Fast and Flexible Indoor Scene Synthesis via Deep Convolutional Generative Models

Daniel Ritchie [*]   Kai Wang [*]   Yu-an Lin

Brown University

{daniel␣ritchie, kai␣wang, yu-an␣lin@brown.edu }

Bedrooms

Living Rooms

Offices

Bathrooms

Figure 1. Synthetic virtual scenes generated by our method. Our model can generate a large variety of such scenes, as well as complete partial scenes, in under two seconds per scene. This performance is enabled by a pipeline of multiple deep convolutional generative models which analyze a top-down representation of the scene.

Abstract

We present a new, fast and flexible pipeline for indoor scene synthesis that is based on deep convolutional generative models. Our method operates on a top-down image-based representation, and inserts objects iteratively into the scene by predicting their category, location, orientation and size with separate neural network modules. Our pipeline naturally supports automatic completion of partial scenes, as well as synthesis of complete scenes. Our method is significantly faster than the previous image-based method and generates result that outperforms it and other state-of-the-art deep generative scene models in terms of faithfulness to training data and perceived visual quality.

1. Introduction

People spend a large percentage of their lives indoors: in bedrooms, living rooms, offices, kitchens, and other such

[*]Equal contribution

spaces. The demand for virtual versions of these real-world spaces has never been higher. Games, virtual reality, and augmented reality experience often take place in such environments. Architects often create virtual instantiations of proposed buildings, which they visualize for customers using computer-generated renderings and walkthrough animations. People who wish to redesign their living spaces can benefit from a growing array of available online virtual interior design tools [25, 21]. Furniture design companies, such as IKEA and Wayfair, increasingly produce marketing imagery by rendering virtual scenes, as it is faster, cheaper, and more flexible to do so than to stage real-world scenes [10]. Finally, and perhaps most significantly, computer vision and robotics researchers have begun turning to virtual environments to train data-hungry models for scene understanding and autonomous navigation [2, 3, 8].

Given the recent interest in virtual indoor environments, a *generative model* of interior spaces would be valuable. Such a model would provide learning agents a strong prior over the structure and composition of 3D scenes. It could

also be used to automatically synthesize large-scale virtual training corpora for various vision and robotics tasks.

We define such a *scene synthesis* model as an algorithm which, given an empty interior space delimited by architectural geometry (floor, walls, and ceiling), decides which objects to place in that space and where to place them. Any model which solves this problem must reason about the existence and spatial relationships between objects in order to make such decisions. In computer vision, the most flexible, general-purpose mechanism available for such reasoning is convolution, especially as realized in the form of deep convolutional neural networks (CNNs) for image understanding. Recent work has attempted to perform scene synthesis using deep CNNs to construct priors over possible object placements in scenes [13]. While promising, this first attempt suffers from many limitations. It reasons locally about object placements and can struggle to globally coordinate an entire scene (e.g. failing to put a sofa into a living room scene). It does not model the size of objects, leading to problems with inappropriate object selection (e.g. an implausibly-long wardrobe which blocks a doorway). Finally, and most critically, it is extremely slow, requiring minutes to synthesize a scene due to its use of hundreds of deep CNN evaluations per scene.

We believe that image-based synthesis of scenes is promising because of the ability to perform precise, pixel-level spatial reasoning, as well as the potential to leverage existing sophisticated machinery developed for image understanding with deep CNNs. In this paper, we present a new image-based scene synthesis pipeline, based on deep convolutional generative models, that overcomes the issues of prior image-based synthesis work. Like the previous method mentioned above, it generates scenes by iteratively adding objects. However, it factorizes the step of adding each object into a different sequence of decisions which allow it (a) to reason globally about which objects to add, and (b) to model the spatial extent of objects to be added, in addition to their location and orientation. Most importantly, it is fast: two orders of magnitude faster than prior work, requiring on average under 2 seconds to synthesize a scene.

We evaluate our method by using it to generate synthetic bedrooms, living rooms, offices, and bathrooms (Figure 1). We also show how, with almost no modification to the pipeline, our method can synthesize multiple automatic completions of partial scenes using the same fast generative procedure. We compare our method to the prior image-based method, another state-of-the art deep generative model based on scene hierarchies, and scenes created by humans, in several quantitative experiments and a perceptual study. Our method performs as well or better than these prior techniques.

2. Related Work

Indoor Scene Synthesis   A considerable amount of effort has been devoted to studying indoor scene synthesis. Some of the earliest work in this area utilizes interior design principles [19] and simple statistical relationships [31] to arrange pre-specified sets of objects. Other early work attempts fully data-driven scene synthesis [6] but is limited to small scale scenes due to the limited availability of training data and the learning methods available at the time.

With the availability of large scene datasets such as SUNCG [28], new data-driven methods have been proposed. [20] uses a directed graphical model for object selection but relies on heuristics for object layout. [23] uses a probabilistic grammar to model scenes, but also requires data about human activity in scenes (not readily available in all datasets) as well as manual annotation of important object groups. In contrast, our model uses deep convolutional generative models to generate all important object attributes—category, location, orientation and size—fully automatically.

