3D-PRNN: Generating Shape Primitives with Recurrent Neural Networks

Chuhang Zou[†] Ersin Yumer[‡] Jimei Yang[‡] Duygu Ceylan[‡] Derek Hoiem[†]
[†]University of Illinois at Urbana-Champaign [‡]Adobe Research
{czou4, dhoiem}@illinois.edu {yumer, jimyang, ceylan}@adobe.com

Abstract

*The success of various applications including robotics, digital content creation, and visualization demand a structured and abstract representation of the 3D world from limited sensor data. Inspired by the nature of human perception of 3D shapes as a collection of simple parts, we explore such an abstract shape representation based on primitives. Given a single depth image of an object, we present **3D-PRNN**, a generative recurrent neural network that synthesizes multiple plausible shapes composed of a set of primitives. Our generative model encodes symmetry characteristics of common man-made objects, preserves long-range structural coherence, and describes objects of varying complexity with a compact representation. We also propose a method based on Gaussian Fields to generate a large scale dataset of primitive-based shape representations to train our network. We evaluate our approach on a wide range of examples and show that it outperforms nearest-neighbor based shape retrieval methods and is on-par with voxel-based generative models while using a significantly reduced parameter space.*

1. Introduction

Many robotics and graphics applications require 3D interpretations of sensory data. For example, picking up a cup, moving a chair, predicting whether a stack of blocks will fall, or looking for keys on a messy desk all rely on at least a vague idea of object position, shape, contact and connectedness. A major challenge is how to represent 3D object geometry in a way that (1) can be predicted from noisy or partial observations; and (2) is useful for reasoning about contact, support, extent, and so on. Recent efforts often focus on voxelized volumetric representations (e.g., [41, 40, 13, 9]). Instead, we propose to represent objects with 3D primitives (oriented 3D rectangles, i.e. cuboids). Compared to voxels, the primitives are much more *compact*, for example 45-D for 5 primitives parameterized by scale-rotation-translation vs 32,256-D for a 32x32x32 voxel grid. Also, primitives are *holistic* — representing an object with a few parts greatly simplifies reasoning about stability, connectedness, and other important properties. Primitive-based 3D object representations have long been popular in psychology (e.g. “geons” by Biederman [3]) and interactive graphics (e.g. “Teddy” [18]), but they are less commonly employed in modern computer vision due to the challenges of learning and predicting models that consist of an arbitrary number of parameterized components.

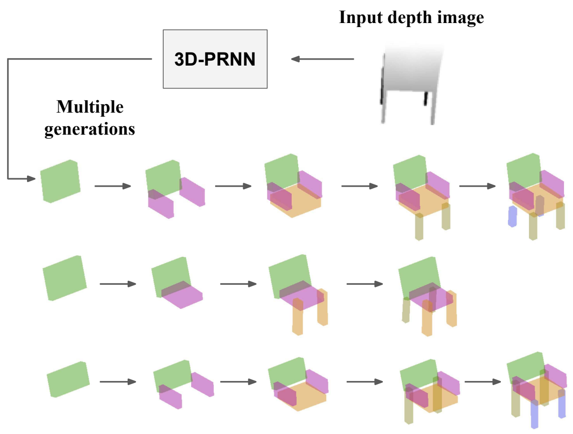


Figure 1. A step-by-step primitive-based shape generation by 3D-PRNN. As an illustration, given single depth image, we sequentially predicts sets of primitives that form the shape. Each time we randomly sample one primitive from a set and generate the next set of primitives conditioning on the current sample.

Our goal is to learn 3D primitive representations of objects from unannotated 3D meshes. We follow an encoder-decoder strategy, inspired by recent work [14, 38], using a recursive neural network (RNN) to encode an implicit shape representation and then sequentially generate primitives to approximate the shape as shown in Fig. 1. One challenge in training such a primitive generation network is acquiring ground truth data for primitive-based shape representations. To address this challenge, we propose an efficient method based on Gaussian Fields and energy minimization [6] to iteratively parse shapes into primitive components. We op-

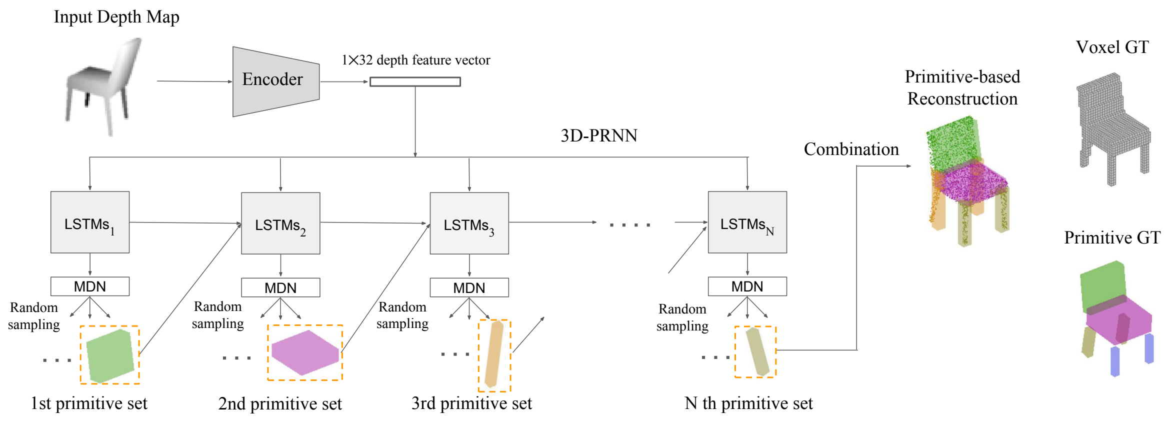


Figure 2. **3D-PRNN overview**. We illustrate the method on the task of single depth shape completion. The network starts from encoding the input depth image into a feature vector, which is then sent to the "recurrent generator" consisting stacks of Long Short-Term Memory (LSTM) and a Mixture Density Network (MDN). At each time step, the network predicts a set of primitives conditioned on both the depth feature and the previously sampled single primitive. The final reconstruction results and ground truth are shown on the right.

timize a differentiable loss function using robust techniques (L-BFGS[43]). We use this (unsupervised) optimization process to create the primitive ground truth, solving for a set of primitives that approximates each 3D mesh in a collection. The RNN can then be trained to generate new primitive-based shapes that are representative of an object class' distribution or to complete an object's shape given a partial observation such as a depth image or point cloud.

