End-to-end Learning for Indoor Scene Synthesis using Low-Dimensional Object Representations

Luis Reyes
ge45dej@mytum.de

Alexander Fuchs
fuchsale@in.tum.de

Abstract

Deep learning techniques have been used to generate realistic indoor scenes by splitting the task into an object prediction and an object retrieval step. The fitted models predict high-level attributes about furniture that will be placed in a synthetic room, which are matched using an external CAD dataset. These methods achieve realistic placement of predetermined furniture. However, this naive retrieval restricts the diversity of these methods, while also resulting in a costly query into a generally large CAD dataset at runtime.

We propose to solve this task end-to-end by directly generating a shape representation as an additional attribute of each object placed in the scene. To this end, we base our implementation on previous works, which model objects in a scene by predicting their class label, orientation, position, and size [7]. We further predict a per-object shape code, from which we will obtain 3D objects using an autoencoder trained on voxel grids. Our model shows good levels of realism and has a low memory footprint, eliminating the need for external resources to generate novel indoor scenes.

1. Introduction

Synthetic 3D scenes are used in real-life products and find use in applications such as video game development and VR/AR technologies. Interior design companies typically spend substantial funds to portray and promote their merchandise to decorate indoor spaces. Therefore, some companies opt to leverage realistic virtual spaces for marketing imagery [11]. This demand has become an incentive in recent years to develop tools for the manual design of 3D content, but more interestingly, for its automatic generation [6, 13].

Recent works in indoor scene synthesis favor auto-regressive generative models that define the arrangement of objects and characteristics to return relevant 3D models from a CAD dataset [7, 8, 14]. From an empty floor plan or partial scene, a set of objects is generated in sequence, each influenced by previous predictions. Each object is characterized by a class label, a 3D location, its orientation, and a 3D size, which are used to position and retrieve the CAD model. These methods achieve convincing results, partly influenced by the realistic pieces of furniture retrieved from a CAD dataset.

However, we feel that this method introduces an unnatural separation of concerns. In particular, these methods completely separate the composition of the objects belonging to different classes in a scene, from the actual representation of the individual objects. This means that the furniture's 3D object representation has no influence on the learning and generation process.

Moreover, this method requires storage and access to a possibly large dataset, while also limiting the diversity of the scenes to the furniture present within this dataset. To alleviate these issues, we propose to tackle scene generation in an end-to-end fashion to forego the need for an external CAD dataset. For this, we create an implementation based on the auto-regressive transformer framework of ATISS [7] and enhance it by additionally generating a *shape code* for each predicted piece of furniture. This is a low-dimensional representation of an object's shape from which we can subsequently decode a 3D object. This encoding allows us to train explicitly on furniture information and also to benefit from scene diversity stemming from the rich auto-regressive generative model. To generate these shape codes for our training data and in order to later on decode these shape codes into new objects, we create a 3D autoencoder for voxelized 3D furniture data. This autoencoder will be trained on the 3D-Future dataset and have two important functions: To compress data to a latent space of shape codes, and to then expand shape codes into complete 3D models. The project code is available at https://github.com/luisdavid64/Adl4cv-synthesis.

2. Related Work

Relevant works have explored different strategies for interior scene synthesis. SceneFormer [10] utilizes a transformer architecture to decode a sequence of objects to place in the scene. ATISS [7] uses a similar architecture but instead generates an unordered set of objects, which produces state-of-the-art results.

Deep neural networks have also been widely used to build models that can learn to generate 3D shapes [5]. Of interest to our project, autoencoders and variational autoencoders [4] have been used successfully to take volumetric data and regenerate 3D shapes [1] for use in object classification.

3. Method

Our goal is to build on the existing approach of the ATISS framework [7] by removing the requirement of an external database containing furniture objects at runtime. We thus broadly inherit the scene generation from this paper. To achieve independence from an external furniture dataset, we will regress an additional shape code per object using the transformer output and object attributes.

