Scene Synthesis via Uncertainty-Driven Attribute Synchronization

Haitao Yang[1] Zaiwei Zhang[1] Siming Yan[1] Haibin Huang[2] Chongyang Ma[2]
Yi Zheng[2] Chandrajit Bajaj[1] Qixing Huang[1]
[1]The University of Texas at Austin [2]Kuaishou Technology

Abstract

Developing deep neural networks to generate 3D scenes is a fundamental problem in neural synthesis with immediate applications in architectural CAD, computer graphics, as well as in generating virtual robot training environments. This task is challenging because 3D scenes exhibit diverse patterns, ranging from continuous ones, such as object sizes and the relative poses between pairs of shapes, to discrete patterns, such as occurrence and co-occurrence of objects with symmetrical relationships. This paper introduces a novel neural scene synthesis approach that can capture diverse feature patterns of 3D scenes. Our method combines the strength of both neural network-based and conventional scene synthesis approaches. We use the parametric prior distributions learned from training data, which provide uncertainties of object attributes and relative attributes, to regularize the outputs of feed-forward neural models. Moreover, instead of merely predicting a scene layout, our approach predicts an over-complete set of attributes. This methodology allows us to utilize the underlying consistency constraints among the predicted attributes to prune infeasible predictions. Experimental results show that our approach outperforms existing methods considerably. The generated 3D scenes interpolate the training data faithfully while preserving both continuous and discrete feature patterns.

1. Introduction

3D scene synthesis is a fundamental problem in deep generative modeling. This task is challenging because 3D scenes exhibit diverse patterns, ranging from continuous ones, such as the size of each object and the relative poses between pairs of shapes, to discrete patterns, such as occurrence and co-occurrence of objects and symmetric relations. Moreover, there are also generic geometric constraints, e.g., synthesized objects in a 3D scene should not inter-penetrate. Developing neural networks to capture all feature patterns while enforcing geometric constraints remains an open problem. Due to the diversity of feature patterns and constraints, the popular approach of developing a single data representation and training approach proves insufficient.

Figure 1: Randomly generated scenes (left) and their nearest neighbours (right) in the training set in 3D-FRONT.

This paper introduces a novel approach to synthesizing 3D scenes represented as a collection of objects. Each object is encoded by its attributes such as size, pose, existence indicator, and geometric codes (c.f. [42, 56]). The theme of our approach is to look at 3D scene synthesis from hybrid viewpoints. Our goal is to combine the strengths of different approaches and representations that can capture diverse feature patterns and enforce different constraints. We execute this hybrid methodology at two levels.

First, instead of merely synthesizing the absolute attributes of each individual object, our approach predicts an *over-complete* set of attributes which also include relative attributes (e.g., relative poses) between object pairs. Such relative attributes better capture spatial correlations among objects compared to only synthesizing absolute attributes. From a robust optimization point of view, over-complete attributes possess generic consistency constraints, e.g., the relative attributes should be consistent with object attributes. These constraints allow us to prune infeasible attributes

in synthesis output (c.f. [17, 10, 19, 2, 21, 54, 15, 48, 39, 55, 51]). This approach is particularly suitable for neural outputs that exhibit weak correlations due to random initialization [55, 14, 38, 29]. We can therefore suppress output errors effectively by enforcing the consistency constraints among absolute and relative attributes.

Second, our approach combines the strengths of neural scene synthesis models and conventional scene generation methods. Neural models possess unbounded expressibility and can encode both continuous and discrete patterns. However, they typically produce single outputs that do not possess useful signals of uncertainties for synchronizing object attributes and relative attributes. For example, suppose we know the uncertainty of object attribute is high. In such cases, we can replace it with another one based on the attributes of other objects and the corresponding relative attributes. Similarly, we can discard a relative attribute if its uncertainty is high. Our approach addresses this issue by learning parametric prior distributions of absolute and relative attributes. Such distributions provide uncertainties of generated object attributes and relative attributes, offering rich signals to regularize them and prune outliers. Moreover, they also help enforce the penetration-free constraints. We introduce a Bayesian framework to integrate neural outputs and parametric prior distributions seamlessly. The hyperparameters of this Bayesian framework are optimized to maximize the performance of the final output.

We evaluate our approach on 3D-FRONT [12]. We also provide results on SUNCG [40] to provide sufficient comparisons with baseline techniques. Experimental results show that our approach can generate 3D scenes different from the training examples while preserving discrete and continuous feature patterns. Our method outperforms baseline approaches both qualitatively and quantitatively. An ablation study justifies the design choices of our approach. Our code is available at https://github.com/yanghtr/Sync2Gen.

2. Related Work

3D scene synthesis has been studied considerably in the past. We refer to [53] for a recent survey and to [8] for a recent tutorial on this topic.

Conventional scene synthesis approaches. Non-deep learning scene synthesis approaches fall into two categories. The first category applies data-driven and non-parametric approaches [13, 26, 7, 37, 50, 18]. The advantages of these methods are that they can handle datasets with significant structural variability. The downside is that these methods require complicated systems and careful parameter tuning. Another category employs probabilistic graphical models (e.g., Bayesian networks) for assembly-based modeling and synthesis [31, 6, 22, 11, 5, 9, 30, 23], these methods show improved performance but reply on hand-crafted modeling of structural patterns. Moreover, they typically only capture low-order correlations among the objects. Similar to these approaches, our approach learns distributions of object attributes and relative attributes. However, unlike using them to synthesize 3D scenes directly, we use the distributions to regularize and prune neural outputs. Most signals of the final output still come from the neural outputs.

Deep scene synthesis approaches. Recent scene synthesis approaches use deep neural networks. Many of them focus on recurrent formulations [27, 33, 16, 36, 57, 46, 34, 28, 45] that model relations between objects. These approaches either explicitly or implicitly utilize a hierarchical structure among the objects. Another category consists of feed-forward models [43, 32, 56] that synthesize a scene layout directly. Our approach advances the state-of-the-art on neural scene models in two ways. First, we propose synthesizing both object attributes and relative attributes and leveraging the underlying consistency constraints to prune wrong synthesis results. Second, we utilize parametric prior distributions to regularize synthesized object attributes and relative attributes further.

Over-complete predictions. Several recent works [48, 39, 55, 51] studied the methodology of first predicting over-complete intermediate geometric representations and then aggregating them into the final output. While our approach also predicts over-complete constraints, i.e., object attributes and relative attributes. We introduce how to integrate prior distributions that provide uncertainties of the predictions. Moreover, the hyperparameters of the synchronization procedure are jointly optimized.

Synchronization. Our approach is also relevant to recent works on transformation synchronization [1, 17, 47, 4, 10, 19, 2, 21, 54, 15, 20], which takes relative transformations among a collection of geometric objects as input and outputs absolute transformations across the entire object collection. Our setting's major difference is to model prior distributions on the relative attributes (including relative poses), which provide uncertainties for inputs to arrangement optimization. Such uncertainties enable us to detect outliers more robustly.

3. Overview

Problem statement. Our goal is to train a variational auto-encoder that takes a 3D scene as input, maps it to a latent code, and finally decodes the latent code into the original input. In this paper, we assume that each scene is an arrangement of objects. Each object is given by its attributes, i.e., category label, size, pose, and a vector (i.e., a shape code) that encodes its geometry. We assume each instance of the training dataset is pre-segmented into these objects.