To model shapes, we propose 3D-PRNN, an RNN-based generative model that predicts context-sensitive sequences of primitives in object-centric coordinates, as shown in Figure 2. To predict shape from depth, the network is trained jointly with the single depth image and a sequence of primitives configurations (shape, translation and rotation) that form the complete shape. During testing, the network gets input of a depth map and sequentially predicts primitives (ending with a stop signal) to reconstruct the shape. Our generative RNN architecture is based on a Long Short-Term Memory (LSTM) and a Mixture Density Network (MDN).

We evaluate our proposed generative model by comparing with baselines and state-of-the-art methods. We show that, even though our method has less degrees of freedom in representation, it achieves comparable accuracy with voxel based reconstruction methods. We also show that encoding further symmetry and rotation axis constraints in our network significantly boosts performance.

Our main contributions are:

- We propose 3D-PRNN: a generative recurrent neural network that reconstructs 3D shapes as sequences of primitives given a single depth image.

- We propose an efficient method to fit primitives from point clouds based on Gaussian-fields and energy minimization. Our primitive representation provides enough training samples for 3D-PRNN in 3D reconstruction.

2. Related Work

Primitive-based Shape Modeling: Biederman, in the early 1980s, popularized the idea of representing shapes as a collection of components or primitives called "geons" [3], and early computer vision algorithms attempted to recover object-centered 3D volumetric primitives from single images [10]. In computer aided design, primitive-based shape representations are used for 3D scene sketches [42, 31], shape completion from point clouds [33, 24, 32]. In the case that scans of shapes often have canonical parts like planes or boxes and efficient solution for large data is required, primitives are used in reconstructions of urban and architectural scenes [7, 23, 5, 35]. Recently, more compact and parametric representations in the form of template objects [39], and set of primitives [37] have been introduced. These representations, however, require non trivial effort to accommodate variable number of configurations within the object class they are trained for. This is mainly because of their single feed-forward design, which implicitly forces the prediction of a discrete number of variables at the same time.

Object 3D shape reconstruction can be attempted given an RGB image [41, 13, 1, 9]or depth image [40, 29, 12] Recently proposed representations and prediction approaches for 3D data in the context of prediction from sensory input have mainly either focused on part- and object-based retrieval from large repositories [29, 1, 25, 20], or voxelized volumetric representations [41, 40, 13, 9]. A better model fitting includes part deformation [8] and symmetry [22]. Wu et al. [40] present preliminary results on automatic shape completion from depth by classifying hidden voxels with a deep network. Wu et al. [39] reconstruct

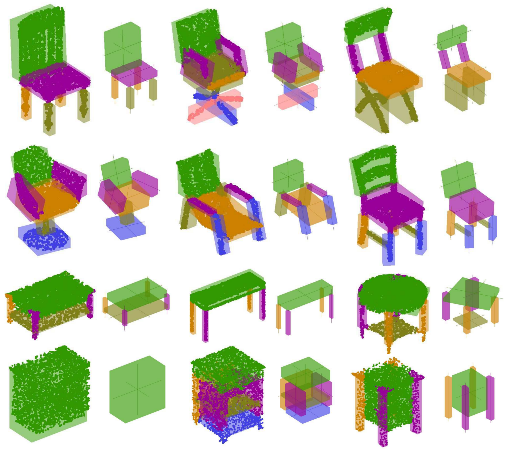


Figure 3. **Sample primitive fitting result**. We show our primitive fitting results on chairs, tables and sofas. We overlay our fitted primitives on the sampled 3D point clouds of each shape.

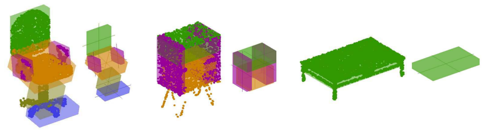


Figure 4. **Failure cases**. Main causes are : too complex shape details to be represented by primitive blocks (left), The smoothing property of Gaussian force fields is not good at describing small hollow shape (middle), small cluster of point clouds are easily missed through our randomized search scheme (middel and right).

shapes based on predicted skeletons. Unlike mesh-based or voxel-based shape reconstruction, our method predicts shapes with aggregations of primitives that has the benefit for lower computational and storage cost.

Generative Models with RNNs: Graves [14] uses Long Short-term Memory recurrent neural networks to generate complex sequences of text and online handwriting. Gregor et al. [15] combine LSTM and a variational auto-encoder, called the Deep Recurrent Attentive Writer (DRAW) architecture, for image generation. The DRAW architecture is a pair of RNNs with an encoder network that compresses the real images presented during training, and a decoder that reconstitutes images after receiving codes. Rezende et al. [19] extend DRAW to learn generative models of 3D structures and recover this structure from 2D images via probabilistic inference. Our 3D-PRNN, which sequentially generates primitives, is inspired by Graves' work to sequentially generate parameterized handwriting strokes and the PixelRNN approach [38] to model natural images as sequentially generated pixels. To produce parameterized 3D primitives (oriented cuboids), we customize the RNN to encode explicit geometric constraints of symmetry and rotation. For example, separately predicting whether a primitive should rotate along each axis and by how much improves results over more simply predicting rotation values, since many objects consist of several (unrotated) cuboids.

