Generative Image Modeling Using Style and Structure Adversarial Networks

Xiaolong Wang[(✉)] and Abhinav Gupta

Robotics Institute, Carnegie Mellon University, Pittsburgh, USA
`xiaolonw@cs.cmu.edu`

Abstract. Current generative frameworks use end-to-end learning and generate images by sampling from uniform noise distribution. However, these approaches ignore the most basic principle of image formation: images are product of: (a) Structure: the underlying 3D model; (b) Style: the texture mapped onto structure. In this paper, we factorize the image generation process and propose Style and Structure Generative Adversarial Network (S^2-GAN). Our S^2-GAN has two components: the Structure-GAN generates a surface normal map; the Style-GAN takes the surface normal map as input and generates the 2D image. Apart from a real vs. generated loss function, we use an additional loss with computed surface normals from generated images. The two GANs are first trained independently, and then merged together via joint learning. We show our S^2-GAN model is interpretable, generates more realistic images and can be used to learn unsupervised RGBD representations.

1 Introduction

Unsupervised learning of visual representations is one of the most fundamental problems in computer vision. There are two common approaches for unsupervised learning: (a) using a discriminative framework with auxiliary tasks where supervision comes for free, such as context prediction [1,2] or temporal embedding [3–8]; (b) using a generative framework where the underlying model is compositional and attempts to generate realistic images [9–12]. The underlying hypothesis of the generative framework is that if the model is good enough to generate novel and realistic images, it should be a good representation for vision tasks as well. Most of these generative frameworks use end-to-end learning to generate RGB images from control parameters (z also called noise since it is sampled from a uniform distribution). Recently, some impressive results [13] have been shown on restrictive domains such as faces and bedrooms.

However, these approaches ignore one of the most basic underlying principles of image formation. Images are a product of two separate phenomena: **Structure:** this encodes the underlying geometry of the scene. It refers to the underlying mesh, voxel representation etc. **Style:** this encodes the texture on the objects and the illumination. In this paper, we build upon this IM101 principle of image formation and factor the generative adversarial network (GAN) into two generative processes as Fig. 1. The first, a structure generative model (namely
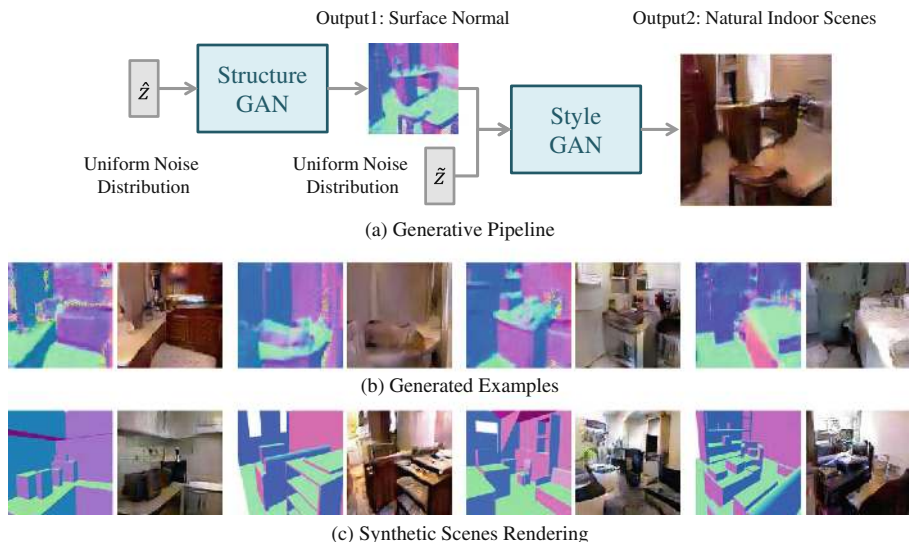
(a) Generative Pipeline

(b) Generated Examples

(c) Synthetic Scenes Rendering

Fig. 1. (a) Generative Pipeline: Given $\hat{z}$ sampled from uniform distribution, our Structure-GAN generates a surface normal map as output. This surface normal map is then given as input with $\tilde{z}$ to a second generator network (Style-GAN) and outputs an image. (b) We show examples of generated surface normal maps and images. (c) Our Style-GAN can be used as a rendering engine: given a synthetic scene, we can use it to render a realistic image. To visualize the normals, we represent facing right with blue, horizontal surface with green, facing left with red (blue $\rightarrow$ X; green $\rightarrow$ Y; red $\rightarrow$ Z). (Color figure online)

Structure-GAN), takes $\hat{z}$ and generates the underlying 3D structure (y_{3D}) for the scene. The second, a conditional generative network (namely Style-GAN), takes y_{3D} as input and noise $\tilde{z}$ to generate the image y_I. We call this factored generative network Style and Structure Generative Adversarial Network (S^2-GAN).

Why S^2-GAN? We believe there are fourfold advantages of factoring the style and structure in the image generation process. Firstly, factoring style and structure simplifies the overall generative process and leads to more realistic high-resolution images. It also leads to a highly stable and robust learning procedure. Secondly, due to the factoring process, S^2-GAN is more interpretable as compared to its counterparts. One can even factor the errors and understand where the surface normal generation failed as compared to texture generation. Thirdly, as our results indicate, S^2-GAN allows us to learn RGBD representation in an unsupervised manner. This can be crucial for many robotics and graphics applications. Finally, our Style-GAN can also be thought of as a learned rendering engine which, given any 3D input, allows us to render a corresponding image. It also allows us to build applications where one can modify the underlying 3D structure of an input image and render a completely new image.

However, learning S^2-GAN is still not an easy task. To tackle this challenge, we first learn the Style-GAN and Structure-GAN in an independent manner. We use the NYUv2 RGBD dataset [14] with more than 200 K frames for learning the initial networks. We train a Structure-GAN using the ground truth surface normals from Kinect. Because the perspective distortion of texture is more directly related to normals than to depth, we use surface normal to represent image structure in this paper. We learn in parallel our Style-GAN which is conditional on the ground truth surface normals. While training the Style-GAN, we have two loss functions: the first loss function takes in an image and the surface normals and tries to predict if they correspond to a real scene or not. However, this loss function alone does not enforce explicit pixel based constraints for aligning generated images with input surface normals. To enforce the pixel-wise constraints, we make the following assumption: if the generated image is realistic enough, we should be able to reconstruct or predict the 3D structure based on it. We achieve this by adding another discriminator network. More specifically, the generated image is not only forwarded to the discriminator network in GAN but also a input for the trained surface normal predictor network. Once we have trained an initial Style-GAN and Structure-GAN, we combine them together and perform end-to-end learning jointly where images are generated from $\hat{z}, \tilde{z}$ and fed to discriminators for real/fake task.

