



Suggestive Site Planning with Conditional GAN and Urban GIS Data

Runjia Tian^(✉)

Harvard University Graduate School of Design, 48 Quincy Street, Cambridge, MA 02138, USA
runjia_tian@gsd.harvard.edu

Abstract. In architecture, landscape architecture, and urban design, site planning refers to the organizational process of site layout. A fundamental step for site planning is the design of building layout across the site. This process is hard to automate due to its multi-modal nature: it takes multiple constraints such as street block shape, orientation, program, density, and plantation. The paper proposes a prototypical and extensive framework to generate building footprints as masterplan references for architects, landscape architects, and urban designers by learning from the existing built environment with Artificial Neural Networks. Pix2PixHD Conditional Generative Adversarial Neural Network is used to learn the mapping from a site boundary geometry represented with a pixelized image to that of an image containing building footprint color-coded to various programs. A dataset containing necessary information is collected from open source GIS (Geographic Information System) portals from the city of Boston, wrangled with geospatial analysis libraries in python, trained with the TensorFlow framework. The result is visualized in Rhinoceros and Grasshopper, for generating site plans interactively.

Keywords: Machine learning · Site planning · Generative adversarial network · GIS · Generative landscape design

1 Introduction

In 1984, Kevin Lynch formalized site planning as “the art of arranging buildings and other structures on the land in harmony with each other” [1]. Site planning has been crucial in the design process of architecture, landscape architecture, and urban planning.

Thus, site planning’s primary goal is the generation and organizational process of the site layout. A fundamental step for site planning is the design of building layout across the site.

There have been various attempts over the past years to automate the site process. However, most approaches adopt rule-based generative design algorithms. Such approaches fail infidelity and diversity as this process is hard to automate due to its multi-modal nature: it takes in multiple constraints such as street orientation, program, density, plantation, and, most importantly, block shape.

In a broad sense, the site planning process in architecture, landscape architecture, urban design, and planning pedagogies refers to the organizational stage of the site planning process. It involves the organization of land-use zoning, access, circulation, plantation design, and other procedures. In a narrow sense, site planning could be formalized as

a conditional generation problem solvable with state-of-the-art machine learning models such as Conditional Generative Adversarial Neural Networks (CGAN).

With the abundance of Geographic Information System (GIS) data, the existing urban environment proves a practical dataset for modelling and automating this design process with the statistical learning approach.

2 Related Works

In 2014, Ian Goodfellow proposed the structure of the Generative Adversarial Network [2]. In the paper, he proposed using two multilayer perceptron, a generator, and a discriminator to generate data and classify as true or false based on training data. In November 2014, Mehdi Mirza and Simon Osindero proposed Conditional GAN (CGAN) to place labels on training data for both generator and discriminator in a GAN structure [3]. In their work, they demonstrate how this approach can generate descriptive tags that are not part of training labels.

Isola et al. proposed image-to-image translation with their model Pix2Pix based on Conditional GAN in November 2018 [4] (Fig. 1).

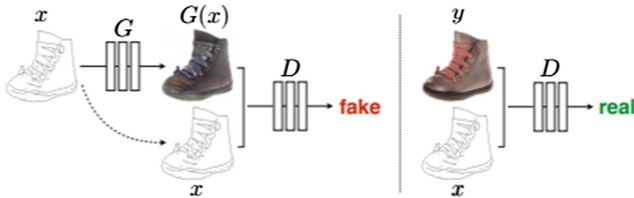


Fig. 1. Pix2Pix image translation, Isola et al.

The Pix2Pix model offers accessible interfaces to designers and has been proved useful in multiple scenarios, including generating architecture plan layouts and texture filling for sketches.

In 2019, Stanislas Chaillou proposed ArchiGAN: A Generative Stack for Apartment Building Design in his thesis [5]. Stanislas unpacked floor plan design into three steps: building footprint massing, program repartition, and furniture layout, and used a Pix2Pix GAN-model for each step for the three tasks correspondingly. The thesis of Stanislas laid a foundation for the generation of building footprint based on the parcel shape. However, his thesis is limited to apartment plan generation and does not incorporate into the complicated urban and landscape context of architecture design.