Other recent methods have adapted deep neural networks for scene synthesis. [33] uses a Generative Adversarial Network to generate scenes in an attribute-matrix form (i.e. one column per scene object). More recently, GRAINS [16] uses recursive neural networks to encode and sample structured scene hierarchies. Most relevant to our work is [13], which also uses deep convolutional neural networks that operate on top-down image representations of scenes and synthesizes scenes by sequentially placing objects. The main difference between our method and theirs is that (1) our method samples each object attribute with a single inference step, while theirs perform hundreds of inferences, and (2) our method models the distribution over object size in addition to category, location, and orientation. Our method also uses separate modules to predict category and location, thus avoiding some of the failure cases their method exhibits.

Deep Generative Models   Deep neural networks are increasingly used to build powerful models which *generate* data distributions, in addition to analyzing them, and our model leverages this capability. Deep latent variable models, in particular variational autoencoders (VAEs) [14] and generative adversarial networks (GANs) [7], are popular for their ability to pack seemingly arbitrary data distributions into well-behaved, lower-dimensional "latent spaces." Our model uses conditional variants of these models—CVAEs [27] and CGANs [18]—to model the potentially multimodal distribution over object orientation and spatial extent. Deep neural networks have also been effectively deployed for decomposing complex distributions into a sequence of simpler ones. Such sequential or autoregressive generative models have been used for unsupervised parsing
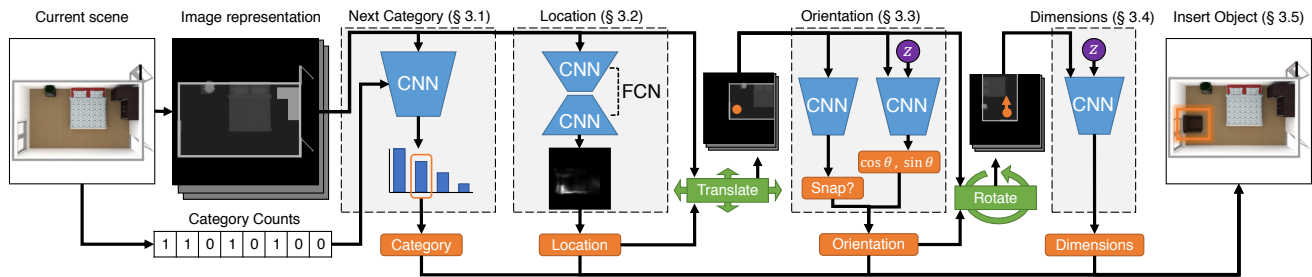
Figure 2. Overview of our automatic object-insertion pipeline. We extract a top-down-image-based representation of the scene, which is fed to four decision modules: which category of object to add (if any), the location, orientation, and dimensions of the object.

of objects in images [5], generating natural images with sequential visual attention [9], parsing images of hand-drawn diagrams [4], generating 3D objects via sequential assemblies of primitives [34], and controlling the output of procedural graphics programs [24], among other applications. We use an autoregressive model to generate indoor scenes, constructing them object by object, where each step is conditioned on the scene generated thus far.

Training Data from Virtual Indoor Scenes Virtual indoor scenes are rapidly becoming a crucial source of training data for computer vision and robotics systems. Several recent works have shown that indoor scene understanding models can be improved by training on large amounts of synthetically-generated images from virtual indoor scenes [32]. The same has been shown for indoor 3D reconstruction [2], as well as localization and mapping [17]. At the intersection of vision and robotics, researchers working on *visual navigation* often rely on virtual indoor environments to train autonomous agents for tasks such as interactive/embodied question answering [3, 8]. To support such tasks, a myriad of virtual indoor scene simulation platforms have emerged in recent years [26, 29, 1, 15, 30, 22]. Our model can complement these simulators by automatically generating new environments in which to train such intelligent visual reasoning agents.

3. Model

Our goal is to build a deep generative model of scenes that leverages precise image-based reasoning, is fast, and can flexibly generate a variety of plausible object arrangements. To maximize flexibility, we use a sequential generative model which iteratively inserts one object at a time until completion. In addition to generating complete scenes from an empty room, this paradigm naturally supports partial scene completion by simply initializing the process with a partially-populated scene. Figure 2 shows an overview of our pipeline. It first extracts a top-down, floor-plan image representation of the input scene, as done in prior work on image-based scene synthesis [13]. Then, it feeds this

representation to a sequence of four decision modules to determine how to select and add objects into the scene. These modules decide which category of object to add to the scene, if any (Section 3.1), where that object should be located (Section 3.2), what direction it should face (Section 3.3), and its physical dimensions (Section 3.4). This is a different factorization than in prior work, which we will show leads to both faster synthesis and higher-quality results. The rest of this section describes the pipeline at a high level; precise architectural details can be found in the supplemental material, and the source code for our system is available at https://github.com/brownvc/fast-synth.