3. Fitting Primitives from Point Clouds

One challenge in training our 3D-PRNN primitive generation network is the lack of large scale ground truth primitive based shape reconstruction data. We propose an efficient method to generate such data. Given a point cloud representation of a shape, our approach finds the most plausible primitives to fit in a sequential manner, e.g. given a table, the algorithm might identify the primitive that fits to the top surface first and then the legs successively. We use rectangular cuboids as primitives which provide a plausible abstraction for most man-made objects. Our method proposes a fast parsing solution to decompose shapes with varying complexity into a set of such primitives.

3.1. Primitive Fitness Energy

We formulate the successive fitting of primitives as an energy minimization scheme. While primitive fitting at each step resembles the method of Iterative Closest Point (ICP) [2], we have additional challenges. ICP ensures accurate registration when provided with a good initialization, but in our case we have no prior knowledge about the number and the rough shape of the primitives. Moreover, we need to solve the more challenging partial matching problem since each primitive matches only part of the shape, which we do not know in advance.

We represent the shape of each primitive with scale parameters $S = [s_x, s_y, s_z]$, which denotes the scale of a unit cube along three orthogonal axes. The position and orientation of the primitive are represented by translation, $T = [t_x, t_y, t_z]$, and Euler angles, $\theta = [\theta_x, \theta_y, \theta_z]$, respectively. Thus the primitive is parameterized by $x = [s_x, s_y, s_z, t_x, t_y, t_z, \theta_x, \theta_y, \theta_z]$. Furthermore, we assume a fixed sampling of the unit cube into a set of points, $P = \{p_m\}_{m=1,\ldots,M}$. Given a point cloud representation of a shape, $Q = \{q_n\}_{n=1,\ldots,N}$, our goal is to find the set of primitives $X = \{x_t\}_{t=1,2,3,\ldots}$ that best fit the shape. We employ the idea of Gaussian Force Fields [6] and Truncated Signed Distance Function (TSDF) [27] to formulate the following continuously differentiable energy function which is convex in a large neighborhood of the parameters:

$$E_p = -\sum_{m,n} V_p \min\left(\exp\left(-\frac{\|R(\theta)Sp_m + T - q_n\|^2}{\sigma^2}\right), \xi\right), \tag{1}$$

where $R(\theta)$ is the rotation matrix, ξ is the truncation parameter ($\xi = 0.9$ in our experiments) and V_p denotes the volumetric-wise sampling ratio that is calculated as the volume of primitive P over its number of sampled points M. V_p helps avoid local minimum that results in a too small or too large primitive. Our formulation represents the error as a smooth sum of Gaussian kernels, where far away point pairs are penalized less to account for partial matching.

The energy function given in Eq. 1 is sensitive to the parameter σ. A larger σ will encourage fitting of large primitives while allowing larger distances between matched point pairs. In order to prefer tighter fitting primitives, we introduce the concept of *negative shape*, Q^-, which is represented as a set of points sampled in the non-occupied space inside the bounding box of a shape. We update our energy function as:

$$E_w = E_P^+ - \alpha E_P^-, \tag{2}$$

where E_P^+ is the fitting energy between the shape and the primitive and E_P^- is the fitting energy between the negative shape and the primitive. Given point samples, both E_P^+ and E_P^- are computed as in Eq. 1. α denotes the relative weighting of these two terms and is defined as $\alpha = \max\big(\min(10, |Q|/|Q^-| \times 5), 0.1\big)$.

3.2. Optimization

Given the energy formulation described in the previous section, we perform primitive fitting in a sequential manner. During each iteration, we randomly initialize 10 primitives, optimize Eq. 2 for each of these primitives and add the best fitting primitive to our primitive collection. We then remove the points in Q that are fit by the selected primitive and iterate. We stop once all the points in Q are fit by a primitive. We optimize Eq. 2 in an iterative manner. We first fix θ and solve for S and T, we then fix S and T and solve for θ. In our experiments this optimization converges in 5 iterations and we use the L-BFGS toolbox [30] at each optimization step. We summarize this process with the pseudo-code given in Alg. 1.

Simplification with symmetry. We utilize the symmetry characteristics of man-made shapes to further speed up the primitive parsing procedure. We use axis-aligned 3D objects where symmetric objects have a common global symmetry plane. We compare the geometry on the two sides of this plane to decide whether an object is symmetric or not. Once we obtain a primitive that lies on one side of the symmetry plane, we automatically generate the symmetric primitive on the other side of the plane.

Refinement. At each step, we fit primitives with a relatively larger Gaussian field ($\sigma = 2$ in Eq. 1) for fast convergence and easier optimization. We then refine the fitting with a finer energy space ($\sigma = 0.5$) to match the primitive to the detailed shape of the object. While our random search scheme enables a fast parsing method, errors may accumulate in the final set of primitives. To avoid such problems, we perform a post-refinement step. We refine the parameters of a single primitive x_t while fixing the other parameters. We use the parameters of x_t obtained from the initial fitting as initialization. We define the energy terms in Eq. 2

Algorithm 1 Primitive fitting

```
1:  Given shape point clouds Q and empty primitive set X;
2:  $\beta = 0.97|Q|$, t = 0;
3:  while $|Q| < \beta$ or $i$ <maxPrimNum do
4:      $E_{best}$ = Inf;
5:      for $i = 1$ :maxRandNum do
6:          $\theta = [0,0,0]$, random initialize $S, T, j = 0$;
7:          while $\delta < 0.01$ or $j$ <maxIter do
8:              fix $\theta$, solve $S, T \to S^*, T^*$ by Eq .2;
9:              fix $S^*, T^*$, update $\theta \to \theta^*$ by Eq .2;
10:             calculate $E_w(S^*, T^*, \theta^*)$ by Eq .2;
11:             if $E_w < E_{best}$ then
12:                 $E_{best} = E_w, x_{best} = [S^*, T^*, \theta^*]$;
13:             $\delta = \|[S,T,\theta] - [S^*,T^*,\theta^*]\|^2$;
14:             $S = S^*, T = T_p^*, k = k+1$;
15:     $x_t = x_{best}$, add $x_t$ to $X$, $t = t+1$;
16:     Remove fitted points from $Q$ and add to non-
        occupied space $Q^-$
    return $X$
```

with respect to the points that are fit by x_t and the points that are not fit by any primitive yet. We note that this sequential refinement is similar to back propagation used to train neural networks. In our experiments, we perform the refinement each time we fit 3 new primitives.