3 Methodology

The synthetic workflow of the project adopts the typical framework of a data science process. The core question for suggestive planning creative design tool can be divided into three steps (Fig. 2):

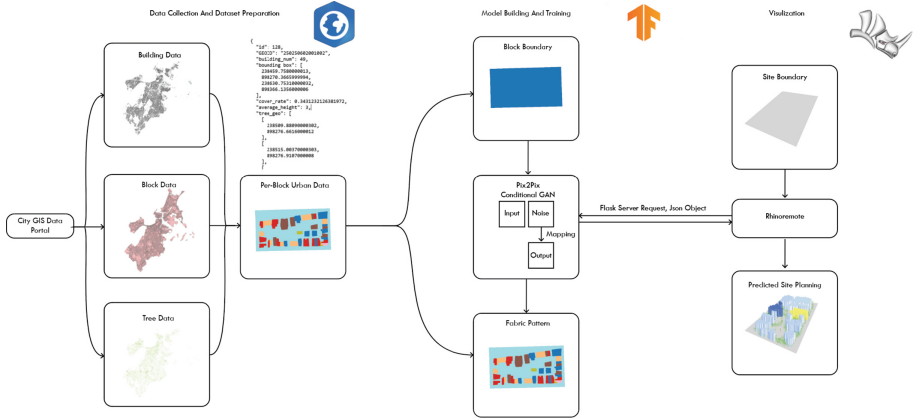


Fig. 2. Workflow diagram

- 1. Data Acquisition and Feature Engineering:** A dataset containing necessary information is collected from open source GIS portals. The dataset was processed with geospatial analytical tools for standardizing and canonicalizing. The dataset of vector geospatial data is eventually rasterized into an image dataset for deep learning.
- 2. Machine Learning:** Pix2PixHD Conditional GAN is used in this process to learn the mapping from a site boundary geometry represented with a pixelized image, to that of an image containing building footprint color-coded to various programs, trained with TensorFlow framework.
- 3. Visualization:** The result is visualized in three-dimensional computer-aided design software with computational design access for generating site planning proposals interactively in real-time.

3.1 Data Acquisition and Feature Engineering

The data is obtained through various open-source GIS Databases. Therefore, the feature engineering and exploratory data analysis were crucial for this project as a spatial join is necessary to canonicalize and standardize the data across various data sources.

3.2 Machine Learning

The model uses the Pix2Pix pipeline for building reliable site boundary image to site planning image translation.

The Pix2Pix model consists of two parts, a generator, and a discriminator. The condition is concatenated with the Gaussian noise as input to the generator and is concatenated again with generator output as discriminator input (Fig. 3) [6]. The model's objective function is the sum of the GAN loss, a binary cross-entropy, and an $L1$ norm between the generated image and the ground truth.

For our research, the site boundary's input image is represented as a 256-pixel by 256-pixel image; the ground truth image is the same site with buildings color-coded

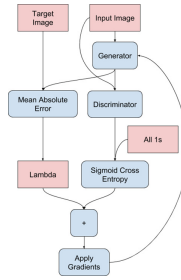


Fig. 3. Pix2Pix architecture

according to specific attributes. For this research, we use height data as a showcase, and similar approaches could be applied for other attributes.

3.3 Visualization

Specifically, the research utilizes Rhinoceros and Grasshopper as a platform for visualizing the result. The site boundary is represented as a closed curve in Rhinoceros. The Grasshopper platform rasterizes the curve geometry and sends the data as a raster image to the deep learning model. Flask framework is utilized for creating the inter-process communication between deep learning models and modelling software. The predicted site layout is then sent back to Grasshopper and visualized in Rhinoceros.

4 Case Study: Taking Boston as Example

The proposed workflow was tested in a course the author participated at Massachusetts Institute of Technology.¹ The objective of the course is to introduce foundational concepts and methodology of machine learning and its applications to design. The project was supported by industry partner Spacemaker.

4.1 Data Acquisition and Feature Engineering

The data for training the generative model is obtained through Boston GIS Data Open Portal, an open-source GIS Database. The author preprocessed GIS data with ArcGIS Pro. The raw GIS data was sparse and dirty. Moreover, it was hard to acquire a dataset that included building information and parcel information for Boston.