3.1. Next Object Category

The goal of our pipeline's first module is, given a top down scene image representation, to predict the category of an object to add to the scene. The module needs to reason about what objects are already present, how many, and the available space in the room. To allow the model to also decide when to stop, we augment the category set with an extra "<STOP>" category. The module uses a Resnet18 [11] to encode the scene image. It also extract the counts of all categories of objects in the scene (i.e. a "bag of categories" representation), as in prior work [13], and encodes this with a fully-connected network. Finally, the model concatenates these two encodings and feeds them through another fully-connected network to output a probability distribution over categories. At test time, the module samples from the predicted distribution to select the next category.

Figure 3 shows some example partial scenes and the most probable next categories that our model predicts for them. Starting with an empty scene, the next-category distribution is dominated by one or two large, frequently-occurring objects (e.g. beds and wardrobes, for bedroom scenes). The probability of other categories increases as the scene begins to fill, until the scene becomes sufficiently populated and the "<STOP>" category begins to dominate.

Prior work in image-based scene synthesis predicted category and location jointly [13]. This lead to the drawback, as the authors has noted, that objects which are very likely
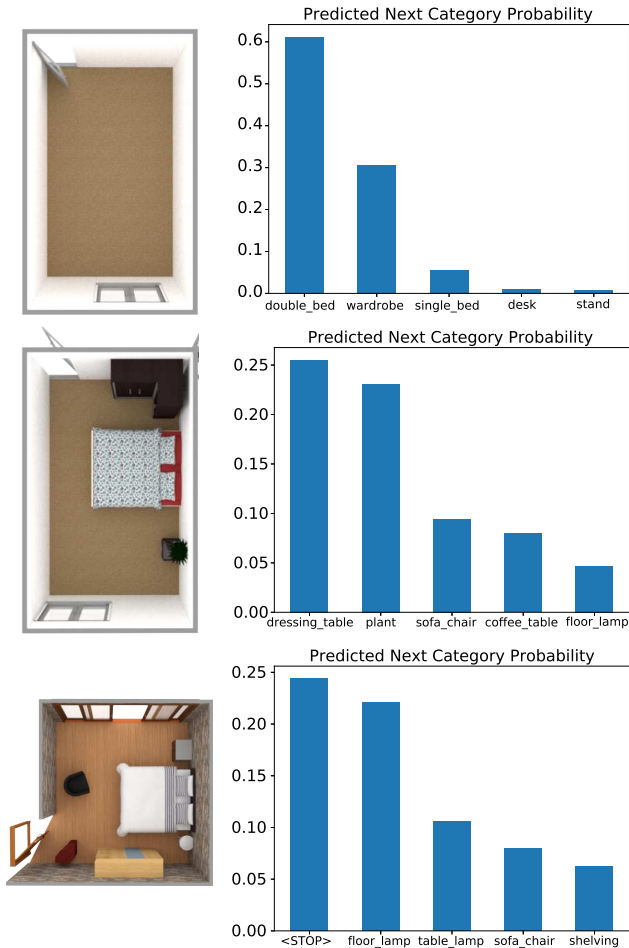
Figure 3. Distributions over the next category of object to add to the scene, as predicted by our model. Empty scenes are dominated by one or two large, frequent object types (*top*), partially populated scenes have a range of possibilities (*middle*), and very full scenes are likely to stop adding objects (*bottom*).

Double Bed   *Wardrobe*   *Nightstand*

Figure 4. Probability densities for the locations of different object types predicted by our fully-convolutional network module.

(a)   (b)   (c)   (d)

Figure 5. Probability distributions for nightstands, without (*(a)* & *(c)*) and with (*(b)* & *(d)*) regularization.

to occur in a location can be repeatedly (i.e. erroneously) sampled, e.g. placing multiple nightstands to the left of a bed. In contrast, our category prediction module reasons about the scene globally and thus avoid this problem.

3.2. Object Location

In the next module, our model takes the input scene and predicted category to determine where in the scene an instance of that category should be placed. We treat this problem as an image-to-image translation problem: given the input top-down scene image, output a 'heatmap' image containing the probability per pixel of an object occurring there. This representation is advantageous because it can be treated as a (potentially highly multimodal) 2D discrete distribution, which we can sample to produce a new location. This pixelwise discrete distribution is similar to that of prior work, except they assembled the distribution pixel-
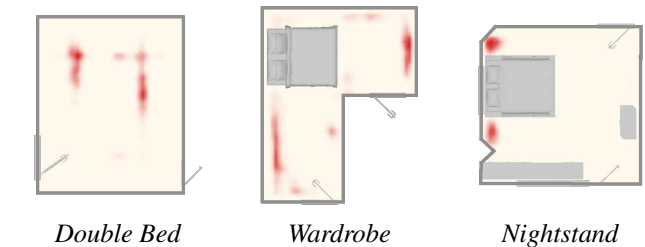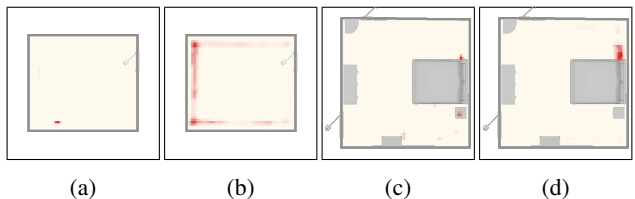
by-pixel, invoking a deep convolutional network once per pixel of the scene [13]. In contrast, our module uses a single forward pass through a fully-convolutional encoder-decoder network (FCN) to predict the entire distribution at once.