2   Related Work

Unsupervised learning of visual representation is one of the most challenging problems in computer vision. There are two primary approaches to unsupervised learning. The first is the discriminative approach where we use auxiliary tasks such that ground truth can be generated without labeling. Some examples of these auxiliary tasks include predicting: the relative location of two patches [2], ego-motion in videos [15,16], physical signals [17–19].

A more common approach to unsupervised learning is to use a generative framework. Two types of generative frameworks have been used in the past. Non-parametric approaches perform matching of an image or patch with the database for tasks such as texture synthesis [20] or super-resolution [21]. In this paper, we are interested in developing a parametric model of images. One common approach is to learn a low-dimensional representation which can be used to reconstruct an image. Some examples include the deep auto-encoder [22,23] or Restricted Boltzmann machines (RBMs) [24–28]. However, in most of the above scenarios it is hard to generate new images since sampling in latent space is not an easy task. The recently proposed Variational auto-encoders (VAE) [10,11] tackles this problem by generating images with variational sampling approach. However, these approaches are restricted to simple datasets such as MNIST. To generate interpretable images with richer information, the VAE is extended to be conditioned on captions [29] and graphics code [30]. Besides RBMs and auto-encoders, there are also many novel generative models in recent literature [31–34]. For example, Dosovitskiy et al. [31] proposed to use CNNs to generate chairs.

In this work, we build our model based on the Generative Adversarial Networks (GANs) framework proposed by Goodfellow et al. [9]. This framework was extended by Denton et al. [35] to generate images. Specifically, they proposed to use a Laplacian pyramid of adversarial networks to generate images in a coarse to fine scheme. However, training these networks is still tricky and unstable. Therefore, an extension DCGAN [13] proposed good practices for training adversarial networks and demonstrated promising results in generating images. There are more extensions include using conditional variables [36–38]. For instance, Mathieu et al. [37] introduced to predict future video frames conditioned on the previous frames. In this paper, we further simplify the image generation process by factoring out the generation of 3D structure and style.

In order to train our S^2-GAN we combine adversarial loss with 3D surface normal prediction loss [39–42] to provide extra constraints during learning. This is also related to the idea of combining multiple losses for better generative modeling [43–45]. For example, Makhzani et al. [43] proposed an adversarial auto-encoder which takes the adversarial loss as an extra constraint for the latent code during training the auto-encoder. Finally, the idea of factorizing image into two separate phenomena has been well studied in [46–49], which motivates us to decompose the generative process to structure and style. We use the RGBD data from NYUv2 to factorize and learn a S^2-GAN model.

3   Background for Generative Adversarial Networks

The Generative Adversarial Networks (GAN) [9] contains two models: generator G and discriminator D. The generator G takes the input which is a latent random vector z sampled from uniform noise distribution and tries to generate a realistic image. The discriminator D performs binary classification to distinguish whether an image is generated from G or it is a real image. Thus the two models are competing against each other (hence, adversarial): network G will try to generate images which will be hard for D to differentiate from real image, meanwhile network D will learn to avoid getting fooled by G.

Formally, we optimize the networks using gradient descent with batch size M. We are given samples as $\mathbf{X} = (X_1, ..., X_M)$ and a set of z sampled from uniform distribution as $\mathbf{Z} = (z_1, ..., z_M)$. The training of GAN is an iterative procedure with 2 steps: (i) fix the parameters of network G and optimize network D; (ii) fix network D and optimize network G. The loss for training network D is,

$$L^D(\mathbf{X}, \mathbf{Z}) = \sum_{i=1}^{M/2} L(D(X_i), 1) + \sum_{i=M/2+1}^{M} L(D(G(z_i)), 0). \qquad (1)$$

Inside a batch, half of images are real and the rest $G(z_i)$ are images generated by G given z_i. $D(X_i) \in [0, 1]$ represents the binary classification score given input image X_i. $L(y^*, y) = -[y \log(y^*) + (1-y)log(1-y^*)]$ is the binary entropy loss. Thus the loss Eq. 1 for network D is optimized to classify the real image as label

Table 1. Network architectures. Top: generator of Structure-GAN; bottom: discriminator of Structure-GAN (left) and discriminator of Style-GAN (right). "conv" means convolutional layer, "uconv" means fractionally-strided convolutional (deconvolutional) layer, where $2(up)$ stride indicates 2x resolution. "fc" means fully connected layer.

Structure-GAN(G)	fc	uconv	conv	conv	conv	conv	uconv	conv	uconv	conv
Input Size	−	9	18	18	18	18	18	36	36	72
Kernel Number	$9 \times 9 \times 64$	128	128	256	512	512	256	128	64	3
Kernel Size	−	4	3	3	3	3	4	3	4	5
Stride	−	$2(up)$	1	1	1	1	$2(up)$	1	$2(up)$	1

Structure-GAN(D)	conv	conv	conv	conv	conv	fc
Input Size	72	36	36	18	9	−
Kernel Number	64	128	256	512	128	1
Kernel Size	5	5	3	3	3	−
Stride	2	1	2	2	1	−

Style-GAN(D)	conv	conv	conv	conv	conv	fc
Input Size	128	64	32	16	8	−
Kernel Number	64	128	256	512	128	1
Kernel Size	5	5	3	3	3	−
Stride	2	2	2	2	1	−

1 and the generated image as 0. On the other hand, the generator G is trying to fool D to classify the generated image as a real image via minimizing the loss:

$$L^G(\mathbf{Z}) = \sum_{i=M/2+1}^{M} L(D(G(z_i)), 1). \qquad (2)$$

4   Style and Structure GAN

GAN and DCGAN approaches directly generate images from the sampled z. Instead, we use the fact that image generation has two components: (a) generating the underlying structure based on the objects in the scene; (b) generating the texture/style on top of this 3D structure. We use this simple observation to decompose the generative process into two procedures: (i) Structure-GAN - this process generates surface normals from sampled $\hat{z}$ and (ii) Style-GAN - this model generates the images taking as input the surface normals and another latent variable $\tilde{z}$ sampled from uniform distribution. We train both models with RGBD data, and the ground truth surface normals are obtained from the depth.