Approach overview. Similar to [44, 56], we model each scene as selecting and deforming objects from a pre-defined over-complete set of objects. Each object is encoded by a vector that concatenates its attributes and a binary indicator

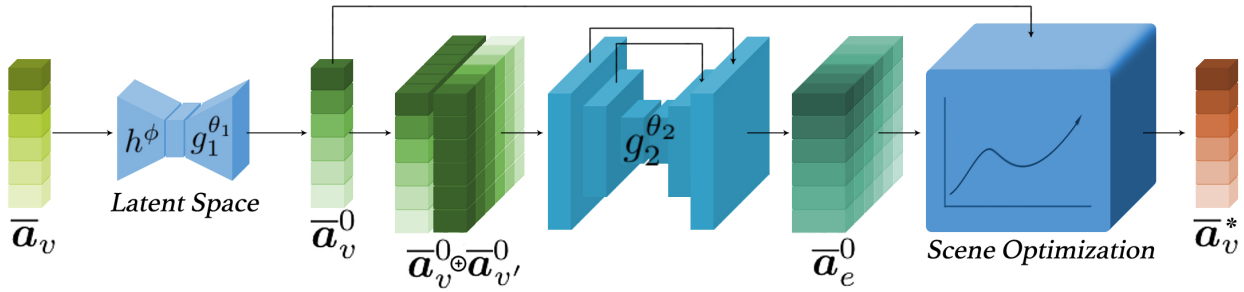


Figure 2: Our network has three modules. The first module is a VAE model on object attributes. During testing, our approach takes a latent code as input and outputs synthesized object attributes. The induced relative attributes are fed into the second VAE module, which outputs synthesized relative attributes. The third module performs Bayesian scene optimization, combining synthesized object and relative attributes and parametric prior distributions to output the final object attributes.

that specifies whether an object is selected or not. A scene is then represented as a matrix, whose columns encode objects.

Unlike standard approaches of designing a feed-forward network that directly outputs the object attributes (c.f. [44, 56]), our method combines two new ideas for scene synthesis. First, our generative model outputs object attributes and relative attributes (e.g., the relative pose between a pair of objects). As shown in Figure 2, this is done by learning a VAE to synthesize object attributes. We will also use the latent space of this VAE to synthesize new scenes. The induced relative attributes from the synthesized object attributes are then fed into another VAE to produce synthesized relative attributes.

The neural outputs provide an over-complete set of constraints on the final object attributes. We can recover accurate object attributes by exploring the underlying consistency constraints among this over-complete set even though some predictions are incorrect. Second, we learn prior distributions of object attributes and relative attributes from the training data. These prior distributions provide uncertainties of the predictions, which further regularize the final output. Note that the prior distributions consider both the continuous variables such as relative poses and discrete variables such as object counts and co-occurrences.

We introduce a Bayesian scene optimization framework that seamlessly integrates neural predictions and prior distributions (See Figure 2). This framework exhibits two novel properties. First, it relaxes object indicators as real variables and employs continuous optimization to refine the object attributes. We show how to solve the induced optimization problem effectively via alternating minimization. Second, we introduce a simple and practical approach to optimize hyperparameters of the induced objective function.

Note that our approach decouples training of the neural synthesis modules and learning of hyperparameters for the Bayesian scene optimization framework. Specifically, the neural synthesis modules are trained on a large-scale training set $\mathcal{T}$. In contrast, the hyperparameters are trained on a separate small-scale validation dataset $\mathcal{T}_{val}$. This approach allows us to alleviate the gap between the distribution of neural synthesis outputs on the training set and that on the testing set. We have found that this approach offers better results than naive end-to-end learning.

4. Bayesian Scene Optimization

This section presents the Bayesian scene optimization framework, which is the main contribution of this paper. It integrates the output of the variational auto-encoders which predict object attributes and relative attributes (see Section 5 for the details) and parametric prior distributions learned from the training data. In the following, we first introduce the problem statement of scene optimization in Section 4.1. We then describe the Bayesian formulation in Section 4.2. Section 4.3 presents the strategy for solving the induced optimization problem. Finally, we describe hyperparameter optimization in Section 4.4.

4.1. Problem Setup

We model a scene as a pre-defined graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where graph vertices encode objects and where edges connect pairs of objects. Each vertex $v \in \mathcal{V}$ is associated with a pre-defined class label $c_v \in \mathcal{C}$, where $\mathcal{C}$ denote all the classes. Scene optimization amounts to recover each object's attributes encoded as a vector $\boldsymbol{a}_v$ and the indicator $z_v \in \{0, 1\}$ that specifies whether v is active or not. In other words, $\{z_v\}$ characterize which objects appear in a scene, and $\{\boldsymbol{a}_v\}$ specify the scene layout. Note that the precise parameterization of $\boldsymbol{a}_v$ will be described in Section 5.1.

Denote $\overline{\boldsymbol{a}}_v = (\boldsymbol{a}_v^T, z_v)^T$. The input to scene optimization consists of a prediction $\overline{\boldsymbol{a}}_v^0$ associated with each vertex $v \in \mathcal{V}$ and a prediction $\overline{\boldsymbol{a}}_e^0$ associated with each edge $e = (v, v') \in \mathcal{E}$, where the ground-truth $\overline{\boldsymbol{a}}_e = \phi(\overline{\boldsymbol{a}}_v, \overline{\boldsymbol{a}}_{v'})$ encodes the relative attributes between $\boldsymbol{a}_v$ and $\boldsymbol{a}_{v'}$ (e.g., relations between beds and nightstands). The explicit expression of $\overline{\boldsymbol{a}}_e$ and ϕ are introduced in Section 5.2. Note that both $\overline{\boldsymbol{a}}_v^0$ and $\overline{\boldsymbol{a}}_e^0$ are outputs of neural networks.

Besides the vertex predictions $\{\overline{\boldsymbol{a}}_v^0\}$ and edge predictions $\{\overline{\boldsymbol{a}}_e^0\}$, our approach also utilizes prior distributions learned

from the input data. The predictions and prior distributions are combined in a Bayesian framework.

4.2. Formulation

We formulate scene optimization as maximizing the following posterior distribution

$$P\big(\{\overline{\boldsymbol{a}}_v\}|\{\overline{\boldsymbol{a}}_v^0\}\cup\{\overline{\boldsymbol{a}}_e^0\}\big) \sim P\big(\{\overline{\boldsymbol{a}}_v^0\}\cup\{\overline{\boldsymbol{a}}_e^0\}|\{\overline{\boldsymbol{a}}_v\}\big)\cdot P\big(\{\overline{\boldsymbol{a}}_v\}\big). \quad (1)$$

where $P(\{\overline{\boldsymbol{a}}_v^0\}\cup\{\overline{\boldsymbol{a}}_e^0\}|\{\overline{\boldsymbol{a}}_v\})$ and $P(\{\overline{\boldsymbol{a}}_v\})$ are total likelihood and prior terms, respectively and $\sim$ denotes equal up to a scaling constant.