This module uses a Resnet34 encoder followed by an up-convolutional decoder. The decoder outputs a $64 \times 64 \times |C|$ image, where $|C|$ is the number of categories. The module then slices out the channel corresponding to the category of interest and treats it as a 2D probability distribution by renormalizing it. We also experimented with using separate FCNs per category that predict a $64 \times 64 \times 1$ probability density image but found it not to work as well. We suspect that training the same network to predict all categories provides the network with more context about different locations, e.g. instead of just learning that it should not predict a wardrobe at a location, it can also learn that this is because a nightstand is more likely to appear there. Before renormalization, the module zeros out any probability mass that falls outside the bounds of the room. When predicting locations for second-tier categories (e.g. table lamps), it also zeros out probability mass that falls on top of an object that was not observed as a supporting surface for that category in the dataset. At test time, we sample from a tempered version of this discrete distribution (we use temperature $\tau = 0.8$ for all experiments in this paper).

Figure 4 shows examples of predicted location distributions for different scenes. The predicted distributions for bed and wardrobe avoid placing probability mass in locations which would block the doors. The distribution for nightstand is bimodal, with each mode tightly concentrated around the head of the bed.

To train the network, we use pixel-wise cross entropy loss. As in prior work, we augment the category set with a category for "empty space," which allows the network
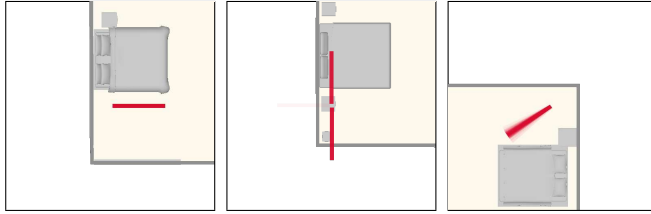
Nightstand   *Table Lamp*   *Armchair*

Figure 6. High-probability object orientations sampled by our CVAE orientation predictor (visualized as a density plot of front-facing vectors). Objects typically either snap to one orientation (*left*) or multiple orientation modes (*middle*), or have a range of values clustered around a single mode (*right*).

Double Bed   *TV Stand*   *Ottoman*

Figure 7. High-probability object dimensions sampled by our CVAE-GAN dimension predictor (visualized as a density plot of bounding boxes). Objects in more constrained locations have lower-variance size distributions (*right*).
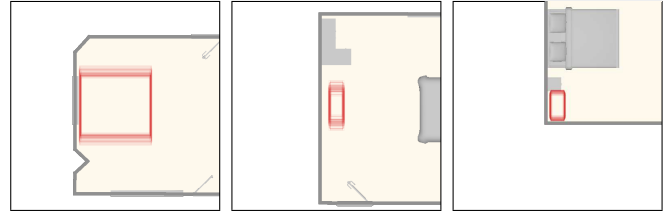
to reason about where objects should *not* be, in addition to where they should. Empty-space pixels are weighted 10 times less heavily than occupied pixels in the training loss computation. As the ground truth label for each training example is a single location instead of a distribution, our model has the potential to overfit to that exact location. This is shown in Figures 5a & 5c, where the predicted distribution collapses to single-point locations. In the second case, the network likely tries to match the input room to several memorized ones, none of which makes sense. To deal with this problem, we handicap the capacity of the network by applying L2 regularization and dropout, forcing it to learn a latent space where structurally similar scenes are close together. This results in averaged output locations, i.e. a continuous distribution of locations (Figures 5b & 5d).

Before moving on to the next module, our system translates the input scene image so that it is centered about the predicted location. This makes the subsequent modules translation-invariant.

3.3. Object Orientation

Given a translated top-down scene image and object category, the orientation module predicts what direction an object of that category should face if placed at the center of the image. We assume each category has a canonical front-facing direction. Rather than predict the angle of rotation θ, which is circular, we instead predict the front direction vector, i.e. $[\cos\theta, \sin\theta]$. This must be a normalized vector, i.e. the magnitude of $\sin\theta$ must be $\sqrt{1 - \cos^2\theta}$. Thus, our module predicts $\cos\theta$ along with a Boolean value giving the sign of $\sin\theta$. Here, we found using separate network weights per category to be most effective.

The set of possible orientations has the potential to be multimodal: for instance, a bed in the corner of a room may be backed up against either wall of the corner. To allow our module to model this behavior, we implement it with a conditional variational autoencoder (CVAE) [27]. Specifically, we use a CNN to encode the input scene, which we then concatenate with a latent code z sampled from a mul-

tivariate unit normal distribution, and then feed to a fully-connected decoder to produce $\cos\theta$ and the sign of $\sin\theta$. At training time, we use the standard CVAE loss formulation (i.e. with an extra encoder network) to learn an approximate posterior distribution over latent codes).