4.1   Structure-GAN

We can directly apply GAN framework to learn how to generate surface normal maps. The input to the network G will be $\hat{z}$ sampled from uniform distribution and the output is a surface normal map. We use a 100-d vector to represent the $\hat{z}$ and the output is in size of $72 \times 72 \times 3$ (Fig. 2). The discriminator D will learn to classify the generated surface normal maps from the real maps obtained from depth. We introduce our network architecture as following.

Generator Network. As Table 1 (top row) illustrates, we apply a 10-layer model for the generator. Given a 100-d $\hat{z}$ as input, it is first fully connected to a 3D block ($9 \times 9 \times 64$). Then we further perform convolutional operations on top of

Fig. 2. Left: 4 Generated Surface Normal maps. Right: 2 Pairs of rendering results on ground truth surface normal maps using the Style-GAN without pixel-wise constraints.

it and generate the surface normal map in the end. Note that "uconv" represents fractionally-strided convolution [13], which is also called as deconvolution. We follow the settings in [13] and use Batch Normalization [50] and ReLU activations after each layer except for the last layer, where a TanH activation is applied.

Discriminator Network. We show the 6-layer network architecture in Table 1 (bottom left). Taking an image as input, the network outputs a single number which predicts the input surface normal is real or generated. We use LeakyReLU [51,52] for activation functions as in [13]. However, we do not apply Batch Normalization here. In our case, we find that the discriminator network easily finds trivial solutions with Batch Normalization.

4.2 Style-GAN

Given the RGB images and surface normal maps from Kinect, we train another GAN in parallel to generate images conditioned on surface normals. We call this network Style-GAN. First, we modify our generator network to a conditional GAN as proposed in [35,36]. The conditional information, i.e., surface normal maps, are given as additional inputs for both the generator G and the discriminator D. Augmenting surface normals as an additional input to D not only forces the generated image to look real, but also implicitly enforces the generated image to match the surface normal map. While training this discriminator, we only consider real RGB images and their corresponding surface normals as the positive examples. Given more cues from surface normals, we generate higher resolution of $128 \times 128 \times 3$ images with the Style-GAN.

Formally, we have a batch of RGB images $\mathbf{X} = (X_1, ..., X_M)$ and their corresponding surface normal maps $\mathbf{C} = (C_1, ..., C_M)$, as well as samples from noise distribution $\tilde{\mathbf{Z}} = (\tilde{z}_1, ..., \tilde{z}_M)$. We reformulate the generative function from $G(\tilde{z}_i)$ to $G(C_i, \tilde{z}_i)$ and discriminative function is changed from $D(X_i)$ to $D(C_i, X_i)$. Then the loss of discriminator network in Eq. 1 can be reformulated as,

$$L_{cond}^{D}(\mathbf{X}, \mathbf{C}, \tilde{\mathbf{Z}}) = \sum_{i=1}^{M/2} L(D(C_i, X_i), 1) + \sum_{i=M/2+1}^{M} L(D(C_i, G(C_i, \tilde{z}_i)), 0), \quad (3)$$

and the loss of generator network in Eq. 2 can be reformulated as,

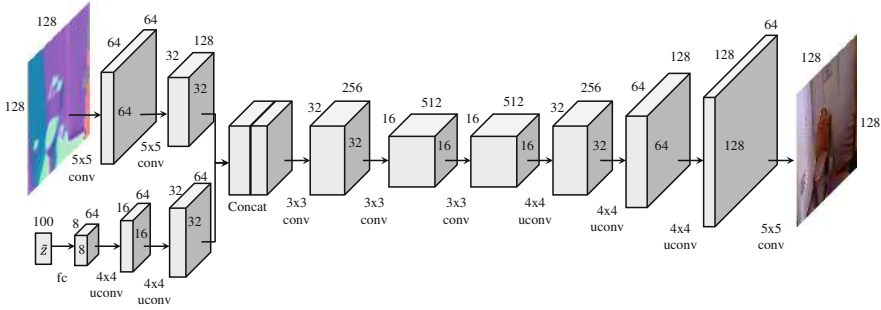$$L_{cond}^{G}(\mathbf{C}, \tilde{\mathbf{Z}}) = \sum_{i=M/2+1}^{M} L(D(C_i, G(C_i, \tilde{z}_i)), 1). \quad (4)$$

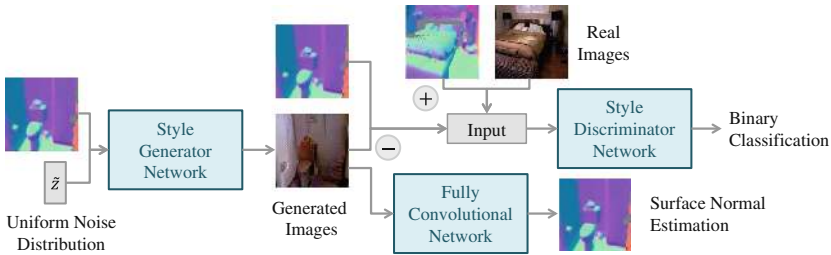**Fig. 3.** The architecture of the generator in Style-GAN.

Fig. 4. Our Style-GAN. Given the ground truth surface normals and $\tilde{z}$ as inputs, the generator G learns to generate RGB images. The supervision comes from two networks: The discriminator network takes the generated images, real images and their corresponding normal maps as inputs to perform classification; The FCN takes the generated images as inputs and predict the surface normal maps.

We apply the same scheme of iterative training. By doing this, we can generate the images with network G as visualized in Fig. 2 (right).

Network Architecture. We show our generator as Fig. 3. Given a $128 \times 128 \times 3$ surface normal map and a 100-d $\tilde{z}$ as input, they are firstly forwarded to convolutional and deconvolutional layers respectively and then concatenated to form $32 \times 32 \times 192$ feature maps. On top of these feature maps, 7 layers of convolutions and deconvolutions are further performed. The output of the network is a $128 \times 128 \times 3$ RGB image. For the discriminator, we apply the similar architecture of the one in Structure-GAN (bottom right in Table 1). The input for the network is the concatenation of surface normals and images ($128 \times 128 \times 6$).

