theta)}_{\text{synthesized statistics}} - \underbrace{\frac{1}{n}\sum_{i=1}^{n}\frac{\partial}{\partial\theta}\mathcal{E}(Y_i;\theta)}_{\text{observed statistics}} \end{aligned}$$

We alternate (1) sampling step and (2) model update step.

Alternating back-propagation

One step of Langevin revision:

$$
\begin{aligned}
Y_{\tau+1} &= Y_\tau - \frac{\delta^2}{2}\frac{\partial}{\partial Y}\mathcal{E}(Y_\tau;\theta) + \mathrm{N}(0, \delta^2 I_D) \\
&= Y_\tau - \frac{\delta^2}{2}\left[\frac{Y_\tau}{s^2} - \frac{\partial}{\partial Y} f(Y_\tau;\theta)\right] + \mathrm{N}(0, \delta^2 I_D)
\end{aligned}
$$

MCMC approximation of the gradient:

$$
\begin{aligned}
L_p'(\theta) &\approx \frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}}\frac{\partial}{\partial\theta}\mathcal{E}(\tilde{Y}_i;\theta) - \frac{1}{n}\sum_{i=1}^{n}\frac{\partial}{\partial\theta}\mathcal{E}(Y_i;\theta) \\
&= \frac{1}{n}\sum_{i=1}^{n}\frac{\partial}{\partial\theta} f(Y_i;\theta) - \frac{1}{\tilde{n}}\sum_{i=1}^{\tilde{n}}\frac{\partial}{\partial\theta} f(\tilde{Y}_i;\theta)
\end{aligned}
$$

Alternating back-propagation:
(1) Sampling back-propagation; (2) Learning back-propagation

Adversarial Interpretation of the learning process

$$L_p^{'}(\theta) \approx \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \frac{\partial}{\partial \theta} \mathcal{E}(\tilde{Y}_i;\theta) - \frac{1}{n} \sum_{i=1}^{n} \frac{\partial}{\partial \theta} \mathcal{E}(Y_i;\theta)$$

$$L_p^{'}(\theta) \approx \frac{\partial}{\partial \theta} \underbrace{\left[\frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} \mathcal{E}(\tilde{Y}_i;\theta) - \frac{1}{n} \sum_{i=1}^{n} \mathcal{E}(Y_i;\theta) \right]}_{V(\{\tilde{Y}_i\},\theta)}$$

- The sampling step finds $\{\tilde{Y}_i\}$ to **decrease** V, because it searches for low energy modes in the landscape defined by $\mathcal{E}(\mathcal{Y};\theta)$ via stochastic gradient descent.
- The learning step finds θ to **increase** V, which can be interpreted as mode shifting by shifting the low energy modes from the synthesized examples $\{\tilde{Y}_i\}$ toward the observed examples $\{Y_i\}$.

$$\max_{\theta} \min_{\{\tilde{Y}_i\}} V(\{\tilde{Y}_i\};\theta)$$

[1] Jianwen Xie*, [2] Zilong Zheng*, [2] Ruiqi Gao, [3] Wenguan Wang, [2] Song-Chun Zhu, [2] Ying Nian Wu

Applications

App 1: 3D shape synthesis

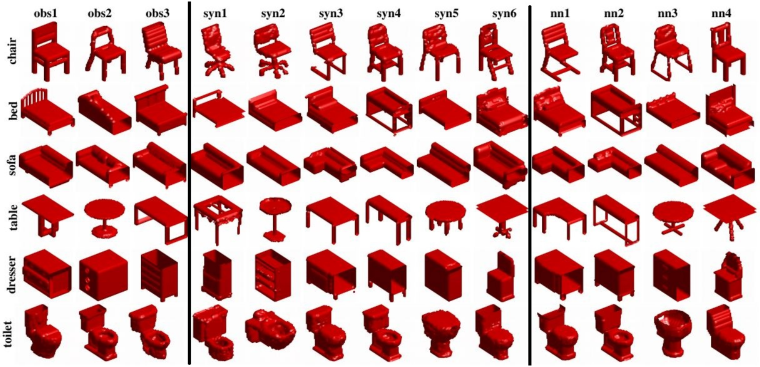


Figure: Each row displays one experiment, where the first three 3D objects are some observed examples, columns 4-9 are 6 of the 100 synthesized 3D objects. The nearest neighbors retrieved from the training set are shown for the last 4 synthesized objects.

App 1: 3D shape synthesis

Table: Inception scores of different methods on learning from 10 3D object categories.

