International Journal of Applied Engineering Research ISSN 0973-4562 Volume 18, Number 2 (2023) pp. 97-102

Automatic Generation of AI-powered Architectural Floor Plans using Grid Data

Hun Lim

Master's Student, Department of Computer Science and Engineering
Incheon National University
(Songdo-dong) 119 Academy-ro, Yeonsu-gu, Incheon, Republic of Korea

Abstract
Architectural floor plans require a lot of effort and time because it needs to reflect various factors such as building regulations, designer requirements, and residents' needs. With the development of artificial intelligence technology, research is underway to automate drawing generation. However, GAN models, which are often used for floor plan generation, have the drawback of consuming a lot of time and resources during training and drastically decreasing performance when the training data is insufficient. In this paper, a method is proposed to compress the original data while preserving important information, enabling fast and effective use of floor plan models in various computing environments. Additionally, through experiments, it is confirmed that the proposed method allows for fast training and stable automatic floor plan generation.

Keywords: Floor plan, GAN, CGAN, Neural Networks, Data Compression

Introduction

According to the International Energy Agency (IEA), the area of world buildings is expected to increase by about 235,000,000,000 m² by 2050 to accommodate population growth [1].
Floor plans are the most essential element in designing a building, as they are created by architects to effectively communicate their intentions to clients and construction workers. To create these drawings, a great deal of effort and time is required, as they must incorporate various factors such as building regulations, the architect's vision, and the needs of the occupants.
With recent advances in artificial intelligence technology, various AI models are being researched, and in particular, Generative Adversarial Networks (GANs) have shown excellent performance in image generation. For this reason, there have been various attempts to use GANs to generate architectural drawings in recent years [3,4,5,6]. However, there is currently no automatic spatial layout technology that can replace skilled architects [7].
Meanwhile, interest in the metaverse is increasing with the development of IT technologies such as artificial intelligence, virtual reality, and blockchain. Accordingly, there is a need for an artificial intelligence model that can be executed quickly and effectively in various computing environments. However, GAN has the disadvantage of rapidly degrading performance when the amount of learning data is small [8,9], and learning itself consumes a lot of time and resources.
To solve these limitations, various methods are being studied, such as using images instead of noise vectors in the generator to improve the performance of the training [10], using filters to extract features from images for training [11], and changing the loss function to improve the instability of the training [12].
In this paper, we aim to address the limitations of GAN model training for architectural drawing generation by collecting floor plan data and converting it into grid data through pre-processing. By compressing the data while maintaining the information, we aim to reduce the amount of computing resources used in training.
To demonstrate the advantages of our proposed method, we conducted performance evaluations using both the original data and the data that underwent our proposed pre-processing method. We aimed to show that our proposed method can generate drawings faster while maintaining comparable performance to using the original data.

Related Works

Research on introducing computer technology into the construction industry has been ongoing for a long time. In the past, research focused on specific calculation methods such as algorithms and agent-based methods, but currently, various AI technologies based on machine learning are being introduced not only in the field of drawing automation but also in real estate and construction industries. In this section, we will briefly examine modern research that has incorporated machine learning and discuss research related to floor plans.
In the past, studies in the construction field have mainly focused on optimization and automation technologies using algorithms [13] and agent-based methods [14]. These studies were conducted in various areas such as building design, construction process, operation and maintenance, and computer simulation and optimization techniques were applied to solve complex problems.
With the development of machine learning technology, research is actively being conducted to introduce machine learning into the construction industry. Convolutional neural networks (CNNs), which show high performance in image recognition, and Generative Adversarial Networks (GANs), which show good performance in generating architectural drawings, are receiving attention in this area.
Recently, various studies have been conducted using GANs with graph structures such as GCGAN (graph-constrained generative adversarial network) [3], which generates rooms constrained by graphs between rooms, and models using

Conditional Generative Adversarial Networks (CGAN) to divide empty rooms into multiple functional areas and learn various functional areas to place furniture in specific functional areas [4].