Likelihood modeling. We model the total likelihood term by multiplying unary terms and pairwise terms associated with vertices and edges:

$$P\big(\{\overline{\boldsymbol{a}}_v^0\}\cup\{\overline{\boldsymbol{a}}_e^0\}|\{\overline{\boldsymbol{a}}_v\}\big) \sim \prod_{v\in\mathcal{V}} P\big(\overline{\boldsymbol{a}}_v^0|\overline{\boldsymbol{a}}_v\big)\cdot\prod_{e=(v,v')\in\mathcal{E}} P\big(\overline{\boldsymbol{a}}_e^0|\overline{\boldsymbol{a}}_v,\overline{\boldsymbol{a}}_{v'}\big).$$

Each unary term $P\big(\overline{\boldsymbol{a}}_v^0|\overline{\boldsymbol{a}}_v\big)$ measures the closeness between the prediction $\overline{\boldsymbol{a}}_v^0$ and the corresponding recovery $\overline{\boldsymbol{a}}_v$. We model the variance and employ a robust norm to handle outliers:

$$P\big(\overline{\boldsymbol{a}}_v^0|\overline{\boldsymbol{a}}_v\big) \sim \exp(-\frac{1}{2}\rho(\|\overline{\boldsymbol{a}}_v^0-\overline{\boldsymbol{a}}_v\|_{\Sigma_{c_v}^{-1}},\alpha_{c_v})) \quad (2)$$

where $\rho(x,\alpha)=x^2/(x^2+\alpha)$ is the Geman-McClure robust function [3]; $\|\boldsymbol{x}\|_A=\boldsymbol{x}^TA\boldsymbol{x}$; α_{c_v} and the covariance matrix $\Sigma_{c_v}\succ 0$ are hyperparameters of class c_v.

We use a similar formulation to model the pariwise term associated with each edge $e=(v,v')\in\mathcal{E}$:

$$P\big(\overline{\boldsymbol{a}}_e^0|\overline{\boldsymbol{a}}_v,\overline{\boldsymbol{a}}_{v'}\big) \sim \exp(-\frac{1}{2}\rho(\|\overline{\boldsymbol{a}}_e^0-\phi(\overline{\boldsymbol{a}}_v,\overline{\boldsymbol{a}}_{v'})\|_{\Sigma_{c_e}^{-1}},\alpha_{c_e})) \quad (3)$$

where $c_e=(c_v,c_{v'})$ denotes the class label of edge e; $\Sigma_{c_e}\succ 0$ and α_{c_e} are hyperparameters of class c_e.

Combing (2) and (3), we arrive at the following formulation for the total likelihood term $P\big(\{\overline{\boldsymbol{a}}_v^0\}\cup\{\overline{\boldsymbol{a}}_e^0\}|\{\overline{\boldsymbol{a}}_v\}\big)$

$$\sim\exp\Big(-\frac{1}{2}\sum_{v\in\mathcal{V}}\rho(\|\overline{\boldsymbol{a}}_v^0-\overline{\boldsymbol{a}}_v\|_{\Sigma_{c_v}^{-1}},\alpha_{c_v}) -\frac{1}{2}\sum_{e=(v,v')\in\mathcal{E}}\rho(\|\overline{\boldsymbol{a}}_e^0-\phi(\overline{\boldsymbol{a}}_v,\overline{\boldsymbol{a}}_{v'})\|_{\Sigma_{c_e}^{-1}},\alpha_{c_e})\Big) \quad (4)$$

Prior modeling. We model the total prior term by decoupling attributes and indicators and by multiplying unary terms and pairwise terms: $P(\{\overline{\boldsymbol{a}}_v\})$

$$\sim\prod_{v\in\mathcal{V}}P_{c_v}(\boldsymbol{a}_v)\prod_{e=(v,v')\in\mathcal{E}}P_{c_e}(\phi(\boldsymbol{a}_v,\boldsymbol{a}_{v'}))P(\{z_v\}) \quad (5)$$

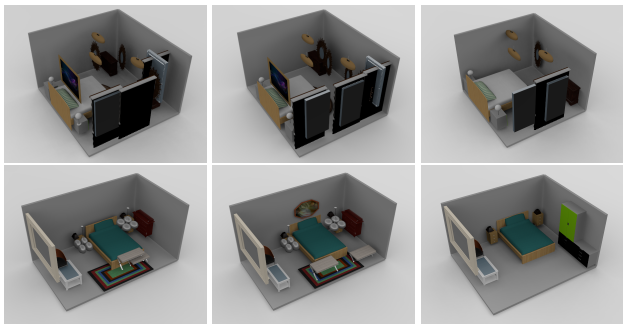


Figure 3: Left: scene layout from predicted object attributes. Middle: scene layout by synchronizing predicted object attributes and relative attributes, i.e., only likelihood terms are used. Right: the output of scene optimization that combines both likelihood terms and prior terms.

where $P_{c_v}(\boldsymbol{a}_v)$ models the attribute prior of the vertex class c_v;$P_{c_e}(\phi(\boldsymbol{a}_v,\boldsymbol{a}_{v'}))$ models the relative attribute prior of the edge class c_e; $P(\{z_v\})$ denotes the object count prior.

We use generalized Gaussian mixture models (or GGMMs) to model P_c and $P_{(c,c')}$:

$$P_c(\boldsymbol{a}_v)=M_{\mu_c}(\boldsymbol{a}_v), \quad (6)$$

$$P_{(c,c')}(\phi(\boldsymbol{a}_v,\boldsymbol{a}_{v'}))=M_{\mu_{(c,c')}}(\phi(\boldsymbol{a}_v,\boldsymbol{a}_{v'})) \quad (7)$$

where μ_c and $\mu_{(c,c')}$ denote hyperparameters of the mixture models. By GGMMs, we mean each mixture component is associated with an optimal mask to model the self-penetration free constraint between pairs of objects. Due to space constraints, we defer details of GGMMs and visualizations of the resulting GGMMs to the supplementary material. Note that our approach learns μ_c and $\mu_{(c,c')}$ from the training data and refines all the hyperparameters jointly to maximize the output of our approach. The joint optimization procedure is explained in Section 4.4.

The prior term $P(\{z_v\})$ models object counts and object co-occurrences. Similar to the likelihood term, we model $P(\{z_v\})$ as a combination of unary and pairwise terms:

$$P(\{z_v\})\sim\prod_{c\in\mathcal{C}}P_c(\boldsymbol{z}_{\mathcal{V}_c})\prod_{c,c'\in\mathcal{C}}P_{(c,c')}(\boldsymbol{z}_{\mathcal{V}_c},\boldsymbol{z}_{\mathcal{V}_{c'}}). \quad (8)$$

where $\boldsymbol{z}_{\mathcal{V}_c}$ collects indicators of vertices that belong to the vertex class c. We again model both P_c and $P_{(c,c')}$ using 1D and 2D GGMMs:

$$P_c(\boldsymbol{z}_{\mathcal{V}_c})=M_{\gamma_c}(\boldsymbol{1}^T\boldsymbol{z}_{\mathcal{V}_c}) \quad (9)$$

$$P_{(c,c')}(\boldsymbol{z}_{\mathcal{V}_c},\boldsymbol{z}_{\mathcal{V}_{c'}})=M_{\gamma_{(c,c')}}\big((\boldsymbol{1}^T\boldsymbol{z}_{\mathcal{V}_c},\boldsymbol{1}^T\boldsymbol{z}_{\mathcal{V}_{c'}})\big) \quad (10)$$

Note that we again initialize γ_c and $\gamma_{(c,c')}$ from data and refine them and other hyperparameters jointly. Please refer to the supplementary material for visualizations of the resulting GGMMs.