4.3   Multi-task Learning with Pixel-Wise Constraints

The Style-GAN can make the generated image look real and also enforce it to match the provided surface normal maps implicitly. However, as shown Fig. 2, the images are noisy and the edges are not well aligned with the edges in the surface normal maps. Thus, we propose to add a pixel-wise constraint to explicitly guide the generator to align the outputs with the input surface normal maps.
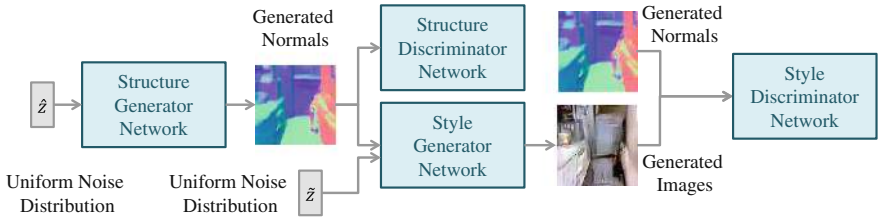
Fig. 5. Full model of our S^2-GAN. It can directly generate RGB images given $\hat{z}, \tilde{z}$ as inputs. For simplicity, we do not visualize the positive samples in training. During joint learning, the loss from Style-GAN is also passed down to the Structure-GAN.

We make the following assumption: If the generated image is real enough, it can be used for reconstructing the surface normal maps. To encode this constraint, we train another network for surface normal estimation. We modify the Fully Convolutional Network (FCN) [53] with the classification loss as mentioned in [39] for this task. More specifically, we quantize the surface normals to 40 classes with k-means clustering as in [39,54] and the loss is defined as

$$L^{FCN}(\mathbf{X}, \mathbf{C}) = \frac{1}{K \times K} \sum_{i=1}^{M} \sum_{k=1}^{K \times K} L_s(F_k(X_i), C_{i,k}), \tag{5}$$

where L_s means the softmax loss and the output surface normal map is in $K \times K$ dimension, and $K = 128$ is in the same size of input image. $F_k(X_i)$ is the output of kth pixel in the ith sample. $C_{i,k}(1 \leqslant C_{i,k} \leqslant 40)$ is the label for the kth pixel in sample i. Thus the loss is designed to enforce each pixel in the image to generate accurate surface normal. Note that when training the FCN, we use the RGBD data which provides indoor scene images and ground truth surface normals. The model is trained from scratch without ImageNet pre-training.

FCN Architecture. We apply the AlexNet [55] following the same training scheme as [53], with modifications on the last 3 layers. Given a generated 128×128 image, it is first upsampled to 512×512 before feeding into the FCN. For the two layers before the last layer, we use smaller kernel numbers of 1024 and 512. The last layer is a deconvolutional layer with stride 2. In the end, upsampling (4x resolution) is further applied to generate the high quality results.

Given the trained FCN model, we can use it as an additional supervision (constraint) in the adversarial learning. Our final model is illustrated in Fig. 4. During training, not only the gradients from the classification loss of D will be passed down to G, but also the surface normal estimation loss from the FCN is passed through the generated image to G. This way, the adversarial loss from D will make the generated images look real, and the FCN will give pixel-wise constraints to make the generated images aligned with surface normal maps.

Formally, we combine the two losses in Eqs. 4 and 5 for the generator G,

$$L_{multi}^G(\mathbf{C}, \tilde{\mathbf{Z}}) = L_{cond}^G(\mathbf{C}, \tilde{\mathbf{Z}}) + L^{FCN}(G(\mathbf{C}, \tilde{\mathbf{Z}}), \mathbf{C}), \tag{6}$$

where $G(\mathbf{C}, \tilde{\mathbf{Z}})$ represents the generated images given a batch of surface normal maps $\mathbf{C}$ and noise $\tilde{\mathbf{Z}}$. The training procedure for this model is similar to the original adversarial learning, which includes three steps in each iteration:

– Fix the generator G, optimize the discriminator D with Eq. 3.
– Fix the FCN and the discriminator D, optimize the generator G with Eq. 6.
– Fix the generator G, fine-tune FCN using generated and real images.

Note that the parameters of FCN model are fixed in the beginning of multi-task learning, i.e., we do not fine-tune FCN in the beginning. The reason is the generated images are not good in the beginning, so feeding bad examples to FCN seems to make the surface normal prediction worse.

4.4   Joint Learning for S²-GAN

After training the Structure-GAN and Style-GAN independently, we merge all networks and train them jointly. As Fig. 5 shows, our full model includes surface normal generation from Structure-GAN, and based on it the Style-GAN generates the image. Note that the generated normal maps are first passed through an upsampling layer with bilinear interpolation before they are forwarded to the Style-GAN. Since we do not use ground truth surface normal maps to generate the images, we remove the FCN constraint from the Style-GAN. The discriminator in Style-GAN takes generated normals and images as negative samples, and ground truth normals and real images as positive samples.

For the Structure-GAN, the generator network receives not only the gradients from the discriminator of Structure-GAN, but also the gradients passed through the generator of Style-GAN. In this way, the network is forced to generate surface normals which not only are realistic but also help generate better RGB images. Formally, the loss for the generator network of Structure-GAN can be represented as combining Eqs. 2 and 4,

$$L_{joint}^{G}(\hat{\mathbf{Z}}, \tilde{\mathbf{Z}}) = L^{G}(\hat{\mathbf{Z}}) + \lambda \cdot L_{cond}^{G}(G(\hat{\mathbf{Z}}), \tilde{\mathbf{Z}}) \qquad (7)$$

where $\hat{\mathbf{Z}} = (\hat{z}_1, ..., \hat{z}_M)$ and $\tilde{\mathbf{Z}} = (\tilde{z}_1, ..., \tilde{z}_M)$ represent two sets of samples drawn from uniform distribution for Structure-GAN and Style-GAN respectively. The first term in Eq. 7 represents the adversarial loss from the discriminator of Structure-GAN and the second term represents that the loss of the Style-GAN is also passed down. We set the coefficient $\lambda = 0.1$ and smaller learning rate for Structure-GAN than Style-GAN in the experiments, so that we can prevent the generated normals from over fitting to the task of generating RGB images via Style-GAN. In our experiments, we find that without constraining λ and learning rates, the loss $L^{G}(\hat{\mathbf{Z}})$ easily diverges to high values and the Structure-GAN can no longer generate reasonable surface normal maps.

5   Experiments

We perform two types of experiments: (a) We qualitatively and quantitatively evaluate the quality of images generates using our model; (b) We evaluate the

quality of unsupervised representation learning by applying the network for different tasks such as image classification and object detection.

Dataset. We use the NYUv2 dataset [14] in our experiment. We use the raw video data during training and extract $200\,K$ frames from the 249 training video scenes. We compute the surface normals from the depth as [39,42].