Data Collection and Pre-processing

In machine learning, data is crucial. Therefore, research on collecting and creating datasets by generating and collecting floor plan is also actively being conducted. Examples of such datasets include SUNCG [15], Structured3D [16], OpenRoom [17], and 3D-FRONT [18], which store drawings in 3D space and place furniture in the space to create 3D drawing datasets, and CubiCasa5k [19], which focuses on 2D drawing data. However, 3D drawing datasets with placed furniture require a lot of resources, and CubiCasa5k is not suitable for this study.

Therefore, in this section, we introduce the method of collecting and pre-processing suitable data for our research. The collected architectural drawing data is a 2D 3-channel RGB indoor drawing image dataset of size 676 x 923 pixels, containing information on the structure and function of the rooms as well as the wall structure represented by black lines. The data was collected based on Korean real estate properties and collected randomly regardless of the structure of the region or building.

a) Data Labelling

The data was collected from the NAVER real estate website in Korea, and the data is already standardized. However, since there is no separate labeling, labels that contain information about the data are needed. Before pre-processing the data, we use image recognition to identify the collected data and extract the function of each room from the image. Since the data is standardized, we can extract the room functions using computer vision technology without a separate trained neural network. Then, we extract the located label data, which consists of the function of each room (bedroom, balcony, living room, etc.) and the number of rooms. The detailed process is shown in Algorithm 1.

Algorithm 1 Labeling Algorithm

```
Input: Room Images
Initialize Label List L
Initialize Room Label List RLL
Initialize Room List RL
foreach image in Room Images
get RL from image
foreach Room in RL
foreach Label in RLL
if Room==Label
L[image.index][Room]++
end
end
end
```

In this research, labeling was performed based on five spaces (bedroom, living room, balcony, kitchen, and bathroom). These spaces correspond to the bedroom, living room, kitchen, balcony, and bathroom, respectively. The labels are arranged in this order, and each label's value ranges from 1 to 10.

Figure 1: Collected data and label data

b) Data Labelling

For the floor plan data, a dataset consisting of various architectural design drawings as shown in Figure 2 is construct.

Figure 2: Floor plan dataset

For this dataset, if the functions of the rooms are the same, the extracted label and the color of the structure are constant. Therefore, by analyzing the labels and colors of the structure in the image, the functions of the rooms can be identified and new fixed values can be assigned to each room, transforming the image to be like (b) in Figure3. By dividing the colors of each room, such as (255,255,0) for the bedroom, (0,0,255) for the balcony, and (255,0,255) for the living room, the information can be stored, and the color of the data can represent the function of the corresponding room.

Then, the transformed data with fixed values assigned is converted into a gridded data of 20x20. In this process, the data is divided into areas of 20x20 horizontally and vertically, and the mode(the most observed number in statistics) value of each area is designated as the representative value and saved as shown in (c) of Figure 3.

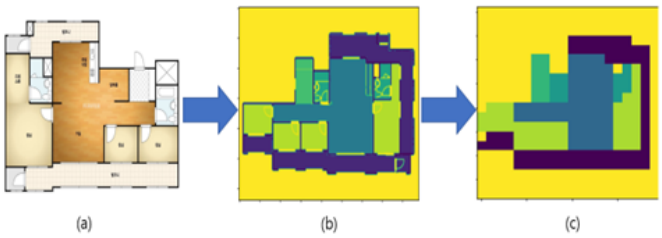

Figure 3: Data preprocessing((a) recognizing the structure of the room, (b) assigning RGB values to fixed values, and (c) creating a 20*20 gridded data)

Training and Generation

To demonstrate the advantages of the gridded data conversion technique in this paper, the authors first used a GAN model to generate images, and then compared and analyzed the data with and without the gridded data conversion technique applied. Next, they proceeded with training and generation using a CGAN model. This section shows how the training and generation for each model were conducted.

a) GAN

The generation of the floor plan involves training a GAN model using the preprocessed floor plan dataset by inputting the noise vector z to the generator neural network and the preprocessed floor plan dataset to the discriminator neural network.