Substituting (9) and (10) into (8) and combing (6) and (7), we arrive at the following prior model:

$$P(\{\overline{\boldsymbol{a}}_v\}) \sim \prod_{v\in\mathcal{V}} M_{\mu_{c_v}}(\boldsymbol{a}_v) \prod_{e=(v,v')\in\mathcal{E}} M_{\mu_{c_e}}(\phi(\boldsymbol{a}_v,\boldsymbol{a}_{v'}))$$
$$\prod_{c\in\mathcal{C}} M_{\gamma_c}(\mathbf{1}^T\boldsymbol{z}_{\mathcal{V}_c}) \prod_{c,c'\in\mathcal{C}} M_{\gamma_{(c,c')}}\big((\mathbf{1}^T\boldsymbol{z}_{\mathcal{V}_c},\mathbf{1}^T\boldsymbol{z}_{\mathcal{V}_{c'}})\big) \quad (11)$$

4.3. Scene Optimization

Our goal is to find $\overline{\boldsymbol{a}}_v, v\in\mathcal{V}$ that maximize the posterior distribution defined in (1). The variables consist of primitive parameters and primitive indicators. Our optimization strategy relaxes the indicator variables as real variables $z_v\in\mathcal{R}$. It then performs alternating optimization to refine these two categories of variables. This relaxation strategy not only makes the optimization problem easy to solve but also facilitates hyperparameter learning (See Section 4.4).

Specifically, when the indicator variables $z_v, v\in\mathcal{V}$ are fixed, the optimization problem reduces to (we minimize the negation of the log-posterior)

$$\min_{\{\boldsymbol{a}_v\}} \sum_{v\in\mathcal{V}} \Big(\frac{1}{2}\rho\big(\|\overline{\boldsymbol{a}}_v^0-\overline{\boldsymbol{a}}_v\|_{\Sigma_{c_v}^{-1}},\alpha_{c_v}\big) - \log\big(M_{\mu_c}(\boldsymbol{a}_v)\big)\Big)$$
$$+ \sum_{e=(v,v')\in\mathcal{E}} \Big(\frac{1}{2}\rho\big(\|\overline{\boldsymbol{a}}_e^0-\phi(\overline{\boldsymbol{a}}_v,\overline{\boldsymbol{a}}_{v'})\|_{\Sigma_{c_e}^{-1}},\alpha_{c_e}\big)$$
$$- \log\big(M_{\mu_{(c,c')}}(\phi(\boldsymbol{a}_v,\boldsymbol{a}_{v'}))\big)\Big) \quad (12)$$

The objective function in (12) is continuous in $\boldsymbol{a}_v$. We employ the limited-memory Broyden-Fletcher-Goldfarb-Shanno (or L-BFGS) algorithm for optimization. The initial solution is first set as $\boldsymbol{a}_v^0$ and then uses the output of the previous iteration.

When $\boldsymbol{a}_v, v\in\mathcal{V}$ are fixed, we solve

$$\min_{\{z_v\}} \frac{1}{2}\sum_{v\in\mathcal{V}} \rho\big(\|\overline{\boldsymbol{a}}_v^0-\overline{\boldsymbol{a}}_v\|_{\Sigma_{c_v}^{-1}},\alpha_{c_v}\big) - \sum_{c\in\mathcal{C}} \log\big(M_{\gamma_c}(\mathbf{1}^T\boldsymbol{z}_{\mathcal{V}_c})\big)$$
$$+ \frac{1}{2}\sum_{e=(v,v')\in\mathcal{E}} \rho\big(\|\overline{\boldsymbol{a}}_e^0-\boldsymbol{h}(\overline{\boldsymbol{a}}_v,\overline{\boldsymbol{a}}_{v'})\|_{\Sigma_{c_e}^{-1}},\alpha_{c_e}\big)$$
$$- \sum_{c,c'\in\mathcal{C}} \log\big(M_{\gamma_{(c,c')}}(\mathbf{1}^T\boldsymbol{z}_{\mathcal{V}_c},\mathbf{1}^T\boldsymbol{z}_{\mathcal{V}_{c'}})\big) \quad (13)$$

We employ the same strategy as (12) for optimization. Our experiments suggest that 20-30 alternating iterations are sufficient. Figure 3 shows typical examples, where relative attributes and prior terms provide effective regularizations for object attributes.

4.4. Joint Hyperparameter Learning

In this section, we present an approach that learns hyperparameters of the scene optimization formulation described above. Specifically, to make the notations uncluttered, let

$$\Phi = \{\Sigma_c,\alpha_c,\mu_c,\gamma_c\}\cup\{\Sigma_{(c,c')},\alpha_{(c,c')},\mu_{(c,c')},\gamma_{(c,c')}\}$$

collect all the hyperparameters. Let $\boldsymbol{x}$ and $\boldsymbol{y}$ denote the inputs $\{\overline{\boldsymbol{a}}_v^0\}\cup\{\overline{\boldsymbol{a}}_e^0\}$ and the optimal solution to $\{\overline{\boldsymbol{a}}_v\}$ to (1). Finally, we denote the objective function in (1) as $f(\Phi,\boldsymbol{x},\boldsymbol{y})$.

Our goal is to train Φ using a validation set $\mathcal{T}_{val} := \{(\boldsymbol{x}_i,\boldsymbol{y}_i^{\text{gt}})\}$ and a regularization loss $l(\Phi)$. Each $(\boldsymbol{x}_i,\boldsymbol{y}_i^{\text{gt}})$ is computed by feeding $\boldsymbol{y}_i^{\text{gt}}$ as the input to the encoder modules and setting $\boldsymbol{x}_i$ as outputs of the decoder modules. The regularization term $l(\Phi)$ combines all the loss terms that learn hyperparameters of the prior distributions, i.e., (6), (7), (9), and (10). We defer the explicit expression of $l(\Phi)$ to the supplementary material.

The performance of scene optimization depends on whether the ground-truth solution is a local minimum and whether the prediction modules' initial solution reaches this local minimum through optimization. We introduce a novel formulation that only involves function values of f to enforce these two constraints:

$$\min_{\Phi} \quad l(\Phi) + \sum_{(\boldsymbol{x}_i,\boldsymbol{y}_i^{\text{gt}})\in\mathcal{T}_{val}} \underset{\boldsymbol{y}_i\sim\mathcal{N}(\boldsymbol{y}_i^{\text{gt}},r_m I)}{E}$$
$$\Big(\max\big(f(\Phi,\boldsymbol{x}_i,\boldsymbol{y}_i^{\text{gt}}) - f(\Phi,\boldsymbol{x}_i,\boldsymbol{y}_i)+\delta,0\big) \quad (14)$$
$$+ \lambda_s \underset{\boldsymbol{y}_i'\sim\mathcal{N}(\boldsymbol{y}_i,r_s I)}{E} \big(f(\Phi,\boldsymbol{x}_i,\boldsymbol{y}_i) - f(\Phi,\boldsymbol{x}_i,\boldsymbol{y}_i')\big)^2\Big) \quad (15)$$

where $\mathcal{N}(\boldsymbol{y},rI)$ is the normal distribution with mean $\boldsymbol{y}$ and variance rI. Specifically, (14) forces the ground-truth solution to be a local minimum. (15) prioritizes that the loss surface of f is smooth, and therefore the local minimum has a large convergence radius. We determine the hyperparameters r_m, r_s, δ, and λ_s via cross-validation to minimize the L2 distances between the optimized object attributes and the ground-truth object attributes on the validation set $\mathcal{T}_{val}$.

5. Attribute Encoding and Synthesis

This section describes the details of predicting initial object attributes and relative attributes. In Section 5.1, we present the encoding of object attributes and the corresponding network architecture. Section 5.2 then presents the encoding of relative attributes and the corresponding network architecture. Finally, we present the training procedure for the neural models described above in Section 5.3.