4.1.1 Data Acquisition

We acquire Boston Building GIS data [7], Boston Tree GIS [8] and data Massachusetts Street Block GIS data [9] from the public data portal of the City of Boston and Massachusetts State (Fig. 4). The GIS data is stored in .shp files and preprocessed with

¹ 4.453/4.S48 Creative Machine Learning for Design, Massachusetts Institute of Technology, Cambridge MA, U.S., 2020. Instructor: Caitlin Mueller, Renaud Danhaive.

ArcGIS Pro before performing any further analysis. Each object in the dataset has a .geojson geometry that indicates the geographical location for the object stored in polygon or point form. Geographical projection is canonicalized for spatial merging in the next steps.

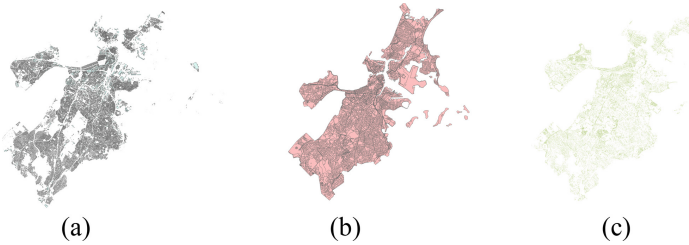


Fig. 4. (a) 90,308 building data acquired from Source [7] (b) 8,841 Street Block Data Acquired from Source [8] (c) 201,845 Tree Data Acquired from Source [9]

4.1.2 Data Merging and Wrangling in ArcGIS

The data wrangling consists of two parts: spatial joining and feature engineering. The spatial joining is accomplished in the ArcGIS Pro environment to speed up the geospatial data merging algorithm (Fig. 5). Feature engineering is accomplished in Anaconda environment with python geospatial analysis libraries include Geopandas. Then we split the dataset by parcel data so that we could bag the building to the closest street parcel that contains the building geometry.

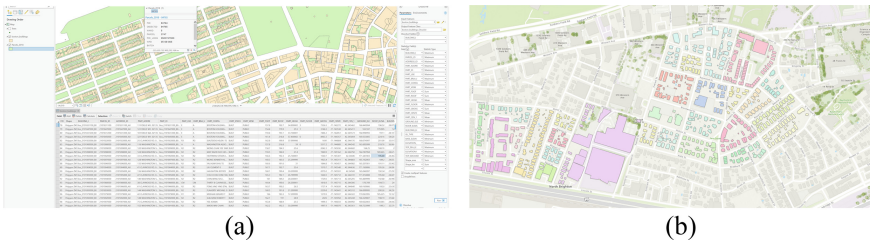


Fig. 5. Spatial joining and profile merging in ArcGIS Pro

After we preprocessed the data, the building data frame and tree data frame, stored as .geojson objects, are spatially joined to the closest parcel that contains such objects. These files are ready for processing in python environments.

4.1.3 Feature Engineering

After this step, buildings within the same parcel are stored in the same tabular data frame object while maintaining the statistical numeric features such as elevation, building

height, and land price. Feature engineering is accomplished in anaconda to calculate important design metrics such as coverage rate, plantation rate, and floor area ratio (Fig. 6).

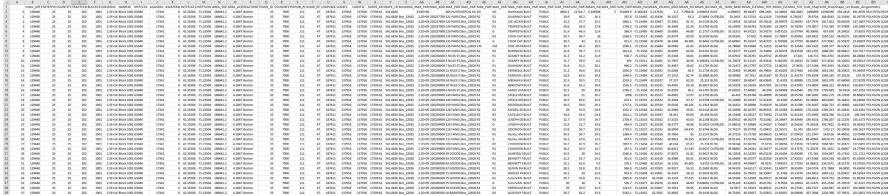


















Fig. 6. Example data frame

After feature engineering, we can easily perform spatial query such as spatial join, which overlay two geospatial feature on top of each other according to selection metrics. The joined metric is stored in shapely geometry format, and we will further work with the geometry file parsed and aggregated this step as our cornerstone. For the program, each building in the site is color coded with a particular RGB value according to the categorical programs feature in the dataset. We use Matplotlib Tab 20 Colormap to maximize differences between various categories (Table 1).