BCE (Binary Cross Entropy) Loss and L2 Loss, also known as MSE (Mean Squared Error) Loss, were used in experiments as the loss functions to train the model. The experimental results showed that even when hyperparameters were tuned to produce the best output results, the L2 Loss caused incomplete data to be generated. Therefore, BCE Loss was adopted as the loss function.

The BCELoss loss function is shown in Equation 1.

$$L(G,D)=E_{x\sim P_{data}(x)}[\log D(x)]+E_{z\sim P_z(z)}[\log(1-D(G(z)))] \quad (1)$$

The discriminator D was trained using the preprocessed dataset from Section 3, and the output of generator G is floor plan data with a size of 20*20. The structure of the generator neural network is shown in Figure 5, which illustrates the architecture of the floor plan generation neural network.

The generator G and discriminator D are implemented as 3-layer neural networks using PyTorch, with the layer structures shown in Figure 6.

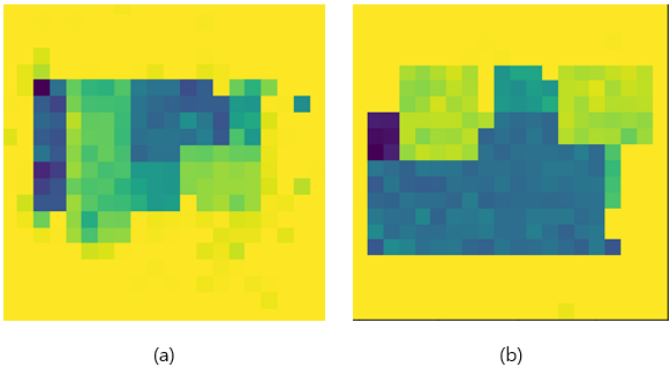

Figure 4: Results when tuned to produce the best output (L2Loss(a), BCELoss(b))

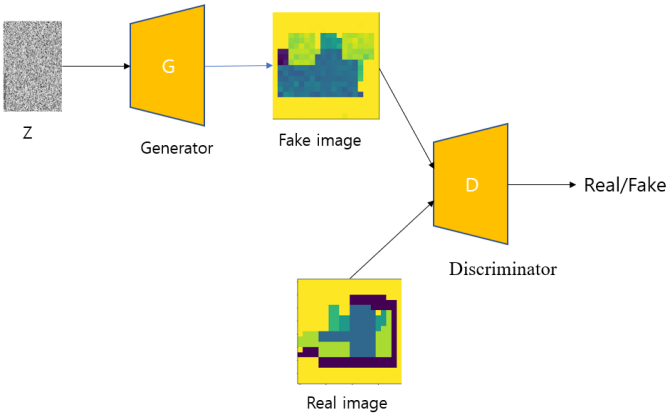

Figure5: Structure of Floor plan GAN

```
Sequential(
  (0): Linear(in_features=64, out_features=256, bias=True)
  (1): ReLU()
  (2): Linear(in_features=256, out_features=256, bias=True)
  (3): ReLU()
  (4): Linear(in_features=256, out_features=400, bias=True)
  (5): Tanh()
)
Sequential(
  (0): Linear(in_features=400, out_features=256, bias=True)
  (1): LeakyReLU(negative_slope=0.2)
  (2): Linear(in_features=256, out_features=256, bias=True)
  (3): LeakyReLU(negative_slope=0.2)
  (4): Linear(in_features=256, out_features=1, bias=True)
  (5): Sigmoid()
)
```

Figure6: The neural network layer structures of the generator(above) and discriminator(below) in a GAN model represented in PyTorch code

b) GAN

CGAN is a GAN model that uses filters to extract image features for learning. It can restrict the generated image by inputting filters through an embedding vector C. In this paper, various experiments were conducted based on the labels generated in Section 3 to verify the results.

The generator G and discriminator D were implemented as 4-layer neural networks using PyTorch, with the layer structures shown in Figure 8.