Parameter Settings. We follow the parameters in [13] for training. We trained the models using Adam optimizer [56] with momentum term $\beta_1 = 0.5, \beta_2 = 0.999$ and batch size $M = 128$. The inputs and outputs for all networks are scaled to $[-1, 1]$ (including surface normals and RGB images). During training the Style and Structure GANs separately, we set the learning rate to 0.0002. We train the Structure-GAN for 25 epochs. For Style-GAN, we first fix the FCN model and train it for 25 epochs, then the FCN model are fine-tuned together with 5 more epochs. For joint learning, we set learning rate as 10^{-6} for Style-GAN and 10^{-7} for Structure-GAN and train them for 5 epochs.

Baselines. We have 4 baseline models trained on NYUv2 training set: (a) DCGAN [13]: it takes uniform noise as input and generate 64×64 images; (b) DCGAN + LAPGAN: we train a LAPGAN [35] on top of DCGAN, which takes lower resolution images as inputs and generates 128×128 images. We apply the same architecture as our Style-GAN for LAPGAN (Fig. 3 and Table 1). (c) DCGANv2: we train a DCGAN with the same architecture as our Structure-GAN (Table 1). (d) DCGANv2+LAPGAN: we train another LAPGAN on top of DCGANv2 as (b) with the same architecture. Note that baseline (d) has the same model complexity as our model.

5.1   Qualitative Results for Image Generation

Style-GAN Visualization. Before showing the image generation results of the full S^2-GAN model, we first visualize the results of our Style-GAN given the ground truth surface normals on the NYUv2 test set. As illustrated in the first 3 rows of Fig. 6, we can generate nice rendering results which are well aligned with the surface normal inputs. By comparing with the original RGB images, we show that our method can generate a different style (illumination, color, texture) of image with the same structure. We also make comparisons on the results of Style-GAN with/without pixel-wise constraints as visualized in Fig. 7. We show that if we train the model without the pixel-wise constraint, the output is less smooth and noisier than our approach.

Rendering on Synthetic Scenes. One application of our Style-GAN is rendering synthetic scenes. We use the 3D models annotated in [57] to generate the synthetic scenes. We use the scenes corresponding to the NYUv2 test set and make some modifications by rotation, zooming in/out. As the last two rows of Fig. 6 show, we can obtain very realistic rendering results on 3D models.

S^2-GAN Visualization. We now show the results of our full generative model. Given the noise $\hat{z}, \tilde{z}$, our model generate both surface normal maps (72×72) and

Input   Output   Original   Input   Output   Original   Input   Output   Original

Input   Output   Input   Output   Input   Output   Input   Output
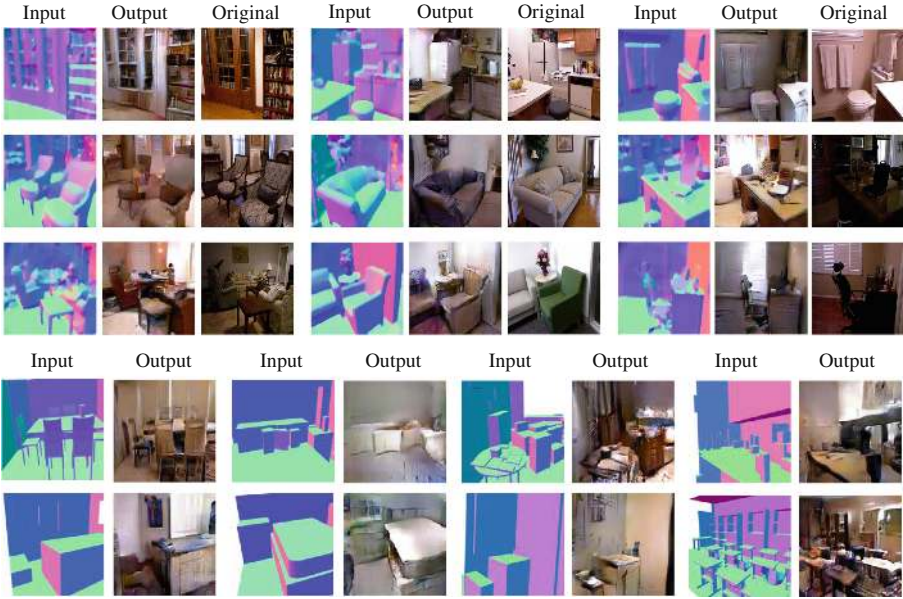

Fig. 6. Results of Style-GAN conditioned on ground truth surface normals (first 3 rows) and synthetic scenes (last 2 rows). For ground truth normals, we show the input normals, our generated images and the original corresponding images.
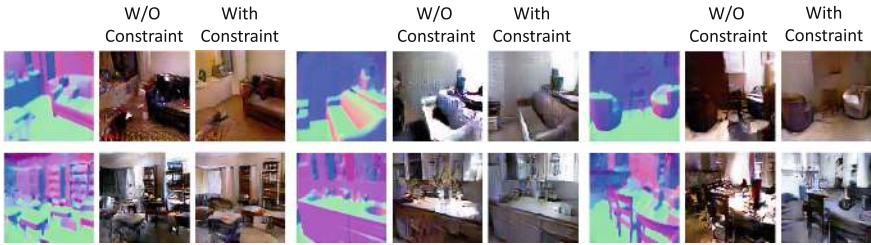
W/O   With   W/O   With   W/O   With
Constraint Constraint   Constraint Constraint   Constraint Constraint

Fig. 7. Comparison between models with and without pixel-wise constraints.

RGB images (128×128) after that, as shown in Fig. 8(a). We compare with the baselines including DCGAN (Fig. 8(b)) and DCGAN + LAPGAN (Fig. 8(c)). We can see that our method can generate more structured indoor scenes, i.e., it is easier to figure out the structure and objects in our image. We also find that using LAPGAN does not help much improving the qualitative results.