Since interior scenes are frequently enclosed by rectilinear architecture, objects in them are often precisely aligned to cardinal directions. A CVAE, however, being a probabilistic model, samples noisy directions. To allow our module to produce precise alignments when appropriate, this module includes a second CNN which takes the input scene and predicts whether the object to be inserted should have its predicted orientation "snapped" to the nearest of the four cardinal directions.

Figure 6 shows examples of predicted orientation distributions for different input scenes. The nightstand snaps to a single orientation, being highly constrained by its relations to the bed and wall. Table lamps are often symmetric, which leads to a predicted orientation distribution with multiple modes. An armchair to be placed in the corner of a room is most naturally oriented diagonally with respect to the corner, but some variability is possible.

Before moving on to the next module, our system rotates the input scene image by the predicted angle of rotation. This transforms the image into the local coordinate frame of the object category to be inserted, making subsequent modules rotation-invariant (in addition to already being translation-invariant).

3.4. Object Dimensions

Given a scene image transformed into the local coordinate frame of a particular object category, the dimensions module predicts the spatial extent of the object. That is, it predicts an object-space bounding box for the object to be inserted. This is also a multimodal problem, even more so than orientation (e.g. many wardrobes of varying lengths can fit against the same wall). Again, we use a CVAE for this: a CNN encodes the scene, concatenates it with z, and then uses a fully-connected decoder to produce the $[x, y]$ dimensions of the bounding box.

The human eye is very sensitive to errors in size, e.g. an object that is too large and thus penetrates the wall next to it. To help fine-tune the prediction results, we also include an adversarial loss term in the CVAE training. This loss uses a convolutional discriminator which takes the input scene concatenated channel-wise with the signed distance field (SDF) of the predicted bounding box. As with the orientation module, this module also uses separate network weights per category.

Figure 7 visualizes predicted size distributions for different object placement scenarios. The predicted distributions capture the range of possible sizes for different object categories, e.g. TV stands can have highly variable length. However, in a situation such as Figure 7 Right, where an ottoman is to be placed between the nightstand and the wall, the predicted distribution is lower-variance due to this highly constrained location.

3.5. Object Insertion

To choose a specific 3D model to insert given the predicted category, location, orientation, and size, we perform a nearest neighbor search through our dataset to find 3D models that closely fit the predicted object dimensions. When multiple likely candidate models exist, we favor ones that have frequently co-occurred in the dataset with other objects already in the room, as this slightly improves the visual style of the generated rooms (though it is far from a general solution to the problem of style-aware scene synthesis). Occasionally, the inserted object collides with existing objects in the room, or, for second-tier objects, overhangs too much over its supporting surface. In such scenarios, we choose another object of the same category. In very rare situations (less than 1%), no possible insertions exist. If this occurs, we resample a different category from the predicted category distribution and try again.

4. Data & Training

We train our model using the SUNCG dataset, a collection of over forty thousand scenes designed by users of an online interior design tool [28]. In this paper, we focus our experiments on four common room types: bedrooms, living rooms, bathrooms, and offices. We extract rooms of these types from SUNCG, performing pre-processing to filter out uncommon object types, mislabeled rooms, etc. After pre-processing, we obtained 6300 bedrooms (with 40 object categories), 1400 living rooms (35 categories), 6800 bathrooms (22 categories), and 1200 offices (36 categories). Further details about our dataset and pre-processing procedures can be found in the supplemental material.

To generate training data for all of our modules, we follow the same general procedure: take a scene from our dataset, remove some subset of objects from it, and task the module with predicting the 'next' object to be added (i.e.

one of the removed objects). This process requires an ordering of the objects in each scene. We infer static support relationships between objects (e.g. lamp supported by table) using simple geometric heuristics, and we guarantee that all supported objects come after their supporting parents in this ordering. We further guarantee that all such supported 'second-tier' objects come after all 'first-tier' objects (i.e. those supported by the floor). For the category prediction module, we further order objects based on their *importance*, which we define to be the average size of a category multiplied by its frequency of occurrence in the dataset. Doing so imposes a stable, canonical ordering on the objects in the scene; without such an ordering, we find that there are too many valid possible categories at each step, and our model struggles to build coherent scenes across multiple object insertions. For all other modules, we use a randomized ordering. Finally, for the location module, the FCN is tasked with predicting not the location of a single next object, but rather the locations of *all* missing objects removed from the training scene whose supporting surface is present in the partial scene.

We train each module in our pipeline separately for different room categories. Empirically, we find that the category module performs best after seeing $\sim 300,000$ training examples, and the location module performs best after $\sim 1,000,000$ examples. As the problems that the orientation and dimension models are solving is more local, their behavior is more stable across different epochs. In practice, with use orientation modules trained with $\sim 2,000,000$ examples and dimension modules trained with $\sim 1,000,000$ examples.

5. Results & Evaluation

Complete scene synthesis Figure 1 shows examples of complete scenes synthesized by our model, given the initial room geometry. Our model captures multiple possible object arrangement patterns for each room type: bedrooms with desks vs. those with extra seating, living rooms for conversation vs. watching television, etc.

Scene completion Figure 8 shows examples of partial scene completion, where our model takes an incomplete scene as input and suggests multiple next objects to fill the scene. Our model samples a variety of different completions for the same starting partial scene. This example also highlights our model's ability to cope with non-rectangular rooms (bottom row), one of the distinct advantages of precise pixel-level reasoning with image-based models.