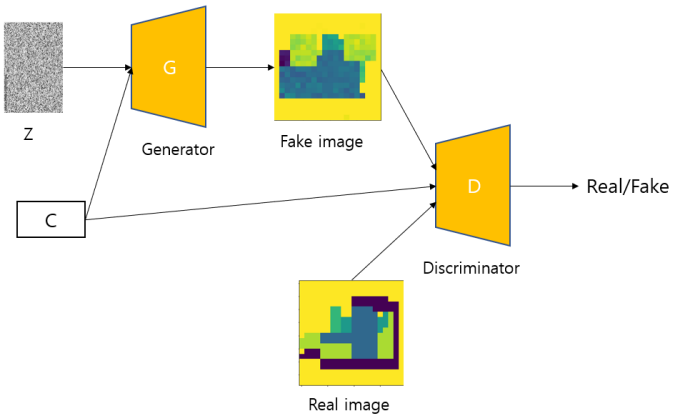

Figure 7: Structure of Floor plan CGAN

```
Sequential(
  (0): Linear(in_features=794, out_features=1024, bias=True)
  (1): LeakyReLU(negative_slope=0.2, inplace=True)
  (2): Dropout(p=0.3, inplace=False)
  (3): Linear(in_features=1024, out_features=512, bias=True)
  (4): LeakyReLU(negative_slope=0.2, inplace=True)
  (5): Dropout(p=0.3, inplace=False)
  (6): Linear(in_features=512, out_features=256, bias=True)
  (7): LeakyReLU(negative_slope=0.2, inplace=True)
  (8): Dropout(p=0.3, inplace=False)
  (9): Linear(in_features=256, out_features=1, bias=True)
  (10): Sigmoid()
)
Sequential(
  (0): Linear(in_features=110, out_features=256, bias=True)
  (1): LeakyReLU(negative_slope=0.2, inplace=True)
  (2): Linear(in_features=256, out_features=512, bias=True)
  (3): LeakyReLU(negative_slope=0.2, inplace=True)
  (4): Linear(in_features=512, out_features=1024, bias=True)
  (5): LeakyReLU(negative_slope=0.2, inplace=True)
  (6): Linear(in_features=1024, out_features=784, bias=True)
  (7): Tanh()
)
```

Figure 8: The neural network layer structures of the generator(above) and discriminator(below) in a CGAN model represented in PyTorch code

Result and Analysis

In this section, we compare the data generated using the grid compression with that generated without using it, by training and generating them using a GAN model. The disadvantage of GAN is that it does not allow for input of desired conditions, making it difficult to control the generated output in a desired direction. So we also experimented with a CGAN model that takes in the label generated in Chapter 3 as input to adjust the generation of floorplans, as conditions such as the number of rooms are essential in floorplan generation.

a) GAN Model Results and Analysis

To verify the advantages of this study, comparison experiments were conducted between the data with and without grid compression. Figure 9 shows the time taken for learning and generating Floor plan for 'grid' (preprocessed data) and 'non_grid' (unprocessed data) according to the batch size and epoch for automatic floor plan generation. Table 1 shows the time taken for learning and generating Floor plan for 'grid' (preprocessed data) and 'non_grid' (unprocessed data) according to the batch size and epoch for automatic floor plan generation.

As shown in Figure 10, the method using grid compression shows relatively stable results even though it uses only about 0.2% of the training data size and about 30% of the training time compared to the method without grid compression. The generated data from the gridded data generally reflects the structure of the room but shows ambiguous boundaries between rooms

In addition, 'non_grid' showed a side effect of generating only similar results in shape, as shown in Figure 11, even when the input noise vector z was randomly generated after reducing the batch size or increasing the training period for extensive learning.