3.1. Autoregressive Scene Generation

We let $\mathcal{X} = \{\mathcal{X}_1, ...\mathcal{X}_N\}$ denote a collection of scenes, where $\mathcal{X}_i = \{\mathcal{O}_i, \mathbf{F}^i\}$ and $\mathcal{O}_i = \{o_i^j\}_{j=1}^M$ is a collection of objects in the scene and $\mathbf{F}^i$ is a floor plan. ATISS seeks a permutation invariant solution to the generation of objects. With this, the task is defined by the following approximation [7] of the log-likelihood for generating a scene in all possible orderings $\mathcal{X}$:

$$log\hat{p}(\mathcal{X}) = \sum_{i=1}^{N} \sum_{\hat{\mathcal{O}} \in \pi(\mathcal{O}_i)} \sum_{j \in \mathcal{O}} log p_\theta(o_j^i | o_{<j}^i, \mathbf{F}^i) \qquad (1)$$

where $log p_\theta(o_j^i | o_{<j}^i, \mathbf{F}^i)$ is the probability of the j-th object, given the previously generated objects for floor plan $\mathbf{F}^i$, and π is a function that calculates the permutations of all objects in the scene.

3.2. Modelling Objects

Unlike previous works [7, 10], we model the scenes with five random variables: category, location, orientation, size, and an additional shape code variable $o_j = \{c_j, t_j, r_j, s_j, sh_j\}$. We then follow the classic autoregressive method used in previous works with the addition of shape code prediction:

$$\begin{aligned} & p_\theta(o_j|o_{<j}, F) = p_\theta(c_j|o_{<j}, F)\, p_\theta(t_j|c_j, o_{<j}, F) \\ & p_\theta(r_j|t_j, c_j, o_{<j}, F)\, p_\theta(s_j|r_j, t_j, c_j, o_{<j}, F) \qquad (2) \\ & p_\theta(sh_j|s_j, c_j, o_{<j}, F) \end{aligned}$$

While the object class c_j is modeled using a categorical variable over the total number of object categories, the position, orientation, and size are modeled using a discrete mixture of logistic distributions [9]. The shape code is regressed on the mean-squared error (MSE) of its D-dimensional vector. For our experiments, we choose $D = 128$.

3.3. Network Architecture

The input to our network consists of a collection of scenes in the form of 3D labeled bounding boxes with their corresponding room shape. We follow the architecture of ATISS which contains: (i) the *layout encoder* which maps the floor plan into a global feature $\mathbf{F}$, (ii) the *structure* encoder which converts objects into a context embedding $\mathbf{C} = {\mathbf{C}_j}_{j=1}^M$, (iii) the *transformer encoder* that takes $\mathbf{F}, \mathbf{C}$ and a query $\mathbf{q}$ that predicts the features of the next object $\hat{\mathbf{q}}$, and a modified *attribute extractor* which predicts the attributes of the next object including its *shape code*. We will now describe our modified *attribute extractor*. Further information about the other parts of the network can be found in the ATISS paper [7].

3.4. Attribute extractor

Following our auto-regressive model, we predict the attributes of the next object in the scene using an MLP for every attribute. The formal definitions are

$$\begin{aligned} & c_\theta : \mathcal{R}^{64} \to \mathcal{R}^C && \hat{\mathbf{q}} \mapsto \hat{\mathbf{c}} \\ & t_\theta : \mathcal{R}^{64} \times \mathcal{R}^{Lc} \to \mathcal{R}^{3\times3\times K} && (\hat{\mathbf{q}}, \lambda(\mathbf{c})) \mapsto \hat{\mathbf{t}} \\ & r_\theta : \mathcal{R}^{64} \times \mathcal{R}^{Lc} \times \mathcal{R}^{Lt} \to \mathcal{R}^{1\times3\times K} && (\hat{\mathbf{q}}, \lambda(\mathbf{c}), \gamma(\mathbf{t}) \mapsto \hat{\mathbf{r}} \\ & s_\theta : \mathcal{R}^{64} \times \mathcal{R}^{Lc} \times \mathcal{R}^{Lt} \times \mathcal{R}^{Lr} \to \mathcal{R}^{3\times3\times K} && (\hat{\mathbf{q}}, \lambda(\mathbf{c}), \gamma(\mathbf{t}), \gamma(\mathbf{r})) \mapsto \hat{\mathbf{s}} \\ & sh_\theta : \mathcal{R}^{64} \times \mathcal{R}^{Lc} \times \mathcal{R}^{Ls} \to \mathcal{R}^D && (\hat{\mathbf{q}}, \lambda(\mathbf{c}), \gamma(\mathbf{s})) \mapsto \mathbf{s\hat{h}} \end{aligned} \qquad (3)$$