Walking the Latent Space. One big advantage of our model is that it is interpretable. Recall that we have two random uniform vectors $\hat{z}, \tilde{z}$ as inputs for Structure and Style networks. We conduct two experiments here: (i) Fix $\tilde{z}$ (style) and manipulate the structure of images by changing $\hat{z}$; (ii) Fix $\hat{z}$ (structure) and manipulate the style of images by changing $\tilde{z}$. Specifically, given an initial set of

(a) Indoor scenes generated by our method

(b) Indoor scenes generated by DCGAN
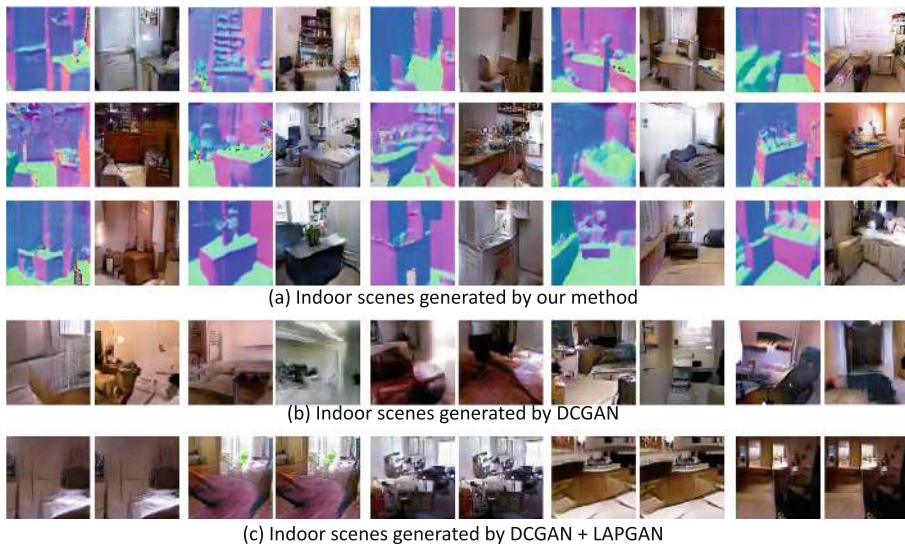

(c) Indoor scenes generated by DCGAN + LAPGAN

Fig. 8. (a) Pairs of surface normals and images generated by S^2-GAN. (b) Results of DCGAN. (c) Results of DCGAN + LAPGAN. For each pair, result on the left is from DCGAN and on the right is applying LAPGAN after it.

$\hat{z}$ and $\tilde{z}$, we pick up a series of 10 random points in $\hat{z}$ or $\tilde{z}$ and gradually add 0.1 to these points for 6–7 times. We show that we can obtain smooth transitions in the outputs by interpolating the inputs as Fig. 9. For the example in the first two rows of Fig. 9, we show that by interpolating $\hat{z}$, we can gradually "grow" a 3D cube in the room and the style of the RGB images are consistent since we fix the $\tilde{z}$. For the last rows in Fig. 9, we fix the structure of the image and interpolate the $\tilde{z}$ so that the window of the room is gradually shut down.

User Study. We collect 1000 pairs of images randomly generated by our method and DCGAN. We let the AMT workers to judge which one is more realistic in each pair and 71 % of the time they think our approach generates better images.

5.2 Quantitative Results for Image Generation

To evaluate the generated images quantitatively, we apply the AlexNet pre-trained (supervised) on Places [58] and ImageNet dataset [59] to perform classification and detection on them. The motivation is: If the generated images are realistic enough, state of the art classifiers and detectors should fire on them with high scores. We compare our method with the three baselines mentioned in the beginning of experiment: DCGAN, DCGANv2 and DCGANv2 + LAPGAN. We generate 10 K images for each model and perform evaluation on them.

Classification on Generated Images. We apply the Places-AlexNet [58] to perform classification on the generated images. If the image is real enough, the
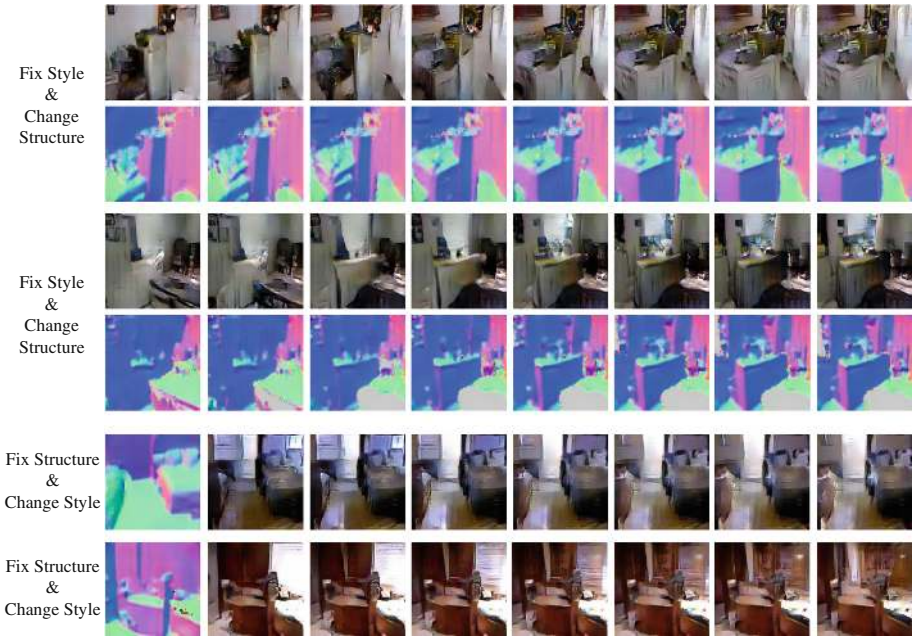
Fig. 9. Walking the latent space: Our latent space is more interpretable and we obtain smooth transitions of generated results by interpolating the inputs.

Places-AlexNet will give high response in one class during classification. Thus, we can use the maximum norm $||\cdot||_\infty$ of the softmax output (i.e., the maximum probability) of Places-AlexNet to represent the image quality. We compute the results for this metric on all generated images and show the mean for different models as Fig. 10(a). S^2-GAN is around 2 % better than the baselines.

Object Detection on Generated Images. We used Fast-RCNN detector [60] fine-tuned on the NYUv2 dataset with ImageNet pre-trained AlexNet. We then apply the detector on generated images. If the image is realistic enough, the detector should find objects (door, bed, sofa, table, counter etc.). Thus, we want to investigate on which images the detector can find more foreground objects. We plot the curves shown in Fig. 10(b) (the x-axis represents the detection threshold, and the y-axis represents average number of detections). We show that the detector can find more foreground objects in the images generated by S^2-GAN. At 0.3 threshold, there are on average 2.2 detections per image and 1.72 detections on images generated by DCGAN.

### 5.3   Representation Learning for Recognition Tasks

We now explore whether the representation learned by the discriminator network in our Style-GAN can be transferred to tasks such as scene classification and object detection. Since the input for the network is RGB image and surface

	S²-GAN	DCGAN	DCGANv2	DCGANv2 + LAPGAN
Maximum Norm	**29.0**	25.6	27.1	27.2

(a) Classification on generated images.