5.1. Object Attributes

We use a similar approach as [42, 56] to encode a 3D scene as a collection of object attributes $\boldsymbol{a}_v$ and object indicators z_v. Each object attribute $\boldsymbol{a}_v$ is encoded as a vector in $\mathcal{R}^{12}$. The elements of $\boldsymbol{a}_v$ include size parameters $\boldsymbol{s}_v\in\mathcal{R}^3$, orientation parameters $\boldsymbol{r}_v\in\mathcal{R}^3$, location parameters $\boldsymbol{t}_v\in\mathcal{R}^3$, and shape codes $\boldsymbol{d}_v\in\mathcal{R}^3$. The size parameters $\boldsymbol{s}_v=(s_v^x,s_v^y,s_v^z)^T$ encode the scalings of v aligned with the axis of the coordinate system associated with each object

v. $\boldsymbol{r}_v = (\theta_v^x, \theta_v^y, \theta_v^z)$ collects the Euler angles that specify the orientation (i.e., a rotation) of v in the world coordinate system. $\boldsymbol{t}_v$ specifies the location of v in the world coordinate system. Finally, $\boldsymbol{d}_v$ is obtained in two steps. The first step uses the pre-trained model [49] to obtain a latent code for each object's shape. The second step then performs PCA among latent codes of all the objects in training set to obtain the coordinates of the top-3 principal vectors. During testing, we use the synthesized code to search for the closest object in the training set.

Let N_c be the maximum number of objects of each object class $c \in \mathcal{C}$. We parameterize a 3D scene using a matrix $\overline{A}_{\mathcal{C}} \in \mathcal{R}^{13\times(\sum_{c\in\mathcal{C}} N_c)}$, where the columns of $\overline{A}_{\mathcal{C}}$ are $\overline{\boldsymbol{a}}_v = (\boldsymbol{a}_v, z_v)^T$ of the corresponding objects.

We adopt the variational auto-encoder (or VAE) architecture in [56]. The network design utilizes sparsely connected layers to alleviate the overfitting issue. As shown in Figure 2, we will sample the latent space of this VAE to synthesis 3D scenes. Specifically, the decoder of this VAE synthesizes object attributes. As we will discuss shortly, the output is then fed into another network to output relative attributes. The object attributes are optimized by feeding the the neural outputs into the scene optimization framework described in Section 4.

5.2. Relative Attributes

Unlike object attributes, relative attributes are forced to capture patterns between pairs of objects, e.g., adjacent objects. Like object attributes, we encode the relative attributes $\overline{\boldsymbol{a}}_e$ of each edge $e = (v, v')$ as $\overline{\boldsymbol{a}}_e = (\boldsymbol{s}_e; \boldsymbol{r}_e; \boldsymbol{t}_e)$. Here $\boldsymbol{s}_e \in \mathcal{R}^9$ denotes the pairwise differences between the three scales of $\boldsymbol{s}_v$ and those of $\boldsymbol{s}_{v'}$. $\boldsymbol{r}_e \in R^3$ denotes the Euler angles of v''s pose in the local coordinate system of v. $\boldsymbol{t}_e \in R^3$ denotes the center of v' in the local coordinate system of v. The entire set of relative primitive parameters is encoded using a tensor $\overline{A}_{\mathcal{E}} \in \mathcal{R}^{\sum_{c\in\mathcal{C}} N_c \times \sum_{c\in\mathcal{C}} N_c \times 15}$.

As shown in Figure 2, the network architecture of this module mimics the U-Net [35]. It is conceptually similar to an auto-encoder with two major differences. First, we do not sample the code space to synthesize relative attributes. Second, its input consists of relative attributes induced from predicted object attributes, which are expected to be noisy. The role of this U-Net is to produce rectified relative attributes. Please refer to the supplementary material for details.

5.3. Network Training

Training attribute synthesis modules extends the standard approach for training VAEs (c.f. [25]). Let h^{ϕ} and $g_1^{\theta_1}$ be the encoder and decoder components of the VAE module for synthesizing object attributes. With $g_2^{\theta_2}$ we denote the U-Net module for synthesizing relative attributes. Here ϕ and $\theta = (\theta_1, \theta_2)$ denote the network parameters. Consider a training set $\mathcal{T} = \{(\mathcal{A}_{\mathcal{V}}, \mathcal{A}_{\mathcal{E}})\}$ where $\mathcal{A}_{\mathcal{V}}$ and $\mathcal{A}_{\mathcal{E}}$ denote encoded object attributes and relative attributes, respectively. We solve the following optimization problem to determine the optimized network weights ϕ and θ:

$$\min_{\phi,\theta} \frac{1}{|\mathcal{T}|} \sum_{(\mathcal{A}_{\mathcal{V}},\mathcal{A}_{\mathcal{E}})\in\mathcal{T}} \Big(\lambda_{\mathcal{E}} \| g_2^{\theta_2}(g_1^{\theta_1}(h^{\phi}(\overline{A}_{\mathcal{V}}))) - \overline{A}_{\mathcal{E}} \|^2 + \| g_1^{\theta_1}(h^{\phi}(\overline{A}_{\mathcal{V}})) - \overline{A}_{\mathcal{V}} \|^2 \Big) + \lambda_{KL} KL\big(\{h^{\phi}(\overline{A}_{\mathcal{V}})\} | \mathcal{N}_d\big)$$

where $\mathcal{N}_d$ is the normal distribution associated with the latent space of the object attribute VAE. The same as [25], the last term forces the latent codes of the training instances to match $\mathcal{N}_d$. This paper sets $\lambda_{\mathcal{E}} = 1$ and $\lambda_{KL} = 0.01$. We use ADAM [24] for network training.

6. Results

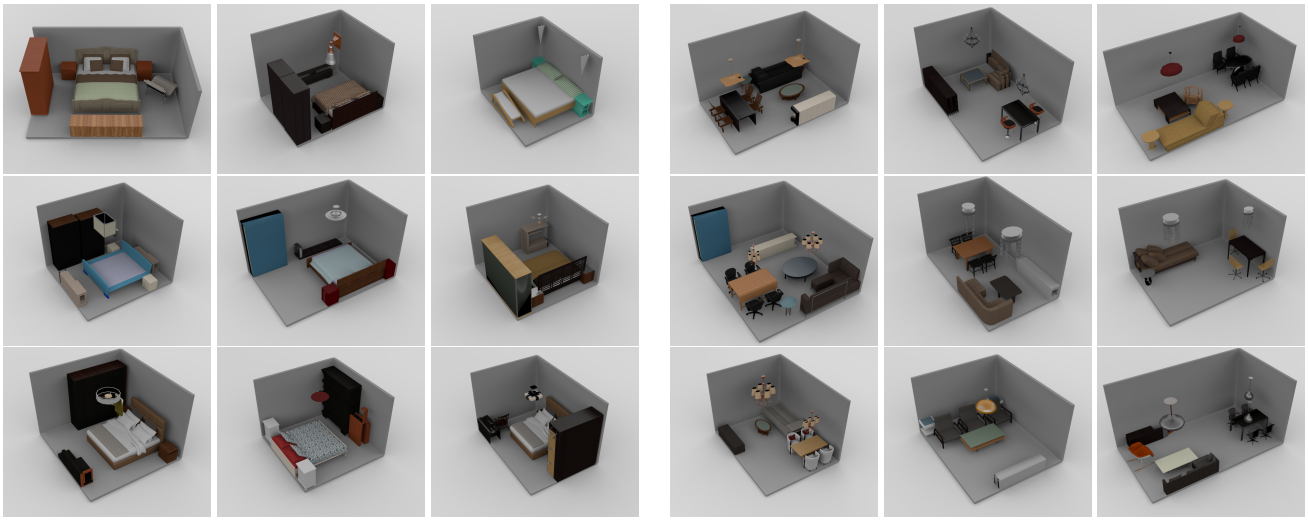
This section presents an experimental evaluation of our approach. In Section 6.1, we describe the experimental setup. In Section 6.2, we demonstrate the results of our approach and compare it with baseline methods. We analyze each component of our approach in Section 6.3. Please refer to the supplementary material for more results and baseline comparisons.