where $\hat{\mathbf{c}}, \hat{\mathbf{t}}, \hat{\mathbf{r}}, \hat{\mathbf{s}}$ are the predicted distributions and $\mathbf{s\hat{h}}$ is a predicted D-dimensional vector. On the other hand, $c_\theta, t_\theta, r_\theta, s_\theta, sh_\theta$ are mappings between the transformer latent space and the low-dimensional space of attributes. As formulated before, the attributes are fed in auto-regressive sequence to predict the next attribute. This means we concatenate $\hat{\mathbf{q}}$ with the previously determined attributes to form the input for the MLP of the next attribute. The exception to this is the generation of the shape code for which only class and size attributes are considered. This mimics the object retrieval used in ATISS and since the transformation and rotation are applied to the finished object, these attributes seem superfluous to the generation of its shape code. The embeddings $\lambda(\cdot), \gamma(\cdot)$ map the attributes into a high dimensional space.

3.5. Training and Inference

During training, a random scene is chosen and a random permutation of the objects in the scene is generated. Then, a subset of these objects is selected to compute $\mathbf{C}$ and we task our network to predict the remaining objects conditioned on $\mathbf{C}, \mathbf{F}$. From this, we obtain the attribute distributions and shape code of the next object. We directly apply the L2-loss on our shape code, allowing us to skip the decoding step during training. We train to maximize the log-likelihood of the predicted object.

During scene generation, we use the decoder part of our autoencoder to create a voxel representation of an object using the generated shape codes. The final 3D models are then obtained by applying marching cubes to this volumetric grid. The autoencoder will be detailed in the next section.

Our network is trained end-to-end on the 3D-FRONT bedroom/living room dataset over 200 epochs of 500 steps each, using a batch size of 128 scenes.

3.6. Autoencoder for 3D Shape Generation

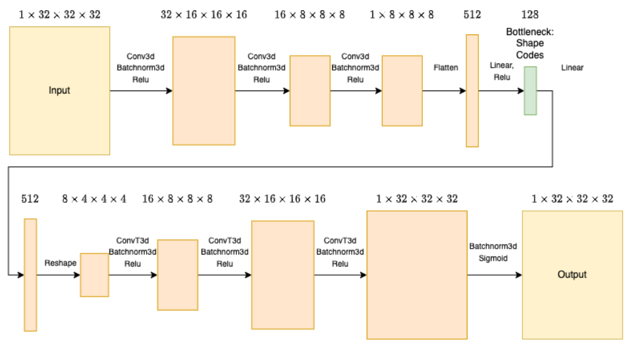
$1 \times 32 \times 32 \times 32$
Input
Conv3d
Batchnorm3d
Relu
$32 \times 16 \times 16 \times 16$
Conv3d
Batchnorm3d
Relu
$16 \times 8 \times 8 \times 8$
Conv3d
Batchnorm3d
Relu
$1 \times 8 \times 8 \times 8$
Flatten
512
Linear,
Relu
128
Bottleneck:
Shape
Codes
Linear
512
Reshape
$8 \times 4 \times 4 \times 4$
ConvT3d
Batchnorm3d
Relu
$16 \times 8 \times 8 \times 8$
ConvT3d
Batchnorm3d
Relu
$32 \times 16 \times 16 \times 16$
ConvT3d
Batchnorm3d
Relu
$1 \times 32 \times 32 \times 32$
Batchnorm3d
Sigmoid
$1 \times 32 \times 32 \times 32$
Output