Object category distribution For a scene generative model to capture the training data well, a necessary condition is that the distribution of object categories which occurs

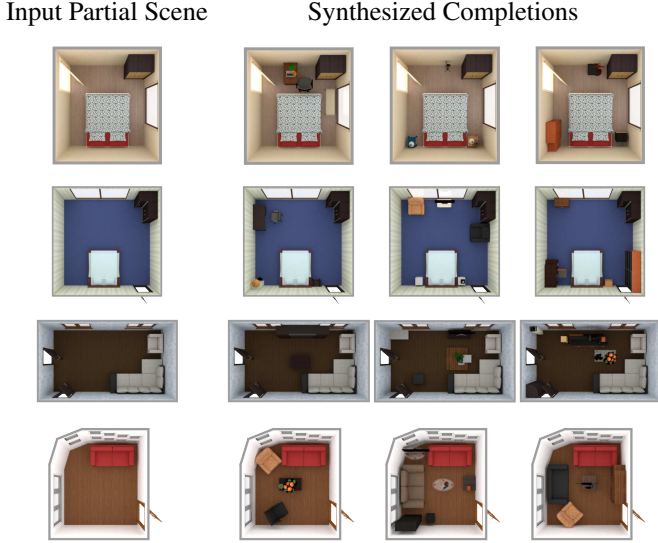Input Partial Scene   Synthesized Completions

Figure 8. Given an input partial scene (*left column*), our method can generate multiple automatic completions of the scene. This requires no modification to the method's sampling procedure, aside from seeding it with a partial scene instead of an empty one.

Method	Bedroom	Living	Bathroom	Office
Uniform	0.6202	0.8858	1.3675	0.7219
Deep Priors [13]	0.2017	0.4874	0.2479	0.2138
GRAINS [16]	0.2135	0.3217	—	—
Ours	**0.0095**	**0.0179**	**0.0240**	**0.0436**

Table 1. KL divergence between the distribution of object categories in synthesized results vs. training set. Lower is better. *Uniform* is the uniform distribution over object categories.

in its synthesized results should closely resemble that of the training set. To evaluate this, we compute the Kullback-Leibler divergence $D_{\text{KL}}(P_{\text{synth}}||P_{\text{dataset}})$ between the category distribution of synthesized scenes and that of the training set. Note that we cannot compute a symmetrized Jensen-Shannon divergence because some of the methods we compare against have zero probability for certain categories, making the divergence infinite. Table 1 shows the category distribution KL divergence of different methods. Our method generates a category distribution that are more faithful to that of the training set than other approaches.

Scene classification accuracy Looking beyond categories, to evaluate how well the distribution of our generated scenes match that of the training scenes, we train a classifier tasked to distinguish between "real" scenes (from the training set) and "synthetic" scenes (generated by our method). The classifier is a Resnet34 that takes as input the same top-down multi-channel image representation that our model uses. The classifier is trained with 1,600 scenes, half real and half synthetic. We evaluate the classifier perfor-

Method	Acc	Method	Acc
GRAINS [16]	96.56	*No Input Alignment (Orient)*	94.10
Deep Priors [13]	84.69	*No Input Alignment (Dims)*	76.60
Ours	**58.75**	*Joint Category + Location*	81.70
Perturbed (1%)	50.00	*Category from [13]*	89.30
Perturbed (5%)	54.69	*Location from [13]*	83.60
Perturbed (10%)	64.38	*Orient + Dims from [13]*	67.30

Table 2. Real vs. synthetic classification accuracy for scenes generated by different methods *(Left)* and our method, modified by changing the design of some of the components or substituting them with similar components from prior works *(Right)*. Lower (closer to 50%) is better.

Figure 9. Correcting failure cases from [13], Fig 14. *(Left)* Our model does not omit sofas for seating. *(Right)* Our model chooses a cabinet that does not block the door.

mance on 320 held out test scenes.

Table 2 shows the performance against different baselines. Compared to previous methods, our results are significantly harder for the classifier to distinguish. In fact, it is marginally *harder* to distinguish our scenes from real training scenes that it is to do so for scenes in which every object is perturbed by a small random amount (standard deviation of 10% of the object's bounding box dimensions).

Effectiveness of our design choices We use the same classification setup to investigate the effectiveness of our individual design choices. As Table 2 suggests, swapping out our model components for those of [13], omitting input alignment for the orient and dimension modules, and predicting location + category jointly all lead to worse results than the full model. We also show qualitatively in Fig 9 that our strategy help to avoid common failure cases from prior work [13]. Using a separate category module allows our model to generate seats for the living room (left), and introducing a dimension module prevents the use of a too-large cabinet that blocks the office door.

Speed comparisons Table 3 shows the time taken for different methods to synthesize a complete scene. It takes on average less than 2 seconds for our model to generate a complete scene on a NVIDIA Geforce GTX 1080Ti GPU, which is two orders of magnitudes faster than the previous image based method (Deep Priors). While slower than end-to-end methods such as [16], our model can also perform tasks such as scene completion and next object suggestion, both of which can be useful in real time applications.