	S²-GAN	DCGAN	GIST	Places-AlexNet
RGB Accuracy	-	21.3	19.7	38.1
D Accuracy	-	19.1	20.1	27.7
RGBD Accuracy	35.3	27.1	23.0	39.0

(c) Classification on SUN RGB-D dataset.
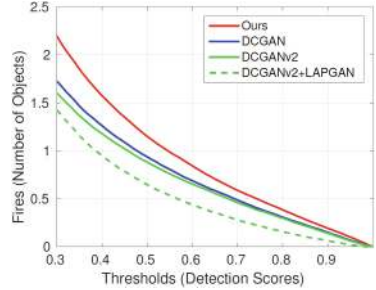
(b) Object detection on generated images.

Fig. 10. (a) The maximum norm of classification results on generated images. (b) Number of fires over different thresholds for object detection on generated images. (c) Scene classification on SUN RGB-D with our model and other methods (no fine-tuning).

normal map, our model can be applied to recognition tasks in RGBD data. We perform the experiments on scene classification on SUN RGB-D dataset [14,61–63] as well as object detection on NYUv2 dataset.

Scene Classification. We use the standard train/test split for scene classification in SUN RGB-D dataset, which includes 19 classes with 4852 training and 4660 testing images. We use our model, taking RGB images and normals as inputs, to extract the feature of the second-to-last layer and train SVM on top of it. We compare our method with the discriminator network in DCGAN and the baselines reported in [61]: GIST [64] feature as well as Places-AlexNet [58]. For the networks trained with only RGB data, we follow [61,65], which directly use them to extract feature on the depth representation. Then the features extracted from both RGB and depth are concatenated together as inputs for SVM classifier. Note that all models are not fine-tuned on the dataset. As Fig. 10(c) shows, our model is 8.2 % better than DCGAN and 3.7 % away from the Places-AlexNet.

Object Detection. In this task, we perform RGBD object detection on the NYUv2 dataset. We follow the Fast-RCNN pipeline [60] and use the code and parameter settings provided in [66]. In our case, we use surface normal to represent the depth. To apply our model for the detection task, we stacked two fully connected layer (4096-d) on top of the last convolutional layer and fine-tune the network end-to-end. We compare against four baselines: network with the same architecture trained from scratch, network pre-trained with DCGAN, DCGANv2, and ImageNet pre-trained AlexNet. For networks pre-trained on only RGB data, we fine-tune them on both the RGB and surface normal inputs separately and average the detection results during testing as [66]. We apply Batch Normalization [50] except for ImageNet pre-trained AlexNet. We show the results in Table 2. Our approach has 1.5 % improvement compared to the model trained from scratch.

Table 2. Detection results on NYU test set.

	mean	bath tub	bed	book shelf	box	chair	counter	desk	door	dresser	garbage bin	lamp	monitor	night stand	pillow	sink	sofa	table	tele vision	toilet
Ours	32.4	**44.0**	67.7	28.4	1.6	34.2	43.9	10.0	17.3	33.9	22.6	28.1	24.8	41.7	31.3	33.1	50.2	21.9	25.1	54.9
Scratch	30.9	35.6	67.7	23.1	2.1	33.1	40.5	10.1	15.2	31.2	19.4	26.8	29.1	39.9	30.5	36.6	43.8	20.4	29.5	52.8
DCGAN	30.4	38.9	67.6	26.3	**2.9**	32.5	39.1	10.6	16.9	23.6	23.0	26.5	25.1	44.5	29.6	**37.0**	45.2	21.0	28.5	38.4
DCGANv2	31.1	35.3	69.0	21.5	2.0	32.6	36.4	9.8	14.4	30.8	25.4	29.2	27.3	39.6	32.2	34.6	47.9	21.1	27.2	54.4
Imagenet	**37.6**	33.1	**69.9**	**39.6**	2.3	**38.1**	**47.9**	**16.1**	**24.6**	**40.7**	**26.5**	**37.8**	**45.6**	**49.5**	**36.1**	34.5	**53.2**	**25.0**	**35.3**	**58.4**

6   Conclusion

We present a novel Style and Structure GAN which factorizes the image generation process. We show our model is more interpretable and generates more realistic images compared to the baselines. We also show that our method can learn RGBD representations in an unsupervised manner.

References

1. Doersch, C., Gupta, A., Efros, A.A.: Context as supervisory signal: discovering objects with predictable context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8691, pp. 362–377. Springer, Heidelberg (2014). doi:10.1007/978-3-319-10578-9_24

2. Doersch, C., Gupta, A., Efros, A.A.: Unsupervised visual representation learning by context prediction. In: ICCV (2015)

3. Wang, X., Gupta, A.: Unsupervised learning of visual representations using videos. In: ICCV (2015)

4. Goroshin, R., Bruna, J., Tompson, J., Eigen, D., LeCun, Y.: Unsupervised learning of spatiotemporally coherent metrics. In: ICCV (2015)

5. Zou, W.Y., Zhu, S., Ng, A.Y., Yu, K.: Deep learning of invariant features via simulated fixations in video. In: NIPS (2012)

6. Li, Y., Paluri, M., Rehg, J.M., Dollar, P.: Unsupervised learning of edges. In: CVPR (2016)

7. Walker, J., Gupta, A., Hebert, M.: Dense optical flow prediction from a static image. In: ICCV (2015)

8. Misra, I., Zitnick, C.L., Hebert, M.: Shuffle and learn: unsupervised learning using temporal order verification. In: ECCV (2016)

9. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: NIPS (2014)

10. Kingma, D., Welling, M.: Auto-encoding variational bayes. In: ICLR (2014)

11. Gregor, K., Danihelka, I., Graves, A., Rezende, D.J., Wierstra, D.: Draw: a recurrent neural network for image generation. CoRR abs/1502.04623 (2015)

12. Li, Y., Swersky, K., Zemel, R.: Generative moment matching networks. In: ICML (2014)

13. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. CoRR abs/1511.06434 (2015)