Table 1. City of Boston GIS data land use code and dataset color coding

Use Code	Description	RGB triplet	Color
R1	Residential 1 Family	(44, 160, 44)	
R2	Residential 2 Family	(148, 103, 189)	
R3	Residential 3 Family	(227, 119, 194)	
R4	Residential 4-6 Family	(255, 127, 14)	
A	Residential 7 or More Units	(255, 187, 120)	
RL	Residential Lot	(152, 223, 138)	
CD	Condominium	(214, 39, 40)	
CC	Commercial Condominium	(255, 152, 150)	
CM	Condo Main	(31, 119, 180)	
C	Commercial	(140, 86, 75)	
RC	Mixed Residential Commercial	(196, 156, 148)	
CL	Commercial Land	(23, 190, 107)	
CP	Condo Parking	(199, 199, 199)	
I	Industrial	(127, 127, 127)	
E	Exempt	(188, 189, 34)	
Background	Background for the Site	(158, 218, 229)	

With the color-coding technique, we rasterize the GIS dataset into an image dataset ready for deep learning (Fig. 7). This research used the program matrix as a show case for the research, as similar approaches could be applied to other data features such as height, price and plantation.

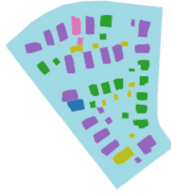


Fig. 7. Example data plotting

4.2 Model Building and Training

To prepare a dataset ready for Pix2Pix training, we need to prepare a dataset composed of training images and predicted images laid side-by-side (Fig. 8).



Fig. 8. Dataset samples

The model is trained in TensorFlow 2.1.0 environment with NVIDIA GeForce RTX 2080 Ti GPU acceleration. The training data consists of 4400 pairs of conditions and ground-truth site plans as training data and 400 validation data (Fig. 9).

4.3 Results and Visualization

The immediate result of the model is visualized in two-dimensional space as a masterplan generation in 2D (Fig. 10). As the training progress, the fidelity of the generation model gradually improves. However, a tendency of bias towards condo type is observed. This could be caused by the unbalanced number across types in the GIS Dataset.

Meanwhile, the color-coding for various program might also cause the bias in the generator. The generated site plans are blobby. This might be caused because various pattern of urban fabric coexist in the dataset, the density of urban setting could vary drastically from one area to another.

4.3.1 Visualization in Rhinoceros and Grasshopper

To further test and visualize the model, we use GH Python Remote [10] and flask server to request prediction and transmit data between the python server and the Rhinoceros-Grasshopper Visualization environment (Fig. 11).

For the current research, the extrusion height for building geometry uses the generic average building height of dataset, which is 16 m. In future work the height feature could be implemented using similar methodologies.

Various Site boundary shape is tested for site planning generation. The model performs best for sites with quadrilateral boundary shape, which might be a result because most city blocks in Boston are quadrilateral shapes. In future work, this aspect might be further enhanced with data augmentation.