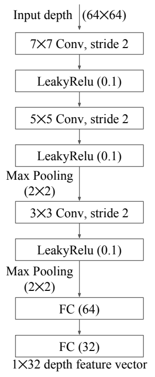
4. 3D-PRNN: 3D Primitive Recurrent Neural Networks

Generating primitive-based 3D shapes is a challenging task due to the complex multi-modal distribution of shapes and the unconstrained number of primitives required to model such complex shapes. We propose 3D-PRNN, a generative recurrent neural network to accomplish this task. 3D-PRNN can be trained to generate novel shapes both randomly and by conditioning on partial shape observations such as a single depth map.

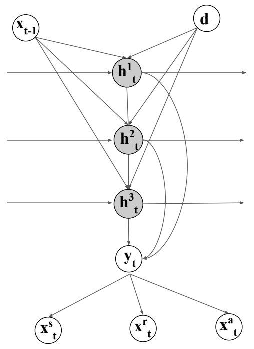
4.1. Network Architecture

An overview of the 3D-PRNN network is illustrated in Fig. 2. The network gets as input a single depth image and sequentially predicts primitives to form a 3D shape. For each primitive, the network predicts its shape (height, length, width), position (i.e. translation), and orientation (i.e. rotation). Additionally, at each step, a binary *end of generation* signal is predicted which indicates no more primitive should be generated.

Depth map encoder. Each input depth map, I, is first resized to be 64×64 in dimension with values in the range $[0, 1]$ (we set the value of background regions to 0). I is passed to an encoder which consists of stacks of convolutional and LeakyRelu [26] layers as shown in Fig. 5 (a): the



(a) Encoder architecture



(b) LSTMs architecture

Figure 5. Detailed architectures of (a) the depth map encoder and (b) the primitive recurrent generator unit in **3D-PRNN**. See the architecture descriptions in Section 4.1.

first layer has 32 kernels of size 7×7 and stride 2, with a LeakyRelu layer of 0.1 slope in the negative part. The second layer consists of 64 kernels of size 5×5 (stride 2), followed by the same setting of LeakyRelu and a max pooling layer. The third layer has 128 kernels of size 3×3 (stride 2) followed by LeakyRelu and max pooling. The next two fully-connected layers has neurons of 64 and 32. The output 1×32 feature vector d is then sent to the recurrent generator to predict a sequence of primitives.

Recurrent generator. We apply the Long Short-Term Memory (LSTM) unit inside the recurrent generator, which is shown to be better at alleviating the vanishing or exploding gradient problems [28] when training RNNs. The architectural design is shown in Fig. 5 (b). The prediction unit consists of L layers of recurrently connected hidden layers (we set $L = 3$, which is found to be sufficient to model the complex primitive distributions) that encode both the depth feature d and the previously predicted primitive x_{t-1} and then computes the output vector, y_t. y_t is used to parametrize a predictive distribution $Pr(x_t|y_t)$ over the next possible primitive x_t. The hidden layer activations are computed by iterating over the following equations in the range $t = [1, T]$ and $l = [2, L]$:

$$z^l_t = W^l_x x_{t-1} + W^l_h h^l_{t-1} + W^l_c h^{l-1}_t + W^l_d d \quad (3)$$

$$[i^l_t, f^l_t, o^l_t] = \sigma(z^l_t) \quad (4)$$

$$g^l_t = \tanh(z^l_t) \quad (5)$$

$$c^l_t = f^l_t c^l_{t-1} + i^l_t g^l_t \quad (6)$$

$$h^l_t = o^l_t \tanh(c^l_t) \quad (7)$$

where z^l_t capsules the input features in the l-th layer (when $l = 1$, there is no hidden value propagated from the previous layers and thus $z^1_t = W^1_x x_t + W^1_h h^1_{t-1} + W^1_c d$), h_t and c_t denote the hidden and cell states, whereas $W^l_x, W^l_h, W^l_c, W^l_d$ denote the linear weight matrix (we omit the bias term for brevity), i_t, f_t, o_t, g_t are respectively the input, forget, output, and context gates, which have the same dimension as the hidden states (size of 400). σ is the logistic sigmoid function and $\tanh$ is the hyperbolic tangent function.

At each time step t, the distribution of the next primitive is predicted as $y_t = W_y[h^1_t, h^2_t, ..., h^L_t]$, where we perform a linear transformation on the concatenation of all the hidden values. This concatenation is similar in spirit to using *skip connections* [36, 17], which is shown to help training and mitigate the vanishing gradient problem. In a similar fashion, we also pass the depth feature d to all the hidden layers. We will explain latter how the primitive configuration x_t is sampled from a distribution predicted from y_t.

We predict parameters of one axis per time conditioned on the previous axis. We model this joint distribution of parameter on each axis $x^s_i = [s_i, t_i]$ (where i indicates one of the 3 axes of space) as a mixture of Gaussians conditioned on previous axis with K mixture components:

$$\left(\{\pi^k_t, \mu^k_t, \sigma^k_t, \rho^k_t\}^K_{k=1}, e_t\right) = f(y_t), \quad (8)$$

where π_t, μ_t, σ_t and ρ_t are the weight, mean, standard deviation, and correlation of each mixture component respectively, predicted from a fully connected layer $f(y_t)$. Note that e_i is the binary stopping sign indicating whether the current primitive is the final one and it helps with predicting a variable-length sequence of primitives. In our experiments we set $K = 20$. We randomly sample a single instance $x^s_i = [s_i, t_i, e_i] \in \mathbb{R} \times \mathbb{R} \times \{0, 1\}$ drawn from the distribution $f(y_t)$. The sequence x_t represents the parameters in the following order: $x^s_x \rightarrow x^s_y \rightarrow x^s_z$ for the shape translation configuration on x, y, z axis of the first primitive and the stopping sign.