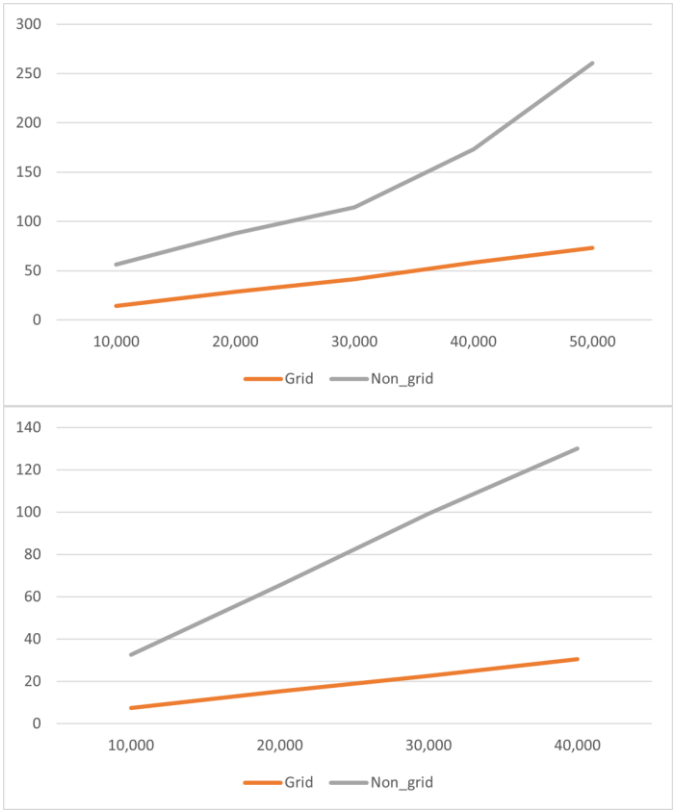

Figure 9: Changes in learning time according to epochs (above: Batch size 25, below: batch size 50)

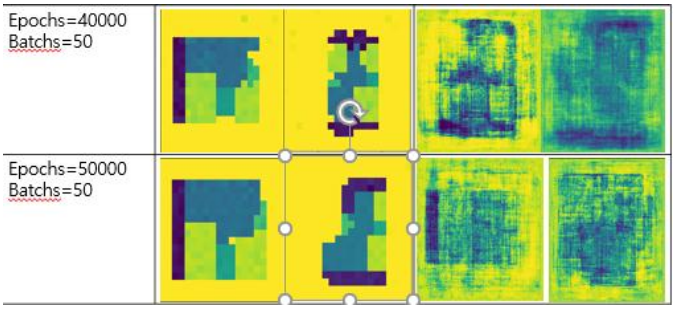

Figure 10: Generation results by batch size when the epochs is 50,000

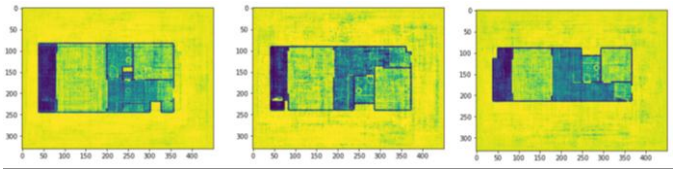

Figure 11: Even when using random tensors, similar-shaped drawings are outputted, indicating overfitting.

b) Number of Bedroom CGAN model Results and Analysis
It was decided to use only the bedroom label, which has the most diverse distribution and is deemed important in drawing generation, as the embedding for training. Figure 12 shows generated images according to the number of rooms (1-9) for 10,000 epochs, a batch size of 50. For smooth experimentation, the output images of the CGAN were printed as black and white images.

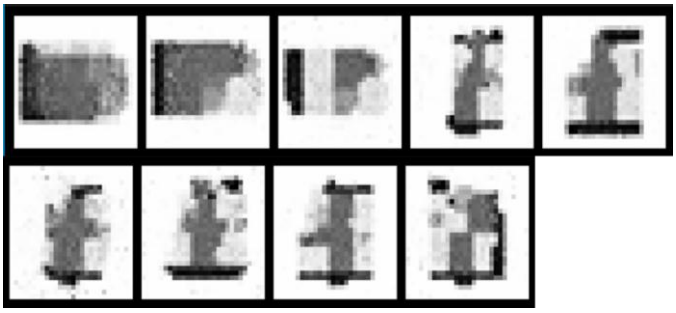

Figure 12: Bedroom Embedded CGAN Generation results when the epochs is 10,000, batch size is 50