Method	Avg. Time (s)
Deep Priors [13]	∼ 240
GRAINS [16]	0.1027
Ours	1.858

Table 3. Average time in seconds to generate a single scene for different methods. Lower is better.

	Ours vs.		
Room Type	GRAINS [16]	Deep Priors [13]	SUNCG
Bedroom	**82.7 ± 3.6**	**56.1 ± 4.1**	48.0 ± 4.7
Living	**74.1 ± 3.8**	52.7 ± 4.5	45.0 ± 4.5
Bathroom	—	**68.6 ± 3.9**	50.0 ± 4.5
Office	—	36.3 ± 4.5	34.8 ± 5.1

Table 4. 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. Bold indicate our scenes are preferred with > 95% confidence; gray indicates our scenes are dis-preferred with > 95% confidence; regular text indicates no preference. — indicates unavailable results.

Perceptual study We also conducted a two-alternative forced choice (2AFC) perceptual study on Amazon Mechanical Turk to evaluate how plausible our generated scenes appear compared those generated by other methods. Participants were shown two top-down rendered scene images side by side and asked to pick which one they found more plausible. Images were rendered using solid colors for each object category, to factor out any effect of material or texture appearance. For each comparison and each room type, we recruited 10 participants, which was sufficient to produce strong 95% confidence intervals. Each participant performed 55 comparisons; 5 of these were "vigilance tests" comparing against a randomly jumbled scene to check that participants were paying attention. We filter out participants who did not pass all vigilance tests.

Table 4 shows the results of this study. Our generated scenes are significantly preferred to those generated by GRAINS across all room types (GRAINS does not provide bathroom or office results). Due to format differences, our reconstruction of GRAINs room geometry is imperfect. We manually removed rooms where objects intersect with the walls, but it should be noted that the reconstructed rooms might still differ slightly from the results presented in their work. Compared to the Deep Priors method, our scenes are preferred for bedrooms and bathrooms, and judged indistinguishable for living rooms. Our generated office scenes are less preferred, however. We hypothesize that this is because the office training data is highly multimodal, containing personal offices, group offices, conference rooms, etc. It appears to us that the rooms generated by the Deep Priors method are mostly personal offices. We also generate high quality personal offices consistently. However, when the category module tries to sample other types of offices, this intent is not communicated well to other modules, resulting in unorganized results e.g. a small table with ten chairs. Finally, compared to held-out human-created scenes from SUNCG, our results are indistinguishable for bedrooms and bathrooms, nearly indistinguishable for living rooms, and again less preferred for offices.

6. Conclusion

In this paper, we presented a new pipeline for indoor scene synthesis using image-based deep convolutional generative models. Our system analyzes top-down view representations of scenes to make decisions about which objects to add to a scene, where to add them, how they should be oriented, and how large they should be. Combined, these decision modules allow for rapid (under 2 seconds) synthesis of a variety of plausible scenes, as well as automatic completion of existing partial scenes. We evaluated our method via statistics of generated scenes, the ability of a classifier to detect synthetic scenes, and the preferences of people in a forced-choice perceptual study. Our method outperforms prior techniques in all cases.

There are still many opportunities for future work in the area of automatic indoor scene synthesis. We would like to address the limitations mentioned previously in our method's ability to generate room types with multiple strong modes of variation, e.g. single offices vs. conference offices. One possible direction is to explore integrating our image-based models with models of higher-level scene structure, encoded as hierarchies a la GRAINS, or perhaps as graphs or programs. Neither our method, nor any other prior work in automatic scene synthesis of which we are aware, addresses the problem of how to generate *stylistically-consistent* indoor scenes, as would be required for interior design applications. Finally, to make automatic scene synthesis maximally useful for training autonomous agents, generative models must be aware of the functionality of indoor spaces, and must synthesize environments that support carrying out activities of interest.

References

[1] S. Brodeur, E. Perez, A. Anand, F. Golemo, L. Celotti, F. Strub, J. Rouat, H. Larochelle, and A. C. Courville. HoME: a Household Multimodal Environment. *CoRR*, arXiv:1711.11017, 2017. 3

[2] A. Dai, D. Ritchie, M. Bokeloh, S. Reed, J. Sturm, and M. Nießner. Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2018. 1, 3

[3] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra. Embodied Question Answering. In *CVPR*, 2018. 1, 3

[4] K. Ellis, D. Ritchie, A. Solar-Lezama, and J. B. Tenenbaum. Learning to Infer Graphics Programs from Hand-Drawn Images. *CoRR*, arXiv:1707.09627, 2017. 3

[5] S. M. A. Eslami, N. Heess, T. Weber, Y. Tassa, D. Szepesvari, K. Kavukcuoglu, and G. E. Hinton. Attend, Infer, Repeat: Fast Scene Understanding with Generative Models. In *NIPS 2016*, 2016. 3

[6] M. Fisher, D. Ritchie, M. Savva, T. Funkhouser, and P. Hanrahan. Example-based Synthesis of 3D Object Arrangements. In *SIGGRAPH Asia 2012*, 2012. 2