This is essentially a mixture density network (MDN) [4] on top of the LSTM output and its loss is defined:

$$L_s(x) = \sum^T_{t=1} -\log\big(\sum_k \pi^k_t N(x_{t+1}|\mu^k_t, \sigma^k_t, \rho^k_t)\big) \\ - \mathbb{I}\big((x_{t+1})_3 = 1\big)\log e_t - \mathbb{I}\big((x_{t+1})_3 \neq 1\big)\log(1 - e_t) \quad (9)$$

The MDN is trained by maximizing the log likelihood of ground truth primitive parameters in each time step, where we calculate gradients explicitly for backpropagation as shown by Graves [14]. We found this stepwise supervised training works well and avoids sequential sampling used in [37, 11].

Geometric constraints. Another challenge in predicting primitive-based shape is to model rotation, given that the rotation axis is sensitive to slight change in rotation values under Euler angles. We found that by jointly predicting the rotation axis x^a and the rotation value x^r, both the rotation prediction performs better and the overall primitive distribution modeling get alleviated as shown in Fig. 6, quantitative experiments are in Sec. 5.3. The rotation axis (x^a) is

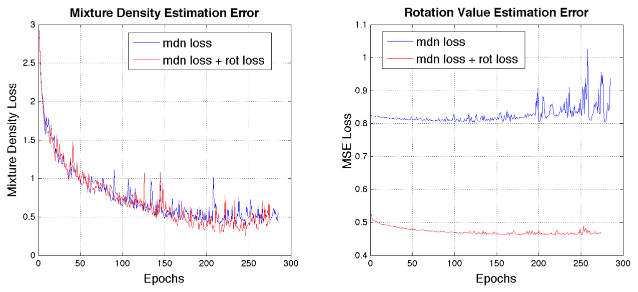


Figure 6. **Training performance comparison on validation set of synthetic depth map from ModelNet.** Both the mixture density loss and the rotation MSE loss are averaged by sequence length. The rotation values are normalized and values can have ranges around 13, compared with the < 1 MSE loss. Our mixture density estimation and rotation value estimation performs better by enforcing loss on predicting rotation axis.

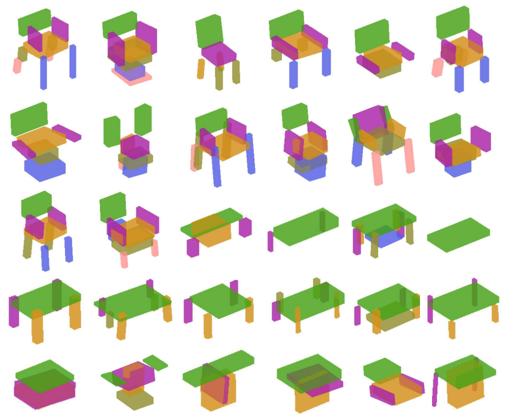


Figure 7. **Shape synthesis result**. We show various random sampled shapes by our 3D-PRNN. The network is trained and tested without context input. The coloring indicates the prediction order.

predicted by a three-layered fully connected network $g(y_t)$ with size 64,32 and 1 and sigmoid function as shown in fig. 5. The rotation value (x^r) is predicted by a separate three-layered fully connected network $g^*(y_t)$ with size 64, 32 and 1 and a $Tanh(\cdot)$ function.

4.2. Loss Function

The overall sequence loss of our network is:

$$L(x) = L_s(x^s) + L_r(x^r) + L_a(x^a), \tag{10}$$

$$L_r(x) = \sum_t \|x_t^r - \hat{x_t^r}\|^2 \tag{11}$$

$$L_a(x) = \sum_t \|x_t^a - \hat{x_t^a}\|^2 \tag{12}$$

$L_s(x)$ is defined in Eq. 9. $L_r(x)$ is a mean square loss between predicted, x_t^r, and target, $\hat{x_t^r}$, rotation. $L_a(x)$ is the mean square loss between the predicted, x_t^a, and ground truth, $\hat{x_t^a}$, rotation axis.

5. Experiments and Discussions

We show quantitative results on automatic shape synthesis. We quantitatively evaluate our 3D-PRNN in two tests: 1) 3D reconstruction on synthetic depth maps and 2) using real depth maps as input.

We train our 3D-PRNN on ModelNet [40] categories: 889 chairs, 392 tables and 200 nightstands. We employ the provided another 100 testing samples from each class for evaluation. We train a single network with all shapes classes jointly. In all experiments, to avoid overfitting, we hold out 15% of the training samples, which are then used to choose the number of training epochs. We then retrain the network using the entire training set. Since a single network is trained to encode all three classes, when predicting shape from depth images, for example, there is an implicit class prediction as well.

5.1. Implementation

We implement 3D-PRNN network using Torch. We train our network on primitive-based shape configurations generated as described in Sec. 3. The parameters of each primitive (i.e. shape, translation and rotation) are normalized to have zero mean and standard deviation. We observe that the order of the primitives generated by the method described in Sec. 3 involves too much randomness that makes training hard. Instead, we pre-sort the primitives based on the height of each shape center in a decreasing fashion. This simple sorting strategy significantly boosts the training performance. Additionally, our network is trained only on one side of the symmetric shapes to shorten the sequence length and speed up the training process. To train with the generative mechanism, we use simple random sampling technique. We use ADAM [21] to update network parameters with a learning rate of 0.001, $\alpha = 0.95$, and $\epsilon = e^{-6}$. We train the network with batch size 380 and 50 on the synthetic data and on the real data respectively.