14. Silberman, N., Hoiem, D., Kohli, P., Fergus, R.: Indoor segmentation and support inference from RGBD images. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7576, pp. 746–760. Springer, Heidelberg (2012). doi:10.1007/978-3-642-33715-4_54
15. Agrawal, P., Carreira, J., Malik, J.: Learning to see by moving. In: ICCV (2015)
16. Jayaraman, D., Grauman, K.: Learning image representations tied to ego-motion. In: ICCV (2015)
17. Owens, A., Isola, P., McDermott, J., Torralba, A., Adelson, E., Freeman, W.: Visually indicated sounds. In: CVPR (2016)
18. Pinto, L., Gupta, A.: Supersizing self-supervision: learning to grasp from 50 k tries and 700 robot hours. In: ICRA (2016)
19. Pinto, L., Gandhi, D., Han, Y., Park, Y.L., Gupta, A.: The curious robot: learning visual representations via physical interactions. In: ECCV (2016)
20. Efros, A.A., Leung, T.K.: Texture synthesis by non-parametric sampling. In: ICCV (1999)
21. Freeman, W.T., Jones, T.R., Pasztor, E.C.: Example-based super-resolution. In: Computer Graphics and Applications (2002)
22. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H.: Greedy layer-wise training of deep networks. In: NIPS (2007)
23. Le, Q.V., Ranzato, M.A., Monga, R., Devin, M., Chen, K., Corrado, G.S., Dean, J., Ng, A.Y.: Building high-level features using large scale unsupervised learning. In: ICML (2012)
24. Ranzato, M.A., Krizhevsky, A., Hinton, G.E.: Factored 3-way restricted Boltzmann machines for modeling natural images. In: AISTATS (2010)
25. Osindero, S., Hinton, G.E.: Modeling image patches with a directed hierarchy of Markov random fields. In: NIPS (2008)
26. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**, 504–507 (2006)
27. Lee, H., Grosse, R., Ranganath, R., Ng, A.Y.: Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: ICML (2009)
28. Taylor, G.W., Hinton, G.E., Roweis, S.: Modeling human motion using binary latent variables. In: NIPS (2006)
29. Mansimov, E., Parisotto, E., Ba, J.L., Salakhutdinov, R.: Generating images from captions with attention. CoRR abs/1511.02793 (2015)
30. Kulkarni, T.D., Whitney, W.F., Kohli, P., Tenenbaum, J.B.: Deep convolutional inverse graphics network. In: NIPS (2015)
31. Dosovitskiy, A., Springenberg, J.T., Brox, T.: Learning to generate chairs with convolutional neural networks. In: CVPR (2015)
32. Tatarchenko, M., Dosovitskiy, A., Brox, T.: Single-view to multi-view: reconstructing unseen views with a convolutional network. CoRR abs/1511.06702 (2015)
33. Theis, L., Bethge, M.: Generative image modeling using spatial LSTMs. CoRR abs/1506.03478 (2015)
34. Oord, A.V.D., Kalchbrenner, N., Kavukcuoglu, K.: Pixel recurrent neural networks. CoRR abs/1601.06759 (2016)
35. Denton, E., Chintala, S., Szlam, A., Fergus, R.: Deep generative image models using a laplacian pyramid of adversarial networks. In: NIPS (2015)
36. Mirza, M., Osindero, S.: Conditional generative adversarial nets. CoRR abs/1411.1784 (2014)
37. Mathieu, M., Couprie, C., LeCun, Y.: Deep multi-scale video prediction beyond mean square error. CoRR abs/1511.05440 (2015)

38. Im, D.J., Kim, C.D., Jiang, H., Memisevic, R.: Generating images with recurrent adversarial networks. CoRR abs/1602.05110 (2016)
39. Wang, X., Fouhey, D.F., Gupta, A.: Designing deep networks for surface normal estimation. In: CVPR (2015)
40. Eigen, D., Fergus, R.: Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In: ICCV (2015)
41. Fouhey, D.F., Gupta, A., Hebert, M.: Data-driven 3D primitives for single image understanding. In: ICCV (2013)
42. Ladický, L., Zeisl, B., Pollefeys, M.: Discriminatively trained dense surface normal estimation. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 468–484. Springer, Heidelberg (2014). doi:10.1007/978-3-319-10602-1_31
43. Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I.J.: Adversarial autoencoders. CoRR abs/1511.05644 (2015)
44. Larsen, A.B.L., Sønderby, S.K., Winther, O.: Autoencoding beyond pixels using a learned similarity metric. CoRR abs/1512.09300 (2015)
45. Dosovitskiy, A., Brox, T.: Generating images with perceptual similarity metrics based on deep networks. CoRR abs/1602.02644 (2016)
46. Barrow, H.G., Tenenbaum, J.M.: Recovering intrinsic scene characteristics from images. In: Computer Vision Systems (1978)
47. Tenenbaum, J.B., Freeman, W.T.: Separating style and content with bilinear models. In: Neural Computation (2000)
48. Fouhey, D.F., Hussain, W., Gupta, A., Hebert, M.: Single image 3D without a single 3D image. In: ICCV (2015)
49. Zhu, S.C., Wu, Y.N., Mumford, D.: Filters, random fields and maximum entropy (frame): towards a unified theory for texture modeling. In: IJCV (1998)
50. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. CoRR abs/1502.03167 (2015)
51. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: ICML (2013)
52. Xu, B., Wang, N., Chen, T., Li, M.: Empirical evaluation of rectified activations in convolutional network. CoRR abs/1505.00853 (2015)
53. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR (2015)
54. Ladický, L., Shi, J., Pollefeys, M.: Pulling things out of perspective. In: CVPR (2014)
55. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS (2012)
56. Kingma, D., Ba, J.: Adam: a method for stochastic optimization. CoRR abs/1412.6980 (2014)
57. Guo, R., Hoiem, D.: Support surface prediction in indoor scenes. In: ICCV (2013)
58. Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., Oliva, A.: Learning deep features for scene recognition using places database. In: NIPS (2014)
59. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet large scale visual recognition challenge. IJCV **115**(3), 211–252 (2015)
60. Girshick, R.: Fast r-cnn. In: ICCV (2015)
61. Song, S., Lichtenberg, S., Xiao, J.: Sun RGB-D: a RGB-D scene understanding benchmark suite. In: CVPR (2015)

62. Janoch, A., Karayev, S., Jia, Y., Barron, J., Fritz, M., Saenko, K., Darrell, T.: A category-level 3-D object dataset: Putting the kinect to work. In: Workshop on Consumer Depth Cameras in Computer Vision (with ICCV) (2011)
63. Xiao, J., Owens, A., Torralba, A.: SUN3D: a database of big spaces reconstructed using SfM and object labels. In: ICCV (2013)
64. Oliva, A., Torralba, A.: Modeling the shape of the scene: a holistic representation of the spatial envelope. IJCV **42**, 145–175 (2011)
65. Gupta, S., Girshick, R., Arbeláez, P., Malik, J.: Learning rich features from RGB-D images for object detection and segmentation. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8695, pp. 345–360. Springer, Heidelberg (2014). doi:10.1007/978-3-319-10584-0_23
66. Gupta, S., Hoffman, J., Malik, J.: Cross modal distillation for supervision transfer. In: CVPR (2016)