Method	Inception score
3D ShapeNets	4.126±0.193
3D GAN	8.658±0.450
3D VAE	11.015±0.420
3D Descriptor Net (ours)	**11.772±0.418**

App 1: 3D shape synthesis

Table: Softmax class probability

	ours	3D GAN	3D VAE	3D ShapeNets
bathtub	**0.8348**	0.7017	0.7190	0.1644
bed	**0.9202**	0.7775	0.3963	0.3239
chair	**0.9920**	0.9700	0.9892	0.8482
desk	**0.8203**	0.7936	0.8145	0.1068
dresser	**0.7678**	0.7010	0.1049	0.2166
monitor	**0.9473**	0.2493	0.8559	0.2767
night stand	**0.7195**	0.6592	0.5426	0.4969
sofa	**0.9480**	0.9276	0.3017	0.4888
table	**0.8910**	0.8377	0.8751	0.7902
toilet	**0.9701**	0.8569	0.6943	0.8832
Avg.	**0.8811**	0.7431	0.7006	0.4596

App 2: 3D shape recovery

We can perform recovering onccluded data by sampling from conditional 3D DescriptorNet:

$$p(Y_M | Y_{\tilde{M}}, \theta),$$

where Y_M and $Y_{\tilde{M}}$ are the masked (occluded) parts and unmasked (visible) parts of the 3D shape.

Keys:

- Training: We train the conditional model from fully observed training pairs $\{(Y^i_M, Y^i_{\tilde{M}}), i = 1, ..., n\}$.
- Sampling: We run the Langevin revision starting from the occluded data, and fix the visible part and only update the occluded part.

App 2: 3D shape recovery

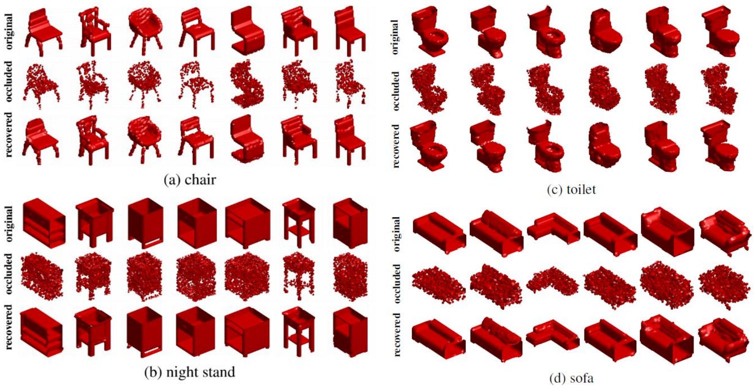


Figure: 3D shape recovery. (70% occlusion)

App 2: 3D shape recovery

Table: Recovery errors in occlusion experiments

	ours	3D GAN	3D ShapeNet
bathtub	**0.0152**	0.0266	0.0621
bed	**0.0068**	0.0240	0.0617
chair	**0.0118**	0.0238	0.0617
desk	**0.0122**	0.0298	0.0731
dresser	**0.0038**	0.0384	0.1558
monitor	**0.0103**	0.0220	0.0783
night stand	**0.0080**	0.0248	0.2925
sofa	**0.0068**	0.0186	0.0563
table	**0.0051**	0.0326	0.0340
toilet	**0.0119**	0.0180	0.0977
Avg.	**0.0085**	0.0258	0.0993

App 3: 3D object super-resolution

We can perform super-resolution by sampling from conditional 3D DescriptorNet:

$$p(Y_{high}|Y_{low};\theta),$$

where Y_{high} and Y_{low} are the high resolution version of Y and low resolution version of Y respectively.

Keys:

- Training: We train the conditional model from fully observed training pairs $\{(Y_{high}^i, Y_{low}^i), i = 1, ..., n\}$.
- Sampling: In each iteration, we first up-scale Y_{low} by expanding each voxel into $d \times d \times d$ block of constant intensity to obtain an up-scaled version Y'_{high} of Y_{low} and then run Langevin dynamics starting from Y'_{high} to obtain Y_{high}.

App 3: 3D object super-resolution

[1] Jianwen Xie*, [2] Zilong Zheng*, [2] Ruiqi Gao, [3] Wenguan Wang, [2] Song-Chun Zhu, [2] Ying Nian Wu

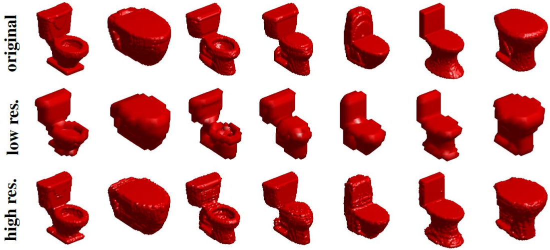
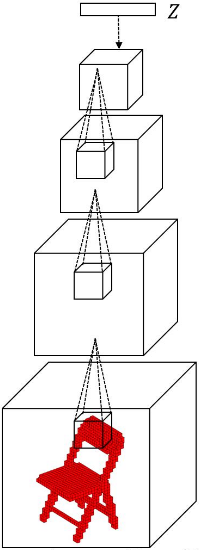


Figure: 3D object super-resolution by conditional 3D DescriptorNet. The first row displays some original 3D objects ($64 \times 64 \times 64$). The second row displays the corresponding low resolution 3D objects ($16 \times 16 \times 16$). The third row displays the corresponding super-resolution results.