c) Sum of All Rooms Embedded CGAN model Results and Analysis
If each label is trained separately, it requires 100,000 possible cases. This requires a significant amount of resources. To solve this issue, the sum of each label was calculated and the distribution of the values was examined. It was determined that the range of 5 to 15 would best represent the data, so the values were mapped to this range, relabeled, and then trained. The distribution of values can be seen in Figure13.

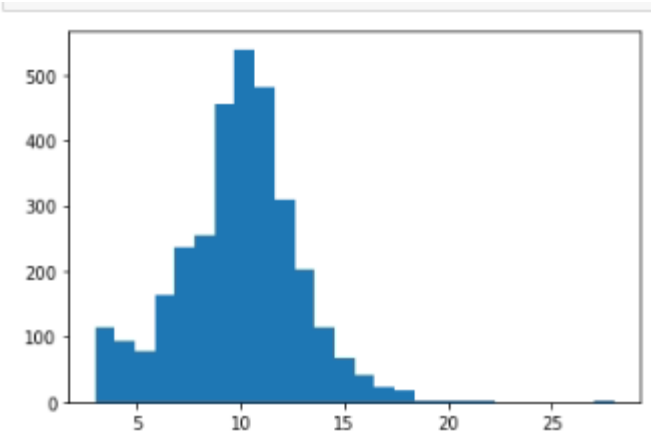

Figure 13: Histogram of embeddings of all rooms

Figure 14 shows the generated images with 10000 Epoch and a batch size of 50. Each generated image was labeled according to the range of 5 to 15 after relabeling and training.

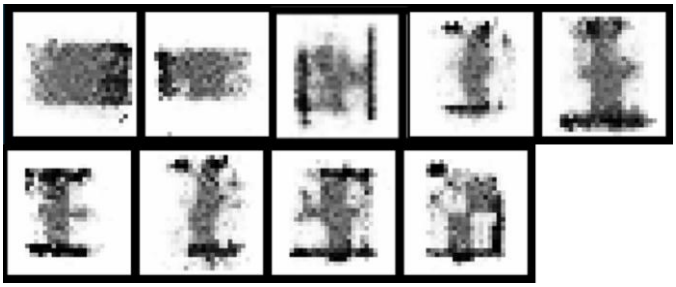

Figure 14: Sum of All Rooms CGAN Generation results when the epochs is 10,000, batch size is 50

Conclusions and Future Research
In this paper, we proposed a method for quickly and effectively using GAN models by compressing data while maintaining essential information in the automatic generation of floor plan. We analyzed and preprocessed the floor plans used in the training to compress the size of the data while minimizing the loss of information. Moreover, it was experimentally confirmed that similar levels of automatic drawing generation were possible with only 0.2% of the original data capacity after preprocessing. However, the generated drawings showed limitations in the ambiguity of the boundaries of the room structure, and not all generated labels were used in the testing process of the CGAN model.
In future research, we will investigate methods to utilize all generated labels and preserve the wall data of the original data before preprocessing. By combining the wall data back into the data after preprocessing, we aim to clearly depict the boundaries between rooms and improve the quality of the generated drawings. Furthermore, as noise remaining in the GAN learning process is critical due to the nature of the drawings, we intend to address this issue.
In future research, we will study a method to improve the quality of the generated floor plan and clearly depict the boundaries between rooms by preserving the wall data from the

original data before preprocessing and synthesizing the wall data back into the data after preprocessing for learning. Additionally, due to the structure of GAN, noise used during the generation remains in the floor plan, which leads to a decrease in the quality of the drawings; therefore, we aim to address this issue.

ACKNOWLEDGMENT
This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 202001410003).