6.1. Experimental Setup

Dataset. We perform experimental evaluation on the new large-scale 3D scene dataset 3D-FRONT [12]. We also include the results on SUNCG [41], on which most works have evaluated. Following [56], we consider bedrooms and living rooms in both datasets and train scene synthesis models from each room type in each dataset. For all datasets, each room type contains 4000 training scenes and the maximum number of objects per class is four. Each room type contains 30 object classses for the SUNCG dataset and 20 object classes for the 3D-FRONT dataset. More details of the datasets are in the supplementary material.

Baseline approaches. We consider five baseline approaches D-Prior [46], Fast [34], PlanIT [45], GRAINS [28], and D-Gen [56]. They represent state-of-the-art in 3D scene synthesis.

Evaluation metrics. We consider two ways to evaluate scene synthesis approaches. The first is a perceptual study, where we employed 10 non-experts to judge the visual quality of 100 synthesized 3D scenes. We compare our results with those of each method and count the percentage of our results that are more plausible than each baseline. The second metric assesses distributions of relative attributes of the generated scenes (c.f [56]).



(a) 3D-FRONT Bedroom (b) 3D-FRONT Livingroom

Figure 4: Randomly generated scenes using our method on 3D-FRONT [12].

6.2. Analysis of the Results

Figure 4 shows randomly generated scenes using our approach. We can see that the generated scenes contain rich sets of objects, meaning our approach can generate complex scenes. The scene layouts are highly plausible from multiple perspectives, including the shape of each object, the spatial relations among multiple objects, and object co-occurrence. Moreover, the generated scenes are diverse. Figure 1 shows that our generated scenes are different from the closest scenes in the training set. These results show that our approach captures diverse feature patterns of scenes and exhibits strong generalization ability.

Table 1 provides a perceptual study between our approach and baseline approaches. We can see that our approach outperforms all baseline approaches considerably (the top performing baseline [45] utilizes additional inputs). The improvements are consistent across all the categories. Moreover, our approach is even competitive against the visual quality of the 3D scenes in the training set. These statistics demonstrate the superior performance of our approach. Please refer to the supplementary material for visual comparisons between our approach and baseline approaches.

Currently, synthesizing one scene takes 2∼5 seconds on a desktop with a 3.2G Hz CPU, 32G main memory, and a 2080 Ti GPU. The majority of the computation is spent on scene optimization, which runs on the CPU.

6.3. Analysis of Our Approach

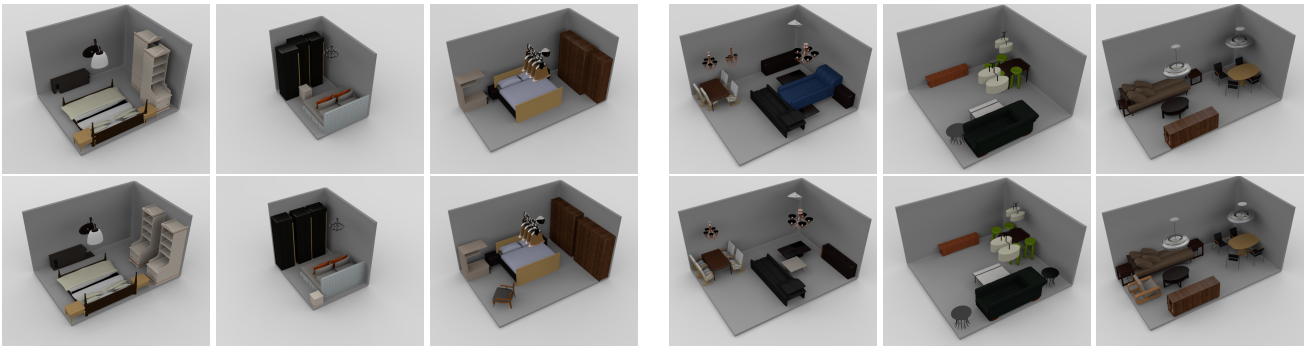
We proceed to analyze the benefits of utilizing relative attributions and prior distributions. As shown in Figure 3, using relative attributes can alleviate the issue of conflicting object attributes such as the relative poses between nightstands and beds. The optimized object locations are more plausible than those from the synthesized object attributes. However, it does not fully address issues such as penetrating objects and redundant objects. Imposing the prior distribution improves the object layout considerably, leading to penetration-free and less-crowded scene layouts.

	Ours vs. other methods					
Room Type	D-Prior	Fast	GRAINS	PlanIT	D-Gen	GT
SUN-bed	56.7±6.3	53.6±5.8	65.4±4.1	52.5±6.8	58.8±4.6	43.8±5.2
SUN-living	55.2±4.9	52.1±4.9	88.3±5.2	51.1±4.7	57.1±5.1	44.2±4.8
3DF-bed	58.1±4.3	56.7±5.9	60.1±6.3	54.4±5.3	54.5±6.3	49.7±5.3
3DF-living	72.8±6.4	73.1±4.4	89.2±5.1	68.9±6.6	64.3±5.9	47.1±4.5

Table 1: Percentage (± standard error) of forced-choice comparisons in which scenes generated by our method are judged as more plausible than scenes from another source. Higher is better. Our approach consistently outperforms baseline approaches.

Figure 5 shows additional visual comparisons between the synthesized attributes and the output of scene optimization. Again, we can see that the improvements are multi-faceted. Our approach improves the objects' locations and adds and deletes objects properly to exhibit more plausible object co-occurrences. Such improvements are justified in Figure 6, which compares the distributions of relative attributes induced from synthesized object attributions and scene optimization counterparts. We can see that those obtained from scene optimization match the underlying ground truth more closely than those induced from the synthesized object attributes. Such improvements come from relative attributes and modeling both the continuous prior distributions such as relative poses and discrete prior distributions such as joint distributions of object count pairs.

In the supplementary material, we provide results to show that even with the scene optimization framework,



(a) 3D-FRONT Bedroom (b) 3D-FRONT Livingroom

Figure 5: Visual comparisons between output of the prediction module (top) and output of the synchronization module (bottom).