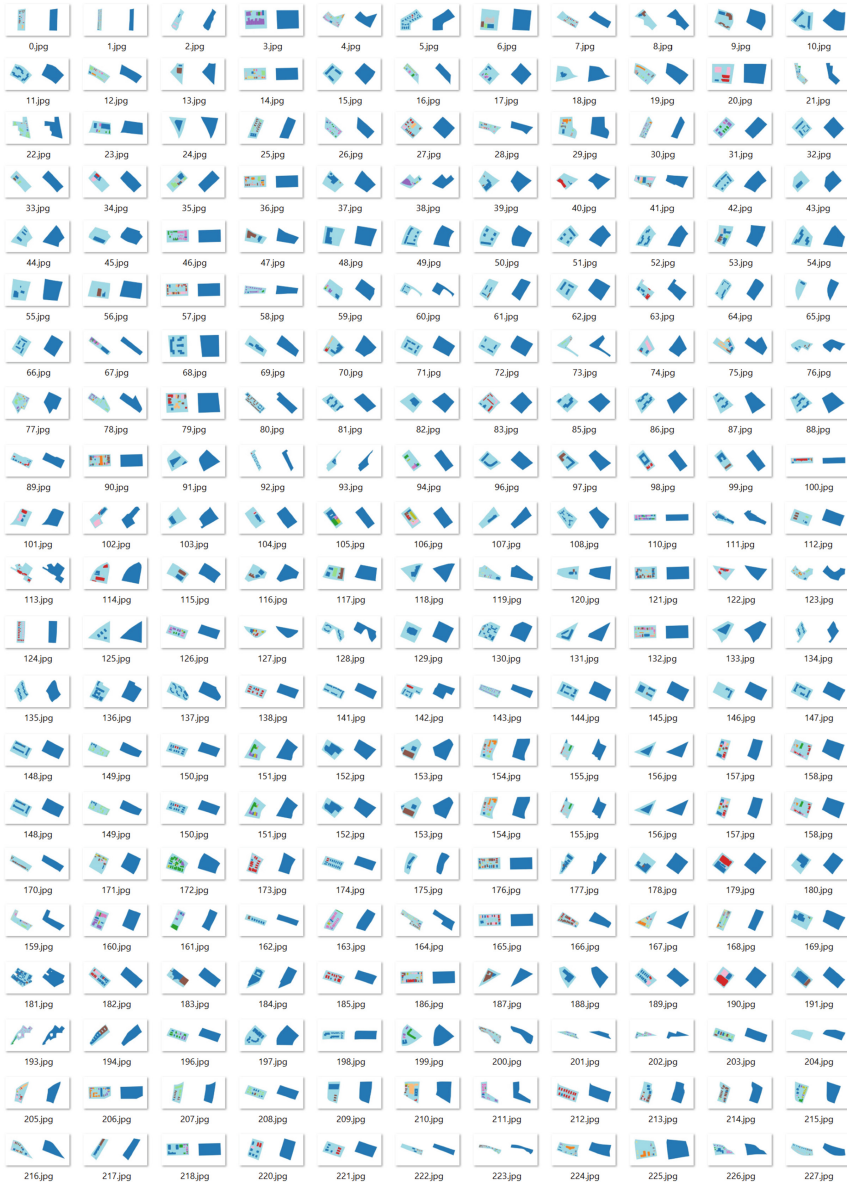


Fig. 9. Training data batch example

4.3.2 Extended Usage: City Style Transfer

Generator essentially learns the mapping of a specific urban fabric pattern of a particular city in our case. Extensions of the model could be applied for urban analytic applications.

One extension is to develop a “city style-transfer” interface. The interface takes the urban parcel as input and output urban fabric. Thus we will be able to test how

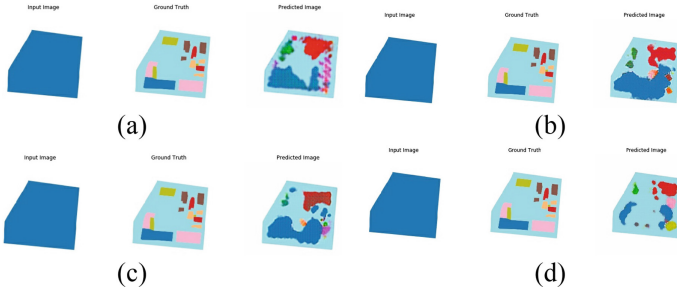


Fig. 10. Model generation after (a) 1 Epoch (b) 50 Epochs (c) 150 Epochs (d) 200 Epochs