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets. In *NIPS 2014*, 2014. 2

[8] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi. IQA: Visual Question Answering in Interactive Environments. In *CVPR*, 2018. 1, 3

[9] K. Gregor, I. Danihelka, A. Graves, and D. Wierstra. DRAW: A recurrent neural network for image generation. In *ICML 2015*, 2015. 3

[10] C. Group. Putting the CGI in IKEA: How V-Ray Helps Visualize Perfect Homes. https://www.chaosgroup.com/blog/putting-the-cgi-in-ikea-how-v-ray-helps-visualize-perfect-homes, 2018. Accessed: 2018-10-13. 1

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *CVPR 2016*, 2016. 3

[12] W. Jakob. Mitsuba renderer, 2010. http://www.mitsuba-renderer.org. 8

[13] Kai Wang, Manolis Savva, Angel X. Chang, and Daniel Ritchie. Deep Convolutional Priors for Indoor Scene Synthesis. In *SIGGRAPH 2018*, 2018. 2, 3, 4, 7, 8

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

[15] E. Kolve, R. Mottaghi, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi. AI2-THOR: an interactive 3d environment for visual AI. *CoRR*, arXiv:1712.05474, 2017. 3

[16] M. Li, A. G. Patil, K. Xu, S. Chaudhuri, O. Khan, A. Shamir, C. Tu, B. Chen, D. Cohen-Or, and H. Zhang. Grains: Generative recursive autoencoders for indoor scenes. *CoRR*, arXiv:1807.09193, 2018. 2, 7, 8

[17] W. Li, S. Saeedi, J. McCormac, R. Clark, D. Tzoumanikas, Q. Ye, Y. Huang, R. Tang, and S. Leutenegger. Interiornet: Mega-scale multi-sensor photo-realistic indoor scenes dataset. In *British Machine Vision Conference (BMVC)*, 2018. 3

[18] S. O. Mehdi Mirza. Conditional generative adversarial nets. *CoRR*, arXiv:1411.1784, 2014. 2

[19] P. Merrell, E. Schkufza, Z. Li, M. Agrawala, and V. Koltun. Interactive Furniture Layout Using Interior Design Guidelines. In *SIGGRAPH 2011*, 2011. 2

[20] V. F. Paul Henderson, Kartic Subr. Automatic Generation of Constrained Furniture Layouts. *CoRR*, arXiv:1711.10939, 2018. 2

[21] Planner5d. Home Design Software and Interior Design Tool ONLINE for home and floor plans in 2D and 3D. https://planner5d.com, 2017. Accessed: 2017-10-20. 1

[22] X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba. VirtualHome: Simulating Household Activities via Programs. In *CVPR*, 2018. 3

[23] Qi, Siyuan and Zhu, Yixin and Huang, Siyuan and Jiang, Chenfanfu and Zhu, Song-Chun. Human-centric Indoor Scene Synthesis Using Stochastic Grammar. In *CVPR 2018*, 2018. 2

[24] D. Ritchie, A. Thomas, P. Hanrahan, and N. D. Goodman. Neurally-Guided Procedural Models: Amortized Inference for Procedural Graphics Programs using Neural Networks. In *NIPS 2016*, 2016. 3

[25] RoomSketcher. Visualizing Homes. http://www.roomsketcher.com. Accessed: 2017-11-06. 1

[26] M. Savva, A. X. Chang, A. Dosovitskiy, T. Funkhouser, and V. Koltun. MINOS: Multimodal indoor simulator for navigation in complex environments. *arXiv:1712.03931*, 2017. 3

[27] K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems 28*. 2015. 2, 5

[28] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic Scene Completion from a Single Depth Image. 2017. 2, 6

[29] Y. Wu, Y. Wu, G. Gkioxari, and Y. Tian. Building Generalizable Agents with a Realistic and Rich 3D Environment. *CoRR*, arXiv:1801.02209, 2018. 3

[30] C. Yan, D. K. Misra, A. Bennett, A. Walsman, Y. Bisk, and Y. Artzi. CHALET: cornell house agent learning environment. *CoRR*, arXiv:1801.07357, 2018. 3

[31] L.-F. Yu, S.-K. Yeung, C.-K. Tang, D. Terzopoulos, T. F. Chan, and S. J. Osher. Make It Home: Automatic Optimization of Furniture Arrangement. In *SIGGRAPH 2011*, 2011. 2

[32] Y. Zhang, S. Song, E. Yumer, M. Savva, J.-Y. Lee, H. Jin, and T. Funkhouser. Physically-based rendering for indoor scene understanding using convolutional neural networks. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3

[33] Z. Zhang, Z. Yang, C. Ma, L. Luo, A. Huth, E. Vouga, and Q. Huang. Deep generative modeling for scene synthesis via hybrid representations. *CoRR*, arXiv:1808.02084, 2018. 2

[34] C. Zou, E. Yumer, J. Yang, D. Ceylan, and D. Hoiem. 3D-PRNN: Generating Shape Primitives with Recurrent Neural Networks. In *ICCV 2017*, 2017. 3