At test time, the network takes a single depth map and sequentially generates primitives until a stop sign is predicted. To initialize the first RNN feature x, we perform a nearest neighbor query based on the encoded feature of the depth map to retrieve the most similar shape in the training set and use the configuration on its first primitive.

5.2. Shape Synthesis

3D-PRNN can be trained to generate new primitive-based shapes. Fig. 7 shows our randomly generated shapes synthesized from all three shape classes. We initialize the first RNN feature x with a random sampled primitive configuration from the training set. Since the first feature corresponds to "width", "translation in x-axis", and "rotation on x-axis" of the primitive, formally this initialization process is defined as drawing a sample from a discrete uniform distribution of these parameters where the discrete samples are

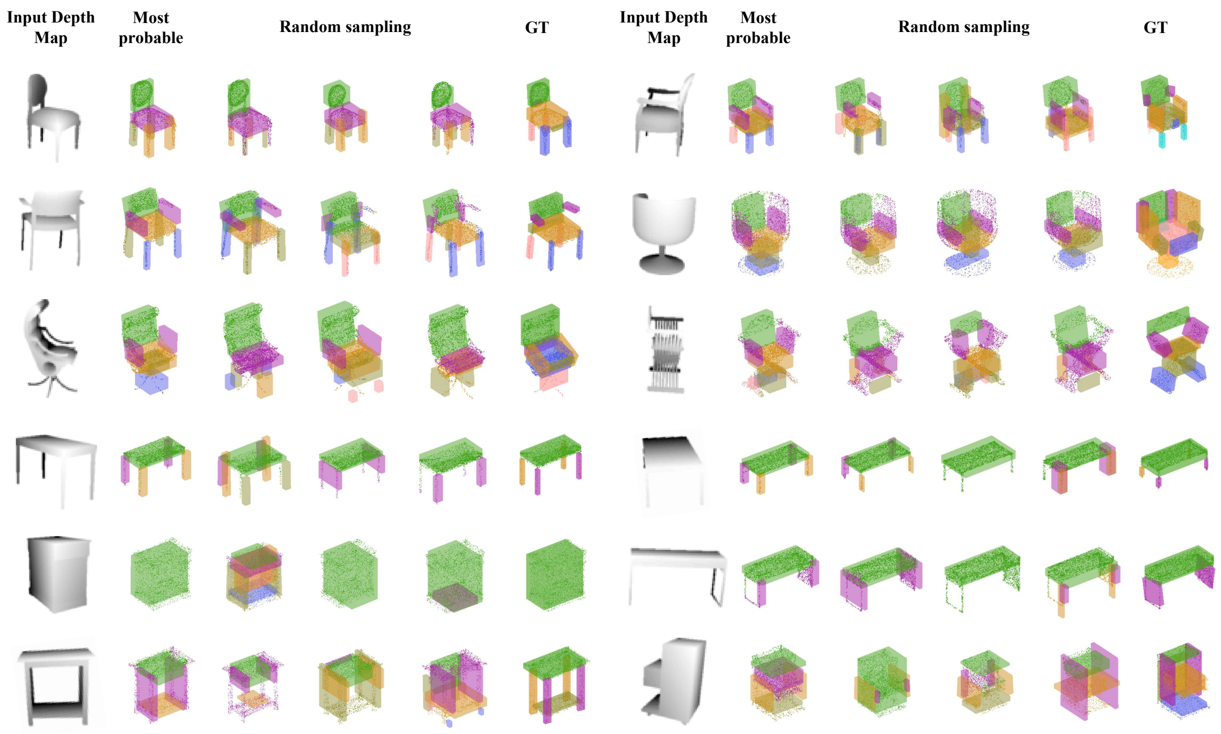


Figure 8. **Sample reconstruction from synthetic data from ShapeNet**. We show the input depth map, with the most probable shape reconstruction from 3D-PRNN, and three successive random sampling results, compared with our ground truth primitive representation.

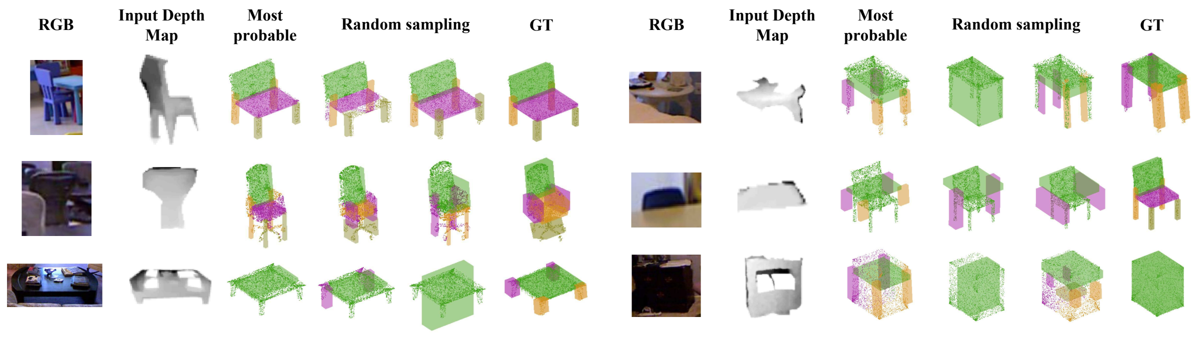


Figure 9. **Sample reconstruction from real depth map in NYUdv2**. We show the input depth map, with the most probable shape reconstruction from 3D-PRNN, and two successive random sampling results, compared with our ground truth primitive representation.

constructed from the training examples. The figure shows that 3D-PRNN can learn to generate representative samples from multiple classes and sometimes creates hybrids from multiple classes.