Figure 1. 3D Autoencoder Architecture

We need a mechanism to generate 3D shapes from our low-dimensional vectors. To this end, we settled on using a 3D convolutional autoencoder, where the latent space is constrained to the size of our desired shape codes. The input of the autoencoder is a $32 \times 32 \times 32$ voxel grid and is trained to reconstruct the input using the MSE (mean squared error) loss function. The final layer uses a sigmoid activation to squeeze outputs between 0 and 1. For the size of our latent space, we opt for sizes that provide a good balance between compression (low-memory demand) and retention of information. Our implementation follows the work by Y. Guan et al. [3] and its architecture can be found in Figure 1.

4. Results

In the following, we will look at the performance and results of our shape autoencoder as well as the entire scene generation network.

4.1. Scene Generation Network

As our network is based on the ATISS architecture, we support all its possible tasks including scene generation from a floor plan or partial scene, scene error detection and correction, as well as object suggestion. For the following results, we are focused on the scene generation aspect, as our work is focused on high-quality object generation as opposed to only scene arrangement.

4.1.1 Qualitative Evaluation

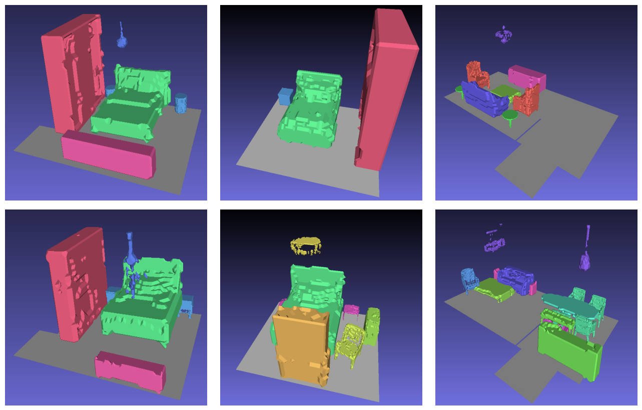

Figure 2. Ground truth (top) and network generated scenes (bottom). The ground truth scenes taken from the 3D-FRONT dataset have been voxelized and remeshed for a direct comparison of the network's reconstruction quality.

For the evaluation of our network, we first look at scenes generated by the network compared to scenes from the 3D-FRONT dataset. In Figure 2 voxelized and remeshed versions of scenes taken from the original bedroom and living room datasets are shown alongside the network's generated scene using the corresponding scene's floor plan. The objects are colored according to their furniture class.
We find that our network produces visually sound and unique scene arrangements. Objects are visually recognizable and generally have a high reconstruction quality.

In particular, solid structures such as beds and wardrobes are generated with high accuracy. However, we find that reconstruction quality varies between object classes. This issue is especially noticeable with lamps, which generally have a finer structure. This is further explored in the following sections.

4.1.2 Quantitative Evaluation

As our method focuses on extending existing approaches to an end-to-end framework directly generating objects, our evaluation will also focus on this reconstruction aspect. The voxelized representation used in our approach makes the FID and scene classification metrics used in previous works not comparable, as objects placed in scenes by our network clearly differ in structure from the ones retrieved from CAD datasets such as 3D-FUTURE. Nevertheless, since our architecture builds on the ATISS network, its previous outputs, class, translation, bounding box size and rotation are preserved.

To evaluate the quality of our individual object reconstructions, we calculate the Chamfer distance average for each furniture class by generating a scene for each floor plan in the test set. The distance is then calculated on the point cloud of voxels from each generated object's shape code to the object with the closest shape code in the 3D-FUTURE dataset. The average result across 10 runs is shown in Table 1.