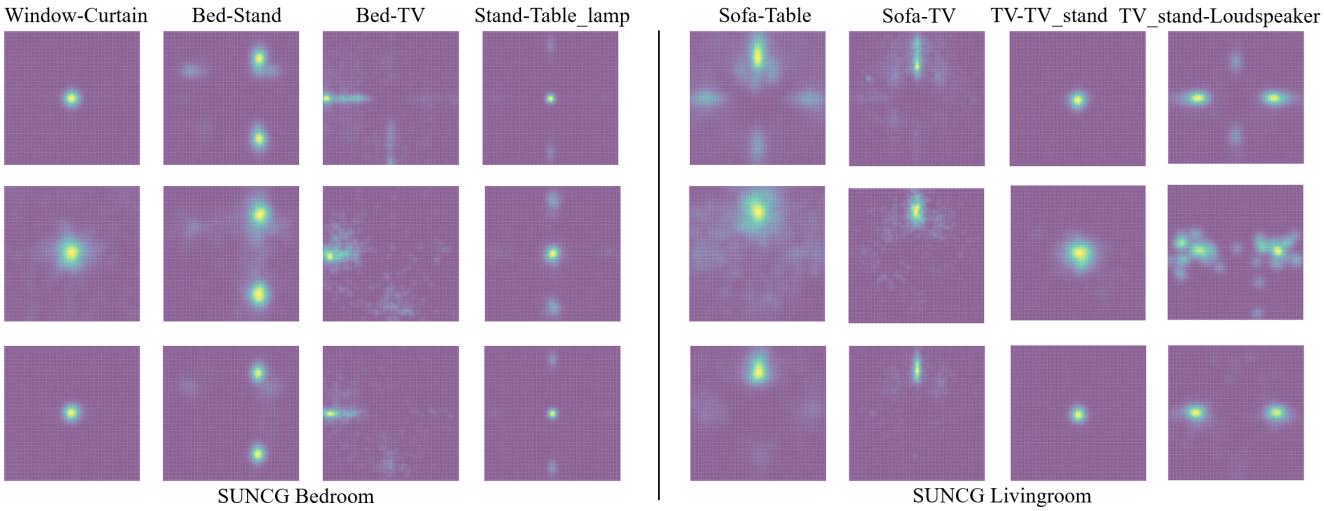


Figure 6: Distributions of relative translation. Top: distribution of the training data. Middle: distribution derived from predicted absolute parameters. Bottom: distribution derived from the optimized absolute parameters after synchronization.

synthesizing relative attributes is critical. This is because the prior distributions have rich local minimums. Utilizing the relative attributes enables us to obtain better initial solutions for scene optimization.

Scene optimization can change the scene layout considerably, e.g., adding/removing objects. The supplementary material also shows that it is challenging to achieve similar results using off-the-shelf scene optimization techniques, e.g., [52]. We can understand this as relative attributes, which are unavailable in other techniques, serve as a critical information source for our approach.

7. Conclusions and Limitations

We have shown that predicting relative attributes and learning prior distributions provide effective regularizations for synthesizing 3D scenes. The generated results preserve diverse feature patterns of 3D scenes and exhibit strong generalization behavior. Experimental results show that our approach outperforms prior state-of-the-art scene synthesis techniques.

Our approach has a couple of limitations. First, unlike end-to-end synthesis, our approach employs an optimization module to generate the final output. Therefore, computing the derivatives between the final output and the latent parameter (e.g., using the implicit function theorem) becomes more complex than end-to-end networks. One way to address this issue is to realize scene optimization using a graph neural network. Another limitation of our approach is that it does not enforce symmetries among objects, which are available in indoor scenes. An interesting question is how to detect and enforce symmetries during scene optimization automatically.

Acknowledgements. Chandrajit Bajaj would like to acknowledge the support from NIH-R01GM117594, by the Peter O'Donnell Foundation, and in part from a grant from the Army Research Office accomplished under Cooperative Agreement Number W911NF-19-2-0333. Qixing Huang would like to acknowledge the support from NSF Career IIS-2047677 and NSF HDR TRIPODS-1934932.

References

[1] Angular synchronization by eigenvectors and semidefinite programming. *Applied and Computational Harmonic Analysis*, 30:20–36, 2011 2011. 2

[2] Chandrajit Bajaj, Tingran Gao, Zihang He, Qixing Huang, and Zhenxiao Liang. SMAC: simultaneous mapping and clustering using spectral decompositions. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 334–343, 2018. 2

[3] Jonathan T. Barron. A general and adaptive robust loss function. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 4331–4339. Computer Vision Foundation / IEEE, 2019. 4

[4] Avishek Chatterjee and Venu Madhav Govindu. Efficient and robust large-scale rotation averaging. In *ICCV*, pages 521–528. IEEE Computer Society, 2013. 2

[5] Siddhartha Chaudhuri, Evangelos Kalogerakis, Stephen Giguere, and Thomas Funkhouser. Attribit: Content creation with semantic attributes. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, pages 193–202, New York, NY, USA, 2013. ACM. 2

[6] Siddhartha Chaudhuri, Evangelos Kalogerakis, Leonidas Guibas, and Vladlen Koltun. Probabilistic reasoning for assembly-based 3d modeling. *ACM Trans. Graph.*, 30(4):35:1–35:10, July 2011. 2

[7] Siddhartha Chaudhuri and Vladlen Koltun. Data-driven suggestions for creativity support in 3d modeling. In *ACM SIGGRAPH Asia 2010 Papers*, SIGGRAPH ASIA '10, pages 183:1–183:10, New York, NY, USA, 2010. ACM. 2

[8] Siddhartha Chaudhuri, Daniel Ritchie, Jiajun Wu, Kai Xu, and Hao Zhang. Learning generative models of 3d structures. *Computer Graphics Forum*, 39(2):643–666, 2020. 2

[9] Kang Chen, Yu-Kun Lai, Yu-Xin Wu, Ralph Martin, and Shi-Min Hu. Automatic semantic modeling of indoor scenes from low-quality rgb-d data using contextual information. *ACM Trans. Graph.*, 33(6):208:1–208:12, Nov. 2014. 2

[10] Yuxin Chen, Leonidas J Guibas, and Qi-Xing Huang. Near-optimal joint object matching via convex relaxation. *International Conference on Machine Learning (ICML)*, pages 100 – 108, 2014. 2

[11] Matthew Fisher, Daniel Ritchie, Manolis Savva, Thomas Funkhouser, and Pat Hanrahan. Example-based synthesis of 3d object arrangements. *ACM Trans. Graph.*, 31(6):135:1–135:11, Nov. 2012. 2

[12] Huan Fu, Bowen Cai, Lin Gao, Lingxiao Zhang, Cao Li, Zengqi Xun, Chengyue Sun, Yiyun Fei, Yu Zheng, Ying Li, Yi Liu, Peng Liu, Lin Ma, Le Weng, Xiaohang Hu, Xin Ma, Qian Qian, Rongfei Jia, Binqiang Zhao, and Hao Zhang. 3d-front: 3d furnished rooms with layouts and semantics. *CoRR*, abs/2011.09127, 2020. 2, 6, 7

[13] Thomas Funkhouser, Michael Kazhdan, Philip Shilane, Patrick Min, William Kiefer, Ayellet Tal, Szymon Rusinkiewicz, and David Dobkin. Modeling by example. *ACM Trans. Graph.*, 23(3):652–663, Aug. 2004. 2

[14] Mario Geiger, Arthur Jacot, Stefano Spigler, Franck Gabriel, Levent Sagun, Stéphane d'Ascoli, Giulio Biroli, Clément Hongler, and Matthieu Wyart. Scaling description of generalization with number of parameters in deep learning. *CoRR*, abs/1901.01608, 2019. 2

[15] Leonidas J. Guibas, Qixing Huang, and Zhenxiao Liang. A condition number for joint optimization of cycle-consistent networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 1005–1015, 2019. 2

[16] David Ha and Douglas Eck. A neural representation of sketch drawings. *CoRR*, abs/1704.03477, 2017. 2

[17] Qixing Huang and Leonidas Guibas. Consistent shape maps via semidefinite programming. In *Proceedings of the Eleventh Eurographics/ACMSIGGRAPH Symposium on Geometry Processing*, pages 177–186, 2013. 2

[18] Qixing Huang, Hai Wang, and Vladlen Koltun. Single-view reconstruction via joint analysis of image and shape collections. *ACM Trans. Graph.*, 34(4):87:1–87:10, July 2015. 2