5.3. Shape Reconstruction from Single Depth View

Synthetic data. We project synthetic depth maps from training meshes. For both training and testing, we perform rejection-sampling on a unit sphere for 5 views, bounded within 20 degrees of the equator. The complete 3D shape is then predicted using a single depth map as input to 3D-PRNN. Our model can generate a sampling of complete shapes that match the input depth, as well as the most likely configuration, determined as the mean of the Gaussian from the most probable mixture. We report 3D intersection over union (IoU) and surface-to-surface distance [29] of the most likely predicted shape to the ground truth mesh. To compute IoU, the ground truth mesh is voxelized to 30 x 30 x 30 resolution, and IoU is calculated based on whether the voxel centers fall inside the predicted primitives or not. Surface-to-surface distance is computed using 5,000 points sampled on the primitive and ground truth surfaces, and the distance is normalized by the diameter of a sphere tightly fit to the ground truth mesh (e.g. 0.05 is 5% of object maximum dimension).

Tables 1 and 2 show our quantitative results. "GT prim" is the ground truth primitive representation generated by our parsing optimization method during training. This serves

	chair	table	night stand
GT prim	0.473	0.533	0.657
NN Baseline	**0.269**	0.220	0.256
Wu et al. [40] (mean)	0.253	0.250	**0.295**
3D-PRNN	0.245	0.188	0.204
3D-PRNN + rot loss	0.238	**0.263**	0.266

Table 1. Shape IoU evaluation in synthetic depth map in ModelNet. We explore two settings of 3D-PRNN with or without rotation axis constrains, and compare it with ground truth primitive and the nearest neighbor baseline. We also compare to the Wu et al. [40] deep network voxel generation method.

	chair	table	night stand
GT prim	0.049	0.044	0.044
NN baseline	0.075	0.089	0.100
Wu et al. [40] (mean)	**0.045**	**0.035**	**0.057**
3D-PRNN	0.074	0.080	0.104
3D-PRNN + rot loss	0.074	0.078	0.092

Table 2. Surface-to-surface distance evaluation in synthetic depth map in ModelNet. We explore two settings of 3D-PRNN with or without rotation axis constrains, and compare it with ground truth primitive and the nearest neighbor baseline.

as an upper bound on performance by our method, corresponding to how well the primitive model can fit the true meshes. "NN Baseline" is the nearest neighbor retrieval of shape in training set based on the embedded depth feature from our network. By enforcing rotation axis constraints ("3D-PRNN + rot loss"), our 3D-PRNN achieves better performance, which conforms with the learning curve as shown in Fig. 6. Though both nearest neighbor and 3D-PRNN are based on the trained encoding, 3D-PRNN outperforms NN Baseline for table and nightstand, likely because it is able to generate a greater diversity of shapes from limited training data. We compare with the voxel-based reconstruction of Wu et al. [40], training and testing their method on the same data using publicly available code. Since Wu et al. generate randomized results, we measure the average result over ten runs. Our method performs similarly to Wu et al. [40] on the IoU measure. Wu et al. performs better on surface distance, which is less sensitive to alignment but more sensitive to details in structures. The performance of our ground truth primitives confirms that much of our reduced performance in surface distance is due to using a coarser abstraction (which though not preserving surface detail has other benefits, as discussed in introduction).

Real data (NYU Depth V2). We also test our model on NYU Depth V2 dataset [34] which is much harder than synthetic due to limited training data and the fact that depth images of objects are in lower resolution, noisy, and often occluded conditions. We employ the ground truth data labelled by Guo and Hoiem [16], where 30 models are manually selected to represent 6 categories of common furniture: chair, table, desk, bed, bookshelf and sofa. We fine-tune our network that was trained on synthetic data using the training

class	chair	table	night stand
GT prim	0.037	0.048	0.020
NN baseline+ft	0.118	0.176	0.162
NN baseline	**0.101**	**0.164**	**0.160**
3D-PRNN+ft	0.112	0.168	0.192
3D-PRNN	0.110	0.181	0.194

Table 3. Surface-to-surface distance evaluation in real depth map in NYUd v2. We explore two settings of 3D-PRNN with (+ft) or without fine-tuning, and compare it with ground truth primitive and the nearest neighbor baseline.

class	chair	table	night stand
GT prim	0.543	0.435	0.892
NN baseline +ft	**0.171**	**0.078**	**0.286**
NN baseline	0.145	0.076	0.262
3D-PRNN +ft	0.158	0.075	0.081
3D-PRNN	0.138	0.052	0.086

Table 4. Shape IoU evaluation in real depth map in NYUd v2. We explore two settings of 3D-PRNN with (+ft) or without fine-tuning, and compare it with ground truth primitive and the nearest neighbor baseline.

set of NYU Depth V2. We report results on test set based on the same evaluation metric as the synthetic test shown in Table 4 and 3. Since nightstand is less common in the training set and often occluded depth regions may be similar to those for tables, the network often predicts primitives in the shapes of tables or chairs for nightstands, resulting in worse performance for that class. Sample qualitative results are shown in Fig. 9.

3D Shape Segmentation. Since our primitive based reconstructions are following meaningful part configurations naturally, another application where our method can apply is shape segmentation. Please refer to our supplemental material for shape segmentation task details and results, where we compare with state of the art methods as well.

Conclusions and Future Work. We present 3D-PRNN, a generative recurrent neural network that uses recurring primitive based abstractions for shape synthesis. 3D-PRNN models complex shapes with a low parametric model, which advantages such as being capable of modeling shapes with fewer training examples available, and a large intra- and inter-class variance. Evaluations on synthetic and real depth map reconstruction tasks show that results comparable to higher degree of freedom representations can be achieved with our method. Future explorations include allowing various primitive configurations beyond cuboids (i.e. cylinders or spheres), encoding explicit joints and spatial relationship between primitives.

Acknowledgements

This research is supported in part by NSF award 14-21521 and ONR MURI grant N00014-16-1-2007. We thank David Forsyth for insightful comments and discussion.