	Bedroom			Living Room	
Class	128	256	Count	128	Count
TV Stand	**0.74**	0.86	252	0.57	1437
Wardrobe	**0.78**	0.82	1301	2.07	55
Single Bed	**0.89**	1.20	149	/	0
Double Bed	**1.14**	1.45	1442	/	0
Children Cabinet	**1.93**	3.64	33	/	0
Bookshelf	2.42	**2.10**	58	3.43	477
Cabinet	**2.61**	3.93	150	2.97	546
(Dining) Chair	**3.09**	5.26	139	1.80	7153
Kids Bed	**3.71**	7.84	55	/	0
Nightstand	**4.19**	5.88	2122	/	0
Dressing Table	**4.28**	4.42	139	/	0
Dressing Chair	**4.59**	9.22	43	/	0
Armchair	**6.76**	30.07	26	5.00	1108
Desk	7.98	**5.26**	107	54.85	17
Coffee Table	**10.06**	17.33	13	12.29	1691
Ceiling Lamp	**11.81**	21.18	519	13.49	407
Sofa	**13.63**	17.70	16	/	0
Shelf	16.04	**10.89**	63	6.74	93
Stool	**20.93**	38.36	91	22.83	531
Table	**24.16**	44.34	167	/	0
Pendant Lamp	37.67	**27.06**	1002	40.54	3169

Table 1. Mean Chamfer Distance per furniture class. Number of class occurences during the test are also shown. The size of the bottleneck layer for the Autoencoder is either 128 or 256, as noted.

The results are sorted by the Chamfer Distance, with simple, uniform objects such as TV stands, wardrobes and beds having superior reconstruction qualities. On the other hand, objects with low occurrence in the dataset suffer a quality loss. Apart from those, we notice the previously mentioned loss of quality during the reconstruction of lamps, particularly those of the structurally diverse and thin pendant lamp class. Surprisingly, an increase in the latent space size for the autoencoder (translating to our shape codes) decreased the final object quality. While the autoencoder itself improves the direct reconstruction of input objects as will be shown in subsection 4.2, it performs worse for most classes when used as a decoder in the scene generation network. We suspect that the increase in latent space size decreases the generalization of the network while reducing the coherence/clustering of object classes in the latent space. Due to these results, we choose to remain with the latent space size of 128 for our experiments.

4.2. Autoencoder Results and Ablation studies

As previously noted, we found that the apparent reconstruction quality using the autoencoder varies between object classes. An overview of each class' reconstruction quality using Chamfer Distance is shown in Table 2 for latent space sizes of 128 and 256. Notably, objects with regular structures display better reconstructions. This is the case for objects such as tables and wardrobes which contain a large rectangular component. On the other hand, pendant lamps and ceiling lamps show comparatively higher Chamfer distances due to their sparse appearance and high intra-class variance.

Class	128	256	Class	128	256
Armchair	0.61	**0.40**	Stool	0.51	**0.3**
Lounge Chair	0.64	**0.39**	Console Table	0.27	**0.17**
Pendant Lamp	3.89	**1.86**	Ceiling Lamp	1.34	**0.89**
Coffee Table	0.75	**0.34**	Round-End Table	1.32	**0.54**
Corner-Side Table	0.55	**0.28**	Desk	0.59	**0.31**
Dining Table	0.44	**0.22**	Single Bed	0.35	**0.26**
Double Bed	0.31	**0.24**	Love-seat Sofa	0.35	**0.28**
Nightstand	0.24	**0.15**	Dining Chair	0.45	**0.29**
Bookshelf	0.34	**0.23**	Lazy Sofa	0.66	**0.43**
Multi-Seat Sofa	0.30	**0.24**	L-Shaped Sofa	0.38	**0.29**
TV Stand	0.20	**0.14**	Wine Cabinet	0.26	**0.2**
Cabinet	0.24	**0.16**	Dressing Table	0.58	**0.29**
Shelf	0.92	**0.54**	Dressing Chair	0.61	**0.37**
Wardrobe	0.16	**0.11**	Kids Bed	0.53	**0.34**
Chinese Chair	0.59	**0.37**	Chaise Lounge Sofa	0.46	**0.32**

Table 2. Mean Chamfer Distance across object classes. Comparison of bottleneck layer sizes.