[19] Xiangru Huang, Zhenxiao Liang, Chandrajit Bajaj, and Qixing Huang. Translation synchronization via truncated least squares. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1459–1468. Curran Associates, Inc., 2017. 2

[20] Xiangru Huang, Zhenxiao Liang, and Qixing Huang. Uncertainty quantification for multi-scan registration. *ACM Trans. Graph.*, 39(4), July 2020. 2

[21] Xiangru Huang, Zhenxiao Liang, Xiaowei Zhou, Yao Xie, Leonidas J. Guibas, and Qixing Huang. Learning transformation synchronization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8082–8091, 2019. 2

[22] Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, and Vladlen Koltun. A probabilistic model for component-based shape synthesis. *ACM Trans. Graph.*, 31(4):55:1–55:11, July 2012. 2

[23] Z. Sadeghipour Kermani, Z. Liao, P. Tan, and H. Zhang. Learning 3d scene synthesis from annotated rgb-d images. *Comput. Graph. Forum*, 35(5):197–206, Aug. 2016. 2

[24] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 6

[25] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. 6

[26] Vladislav Kreavoy, Dan Julius, and Alla Sheffer. Model composition from interchangeable components. In *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications*, pages 129–138, Washington, DC, USA, 2007. IEEE Computer Society. 2

[27] Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. Grass: Generative recursive

autoencoders for shape structures. *ACM Trans. Graph.*, 36(4):52:1–52:14, July 2017. 2

[28] Manyi Li, Akshay Gadi Patil, Kai Xu, Siddhartha Chaudhuri, Owais Khan, Ariel Shamir, Changhe Tu, Baoquan Chen, Daniel Cohen-Or, and Hao Zhang. Grains: Generative recursive autoencoders for indoor scenes. *ACM Trans. Graph.*, 38(2), Feb. 2019. 2, 6

[29] Etai Littwin, Ben Myara, Sima Sabah, Joshua Susskind, Shuangfei Zhai, and Oren Golan. Collegial ensembles. 2020. 2

[30] Tianqiang Liu, Siddhartha Chaudhuri, Vladimir G. Kim, Qixing Huang, Niloy J. Mitra, and Thomas Funkhouser. Creating consistent scene graphs using a probabilistic grammar. *ACM Trans. Graph.*, 33(6):211:1–211:12, Nov. 2014. 2

[31] Paul Merrell, Eric Schkufza, and Vladlen Koltun. Computer-generated residential building layouts. *ACM Trans. Graph.*, 29(6):181:1–181:12, Dec. 2010. 2

[32] C. Nash and C. K. I. Williams. The shape variational autoencoder: A deep generative model of part-segmented 3d objects. *Comput. Graph. Forum*, 36(5):1–12, Aug. 2017. 2

[33] Daniel Ritchie, Anna Thomas, Pat Hanrahan, and Noah D. Goodman. Neurally-guided procedural models: Amortized inference for procedural graphics programs using neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 622–630, USA, 2016. Curran Associates Inc. 2

[34] Daniel Ritchie, Kai Wang, and Yu-an Lin. Fast and flexible indoor scene synthesis via deep convolutional generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6182–6190, 2019. 2, 6

[35] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*, 2015. 6

[36] Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. Csgnet: Neural shape parser for constructive solid geometry. *CoRR*, abs/1712.08290, 2017. 2

[37] Chao-Hui Shen, Hongbo Fu, Kang Chen, and Shi-Min Hu. Structure recovery by part assembly. *ACM Trans. Graph.*, 31(6):180:1–180:11, Nov. 2012. 2

[38] Ravid Shwartz-Ziv and Alexander A. Alemi. Information in infinite ensembles of infinitely-wide neural networks. *CoRR*, abs/1911.09189, 2019. 2

[39] Chen Song, Jiaru Song, and Qixing Huang. HybridPose: 6D Object Pose Estimation Under Hybrid Representations. In *Conference on Computer Vision and Pattern Recognition*, pages 428–437, 2020. 2

[40] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1746–1754, 2017. 2

[41] Shuran Song, Fisher Yu, Andy Zeng, Angel X. Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 6

[42] Shubham Tulsiani, Hao Su, Leonidas J. Guibas, Alexei A. Efros, and Jitendra Malik. Learning shape abstractions by assembling volumetric primitives. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 1466–1474. IEEE Computer Society, 2017. 1, 5

[43] Shubham Tulsiani, Hao Su, Leonidas J Guibas, Alexei A Efros, and Jitendra Malik. Learning shape abstractions by assembling volumetric primitives. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2635–2643, 2017. 2

[44] Shubham Tulsiani, Tinghui Zhou, Alexei A Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2626–2634, 2017. 2, 3

[45] Kai Wang, Yu-An Lin, Ben Weissmann, Manolis Savva, Angel X Chang, and Daniel Ritchie. Planit: Planning and instantiating indoor scenes with relation graph and spatial prior networks. *ACM Transactions on Graphics (TOG)*, 38(4):1–15, 2019. 2, 6, 7

[46] Kai Wang, Manolis Savva, Angel X Chang, and Daniel Ritchie. Deep convolutional priors for indoor scene synthesis. *ACM Transactions on Graphics (TOG)*, 37(4):70, 2018. 2, 6

[47] Lanhui Wang and Amit Singer. Exact and stable recovery of rotations for robust synchronization. *Information and Inference: A Journal of the IMA*, 2:145–193, December 2013. 2

[48] Peng Wang, Xiaohui Shen, Bryan C. Russell, Scott Cohen, Brian L. Price, and Alan L. Yuille. SURGE: Surface Regularized Geometry Estimation from a Single Image. In *Advances in Neural Information Processing Systems 29*, pages 172–180, 2016. 2

[49] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 82–90, 2016. 6

[50] Kai Xu, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. Fit and diverse: Set evolution for inspiring 3d shape galleries. *ACM Trans. Graph.*, 31(4):57:1–57:10, July 2012. 2

[51] Zhenpei Yang, Siming Yan, and Qixing Huang. Extreme relative pose network under hybrid representations. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2452–2461, 2020. 2

[52] Lap-Fai Yu, Sai-Kit Yeung, Chi-Keung Tang, Demetri Terzopoulos, Tony F. Chan, and Stanley J. Osher. Make it home: Automatic optimization of furniture arrangement. *ACM Trans. Graph.*, 30(4), July 2011. 8

[53] Song-Hai Zhang, Shao-Kui Zhang, Yuan Liang, and Peter Hall. A survey of 3d indoor scene synthesis. *Journal of Computer Science and Technology*, 34(3):594–608, 2019. 2

[54] Zaiwei Zhang, Zhenxiao Liang, Lemeng Wu, Xiaowei Zhou, and Qixing Huang. Path-invariant map networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11084–11094, June 2019. 2

[55] Zaiwei Zhang, Bo Sun, Haitao Yang, and Qixing Huang. H3dnet: 3d object detection using hybrid geometric primitives. *CoRR*, abs/2006.05682, 2020. 2

[56] Zaiwei Zhang, Zhenpei Yang, Chongyang Ma, Linjie Luo, Alexander Huth, Etienne Vouga, and Qixing Huang. Deep generative modeling for scene synthesis via hybrid representations. *ACM Trans. Graph.*, 39(2):17:1–17:21, 2020. 1, 2, 3, 5, 6

[57] Chuhang Zou, Ersin Yumer, Jimei Yang, Duygu Ceylan, and Derek Hoiem. 3d-prnn: Generating shape primitives with recurrent neural networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 900–909, 2017. 2