References

[1] M. Aubry, D. Maturana, A. A. Efros, B. C. Russell, and J. Sivic. Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models. In *IEEE CVPR*, pages 3762–3769, 2014. 2

[2] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE PAMI*, 14(2):239–256, 1992. 3

[3] I. Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115, 1987. 1, 2

[4] C. M. Bishop. Mixture density networks. 1994. 5

[5] A. Bódis-Szomorú, H. Riemenschneider, and L. Van Gool. Fast, approximate piecewise-planar modeling based on sparse structure-from-motion and superpixels. In *IEEE CVPR*, pages 469–476, 2014. 2

[6] F. Boughorbel, M. Mercimek, A. Koschan, and M. Abidi. A new method for the registration of three-dimensional point-sets: The gaussian fields framework. *Image and Vision Computing*, 28(1):124–137, 2010. 1, 3

[7] A.-L. Chauve, P. Labatut, and J.-P. Pons. Robust piecewise-planar 3d reconstruction and completion from large-scale unstructured point data. In *IEEE CVPR*, pages 1261–1268, 2010. 2

[8] T. Chen, Z. Zhu, A. Shamir, S.-M. Hu, and D. Cohen-Or. 3-sweep: Extracting editable objects from a single photo. *ACM TOG*, 32(6):195, 2013. 2

[9] 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*, pages 628–644, 2016. 1, 2

[10] S. J. Dickinson, A. Rosenfeld, and A. P. Pentland. Primitive-based shape modeling and recognition. In *Visual Form*, pages 213–229. 1992. 2

[11] S. A. Eslami, N. Heess, T. Weber, Y. Tassa, D. Szepesvari, G. E. Hinton, et al. Attend, infer, repeat: Fast scene understanding with generative models. In *Advances in Neural Information Processing Systems*, pages 3225–3233, 2016. 5

[12] M. Firman, O. Mac Aodha, S. Julier, and G. J. Brostow. Structured prediction of unobserved voxels from a single depth image. In *IEEE CVPR*, pages 5431–5440, 2016. 2

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

[14] A. Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013. 1, 3, 5

[15] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra. DRAW: A recurrent neural network for image generation. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015*, pages 1462–1471, 2015. 3

[16] R. Guo and D. Hoiem. Support surface prediction in indoor scenes. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2144–2151, 2013. 8

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 5

[18] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: a sketching interface for 3d freeform design. In *ACM SIGGRAPH '99*, pages 409–416. 1

[19] D. Jimenez Rezende, S. M. A. Eslami, S. Mohamed, P. Battaglia, M. Jaderberg, and N. Heess. Unsupervised learning of 3d structure from images. In *Advances in Neural Information Processing Systems 29*, pages 4996–5004. Curran Associates, Inc., 2016. 3

[20] N. Kholgade, T. Simon, A. Efros, and Y. Sheikh. 3d object manipulation in a single photograph using stock 3d models. *ACM TOG*, 33(4):127, 2014. 2

[21] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 6

[22] C. Kurz, X. Wu, M. Wand, T. Thormählen, P. Kohli, and H.-P. Seidel. Symmetry-aware template deformation and fitting. In *CGF*, volume 33, pages 205–219, 2014. 2

[23] F. Lafarge and P. Alliez. Surface reconstruction through point set structuring. In *Computer Graphics Forum*, volume 32, pages 225–234, 2013. 2

[24] Y. Li, X. Wu, Y. Chrysathou, A. Sharf, D. Cohen-Or, and N. J. Mitra. Globfit: Consistently fitting primitives by discovering global relations. In *ACM TOG*, volume 30, page 52, 2011. 2

[25] J. J. Lim, H. Pirsiavash, and A. Torralba. Parsing ikea objects: Fine pose estimation. In *IEEE CVPR*, pages 2992–2999, 2013. 2

[26] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, volume 30, 2013. 4

[27] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *IEEE ISMAR*, pages 127–136, 2011. 3

[28] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. *ICML*, 28:1310–1318, 2013. 5

[29] J. Rock, T. Gupta, J. Thorsen, J. Gwak, D. Shin, and D. Hoiem. Completing 3d object shape from one depth image. In *IEEE CVPR*, pages 2484–2493, 2015. 2, 7

[30] M. Schmidt. minfunc: unconstrained differentiable multivariate optimization in matlab. *URL http://www.di.ens.fr/mschmidt/Software/minFunc.html*, 2012. 4

[31] R. Schmidt, B. Wyvill, M. C. Sousa, and J. A. Jorge. Shapeshop: Sketch-based solid modeling with blobtrees. In *ACM SIGGRAPH 2007 courses*, page 43, 2007. 2

[32] R. Schnabel. *Efficient point-cloud processing with primitive shapes.* PhD thesis, University of Bonn, 2010. 2

[33] R. Schnabel, P. Degener, and R. Klein. Completion and reconstruction with primitive shapes. In *CGF*, volume 28, pages 503–512, 2009. 2

[34] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, pages 746–760, 2012. 8

[35] S. N. Sinha, D. Steedly, R. Szeliski, M. Agrawala, and M. Pollefeys. Interactive 3d architectural modeling from unordered photo collections. In *ACM TOG*, volume 27, page 159, 2008. 2

[36] R. K. Srivastava, K. Greff, and J. Schmidhuber. Training very deep networks. In *NIPS*, pages 2377–2385, 2015. 5

[37] S. Tulsiani, H. Su, L. J. Guibas, A. A. Efros, and J. Malik. Learning shape abstractions by assembling volumetric primitives. *arXiv preprint arXiv:1612.00404*, 2016. 2, 5

[38] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016*, pages 1747–1756, 2016. 1, 3

[39] 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*, pages 365–382, 2016. 2

[40] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015. 1, 2, 6, 8

[41] 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*, pages 1696–1704, 2016. 1, 2

[42] R. C. Zeleznik, K. P. Herndon, and J. F. Hughes. Sketch: An interface for sketching 3d scenes. In *ACM SIGGRAPH 2007 courses*, page 19. ACM, 2007. 2

[43] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560, 1997. 2