Moreover, the table demonstrates the positive correlation between reconstruction quality and bottleneck layer size. Doubling the bottleneck size from 128 to 256 halves Chamfer Distances across most objects, with the effect being less dramatic on already well-reconstructed objects such as wardrobes. This increase also means doubling the memory footprint of our shape codes down the line.

However as previously stated in subsubsection 4.1.2, we found that this does not lead to better results during scene generation. This is likely because the autoencoder is trained on the entire dataset and increasing the size of the latent space allows it to overfit more and generalize less, leading to individual classes in the latent space being more loosely spread.

5. Conclusion and Discussion

Our auto-regressive model achieves an acceptable level of realism without the burden of an additional dataset to populate our predicted scenes. It is flexible and retains the capabilities of ATISS to work on different applications, including scene generation scene completion, object suggestion and failure detection. Our results show that scene generation can be tackled end-to-end, with explicit supervision on shape codes to generate realistic objects to populate scenes with.

As expected, the down-sampling of the CAD dataset for our voxel autoencoder, together with the compression effect of the latent space, can lead to undesired artifacts or sparsity in final output objects. These issues could be remediated with larger voxel grids or training an autoencoder with a completely different representation, such as point clouds [12] or meshes [2, 15]. Similarly, the use of RGB voxel grids would increase realism by re-adding color data.

References

[1] Andrew Brock, Theodore Lim, James M. Ritchie, and Nick Weston. Generative and discriminative voxel modeling with convolutional neural networks. *ArXiv*, abs/1608.04236, 2016. 2

[2] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. A papier-mache approach to learning 3d surface generation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 216–224, 2018. 4

[3] Yanran Guan, Tansin Jahan, and Oliver van Kaick. Generalized autoencoder for volumetric shape generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020. 3

[4] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014. 2

[5] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. pages 2836–2844, 06 2021. 2

[6] Paul C. Merrell, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. Interactive furniture layout using interior design guidelines. *ACM SIGGRAPH 2011 papers*, 2011. 1

[7] Despoina Paschalidou, Amlan Kar, Maria Shugrina, Karsten Kreis, Andreas Geiger, and Sanja Fidler. Atiss: Autoregressive transformers for indoor scene synthesis. In *NeurIPS*, 2021. 1, 2

[8] Daniel Ritchie, Kai Wang, and Yu-An Lin. Fast and flexible indoor scene synthesis via deep convolutional generative models. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6175–6183, 2019. 1

[9] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *CoRR*, abs/1701.05517, 2017. 2

[10] Xinpeng Wang, Chandan Yeshwanth, and Matthias Nießner. Sceneformer: Indoor scene generation with transformers. *2021 International Conference on 3D Vision (3DV)*, pages 106–115, 2021. 1, 2

[11] Henry Winchester. Putting the cgi in ikea: How v-ray helps visualize perfect homes, Mar 2018. 1

[12] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. 4

[13] Lap-Fai Craig Yu, Sai-Kit Yeung, Chi-Keung Tang, Demetri Terzopoulos, Tony F. Chan, and S. Osher. Make it home: automatic optimization of furniture arrangement. *ACM SIGGRAPH 2011 papers*, 2011. 1

[14] Song-Hai Zhang, Shaokui Zhang, Wei-Yu Xie, Cheng-Yang Luo, and Hong-Bo Fu. Fast 3d indoor scene synthesis with discrete and exact layout pattern extraction. *ArXiv*, abs/2002.00328, 2020. 1

[15] Yi Zhou, Chenglei Wu, Zimo Li, Chen Cao, Yuting Ye, Jason M. Saragih, Hao Li, and Yaser Sheikh. Fully convolutional mesh autoencoder using efficient spatially varying kernels. *CoRR*, abs/2006.04325, 2020. 4