References

[1] M. W¨orsd¨orfer, T. Gü, J. Dulac, T. Abergel, C. Delmastro, P. Janoska, K. Lane, A. Prag, Perspectives for the Clean Energy Transition – The Critical Role of Buildings, Paris. www.iea.org, 2019

[2] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, "Generative adversarial nets", In NIPS, pp. 2672–2680, 2014

[3] Nelson Nauata, Kai-Hung Chang, Chin-Yi Cheng, Greg Mori, Yasutaka Furukawa, "House-GAN: Relational Generative Adversarial Networks for Graph-constrained House Layout Generation", Computer Vision – ECCV 2020, 162-177, 2020

[4] Bailin Yang; Liuliu Li; Chao Song; Zhaoyi Jiang; Yun Ling, "Automatic Furniture Layout Based on Functional Area Division", 2019 International Conference on Cyberworlds (CW), 2019

[5] Yong Hyun Kim, Syng-Yup Ohn, "HomeGAN: Two stage GAN for enhanced floor plan image generation", 2020 Korea Computer Congress(KCC), P527-529

[6] S. Chaillou. "Master Thesis: AI + Architecture Towards a New Approach". In Harvard University, pages 27-47, 2019

[7] Ramon Elias Weber, Caitlin Mueller, Christoph Reinhart, Automated floorplan generation in architectural design: A review of methods and applications", Automation in Construction, 2022

[8] Miyato, Takeru, et al., "Spectral normalization for generative adversarial networks," International Conference on Learning Representations, 2018

[9] Zhao, Shengyu, et al., "Differentiable augmentation for data-efficient gan training," NeurIPS 2020), 2020

[10] Isola, Phillip, et al., "Image-to-image translation with conditional adversarial networks," in Computer Vision and Pattern Recognition, 2017

[11] A. Radford, L. Metz, and S. Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". arXiv preprint arXiv:1511.06434, 2015

[12] X. Mao, Q. Li, H. Xie, Y. R. Lau, Z. Wang, and S. P.Smolley. "Least squares generative adversarial networks". In ICCV, pages 2813–2821, 2017

[13] V. Calixto, G. Celani, A literature review for space planning optimization using an evolutionary algorithm approach: 1992-2014, in: Siggradi, Blucher Design Proceedings, 2015

[14] J. Rhee, R. Krishnamurti, P. Veloso, J. Rhee, R. Krishnamurti, Multi-agent space planning a literature review (2008-2017), in: J.-H. Lee (Ed.), "Hello, Culture" 18th International Conference, CAAD Futures 2019 Proceedings, Daejeon, Republic of Korea, 2019, ISBN 978-89-89453-05-5

[15] 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 30th IEEE Conference on Computer Vision and Pattern Recognition, 2017

[16] Jia Zheng, Junfei Zhang, Jing Li, Rui Tang, Shenghua Gao, and Zihan Zhou. "Structured3d: A large photorealistic dataset for structured 3d modeling". arXiv preprint arXiv:1908.00222, 2019

[17] Zhengqin Li, Ting-Wei Yu, Shen Sang, Sarah Wang, Sai Bi, Zexiang Xu, Hong-Xing Yu, Kalyan Sunkavalli, Milos Hasan, Ravi Ramamoorthi, et al. "Openrooms: An end-to-end open framework for photorealistic indoor scene datasets." arXiv preprint arXiv:2007.12868, 2020

[18] Huan Fu, Bowen Cai, Lin Gao, Lingxiao Zhang, Jiaming Wang Cao Li, Zengqi Xun, Chengyue Sun, Rongfei Jia, Binqiang Zhao, Hao Zhang, "3D-FRONT: 3D Furnished Rooms with layOuts and semaNTics", Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2021, pp. 10933-10942

[19] Ahti Kalervo, Juha Ylioinas, Markus Häikiö, Antti Karhu, Juho Kannala, "CubiCasa5K: A Dataset and an Improved Multi-Task Model for Floorplan Image Analysis", Image Analysis. SCIA 2019. Lecture Notes in Computer Science, vol 11482