App 4: Teaching 3D generator net



3D voxel output $Y = g(Z; \alpha)$

The 3D generator net [Kingma, Welling, 2013 on VAE; Goodfellow, et al. 2014 on GAN] is in the form of

$$Z \sim \mathrm{N}(0, I_d),$$
$$Y = g(Z; \alpha) + \epsilon, \ \epsilon \sim \mathrm{N}(0, \sigma^2 I_D).$$

$g(Z; \alpha)$ is a 3D top-down ConvNet, and α denotes the parameters of the generator.

App 4: Teaching 3D generator net

Cooperative Learning algorithm via MCMC teaching

Input:

training examples $\{Y_i, i = 1, ..., n\}$

Output:

(1) parameters θ and α, (2) synthetic data $\{\hat{Y}_i, \tilde{Y}_i, i = 1, ..., \tilde{n}\}$

Let $t \leftarrow 0$, initialize θ and α.

repeat

(1) For $i = 1, ..., \tilde{n}$, generate $Z_i \sim \mathrm{N}(0, I_d)$, and generate $\hat{Y}_i = g(Z_i; \alpha^{(t)}) + \epsilon_i$.

(2) For $i = 1, ..., \tilde{n}$, starting from $\hat{Y}_i$, run l steps of Langevin revision dynamics to obtain $\tilde{Y}_i$.

(3) Update $\theta^{(t+1)} = \theta^{(t)} + \gamma_t L'_p(\theta^{(t)})$.

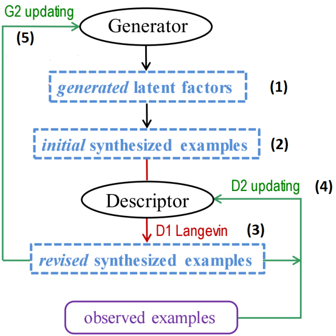
(4) Update $\alpha^{(t+1)}$ by gradient descent on $\sum_{i=1}^{\tilde{n}} \|\tilde{Y}_i - g(Z_i; \alpha^{(t)})\|^2$.

Let $t \leftarrow t + 1$

until $t = T$

App 4: Teaching 3D generator net

Cooperative learning / MCMC teaching / knowledge distillation



(Step 1,2) Generator → initial draft;

(Step 3) Descriptor → revised draft

(Step 4) Descriptor shifts from revised towards observed examples

(Step 5) Generator reconstructs the revised, knowing latent factors.

App 4: Teaching 3D generator net

We learn smooth generator model that traces the manifold of the 3D data distribution.

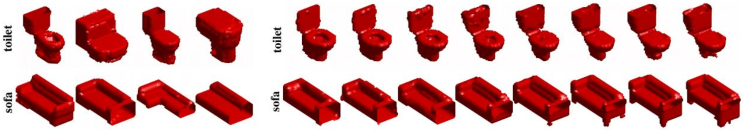


Figure: Left: 3D shape synthesis by 3D generators. Right: Interpolation between latent vectors of the 3D objects on two ends.

It encodes semantic knowledge of 3D shapes in the latent space

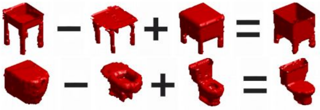


Figure: 3D shape arithmetic

App 5: 3D shape classification

(1) **unsupervised feature learning**: we train a single model from all categories of the training data in an unsupervised manner. We use the model as a feature extractor.

(2) **supervised classifier training**: we learn a classifier from labeled data based on the extracted features for classification.

Table: 3D object classification on ModelNet10 dataset

Method	Accuracy
Geometry Image	88.4%
PANORAMA-NN	91.1%
ECC	90.0%
3D ShapeNets	83.5%
DeepPano	85.5%
SPH	79.8%
VConv-DAE	80.5%
3D-GAN	91.0%
3D DescriptorNet (ours)	**92.4%**

Conclusion

Conclusion

+ We propose the 3D DescriptorNet for volumetric objects.
+ We propose the conditional 3D DescriptorNet for 3D object recovery and 3D object super resolution.
+ The proposed model can be used to train a 3D generator via cooperative training.
+ The model can be useful for semi-supervised learning in 3D object classification.

[1] Jianwen Xie*, [2] Zilong Zheng*, [2] Ruiqi Gao, [3] Wenguan Wang, [2] Song-Chun Zhu, [2] Ying Nian Wu

Thank you!