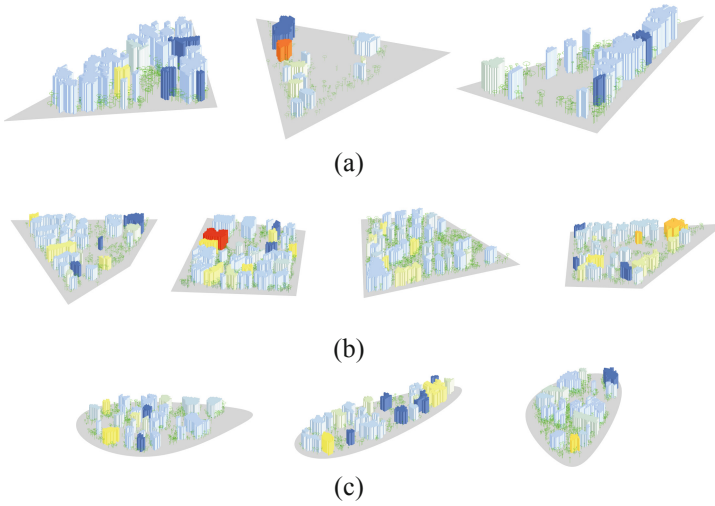


Fig. 11. Site planning suggestion for (a) Triangle shape site boundary (b) Quadrangle shape site boundary (c) Spline shape site boundary

the urban fabric of one city could be overlaid and juxtaposed onto another city. For demonstration, the generator trained on the GIS data of Boston is applied to the lower Manhattan of New York City (Fig. 12).

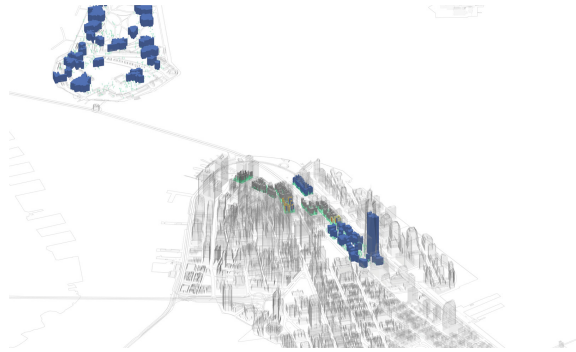


Fig. 12. City style transfer overlaying boston building footprint to lower Manhattan

5 Summary

The innovation for this project is that it proposes a generic workflow for building and fitting deep generative model that generate suggestive site planning in an accessible way. The toolkit could be utilized both in the context of site planning but for morphological analysis for city's urban fabric. Encoder-decoder network could be realized with transfer learning to the model's latent space. Here we provide some potential impact of the model and potential future improvements.

The project might have great impact for urban planners and urban designers, especially during the stage of concept design and massing studies. The project might also impact real estate industry much to a similar impact to urban designers. Real estate developers could use the tool for fast-prototyping design iterations at the very early stage of a real state financing project that involves complex site planning.

There are still several aspects that this model could be further improved. With the extensible framework, multiple GAN models could be trained to learn the mapping about multiple modalities of the data, and thus provides more comprehensive generation of building data.

Another possible improvement is the optimization of network structure. With structural Neural Networks and inference methods, we can achieve multimodal generation using a combination of conditional autoencoders and generative adversarial network such as CVAE-GANs.

Acknowledgements. The author would like to express great appreciation to the Professor Catlin Muller and Dr. Renaud Danhaive for their contribution in teaching. The also would also like to express deep gratitude to Ms. Karoline Skatteboe from Spacemaker who supported the research and offered insightful advice and feedback.

References

1. Lynch, K., Hack, G.: Site Planning, 3rd edn. MIT Press, Cambridge (1984)

2. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Bengio, Y.: Generative Adversarial Networks (2014). ArXiv.org, 10 June 2014
3. Mirza, M., Osindero, S.: Conditional Generative Adversarial Nets (2014)
4. Isola, P., Jun-Yan, Z., Zhou, T., Efros, A.: Image-to-Image Translation with Conditional Adversarial Networks (2017). ArXiv.org, 22 Nov 2017
5. Chaillou, S., Witt, A.: Harvard University. Graduate School of Design. Architecture for the 90%. The Iceberg (2019). http://stanislaschaillou.com/thesis/GAN/unit_program/
6. Pix2Pix Tutorial with Tensorflow. <https://www.tensorflow.org/tutorials/generative/pix2pix>
7. Boston Buildings - ArcGIS Online. <https://www.arcgis.com/home/item.html?id=c423eda7a64b49c98a9ebdf5a6b7e135>
8. MassGIS Data: Building Structures. <https://data.boston.gov/dataset/trees>
9. MassGIS Data: Trees. <https://docs.digital.mass.gov/dataset/massgis-data-datalayers-2010-us-census>
10. Pierre, C.: Massachusetts Institute of Technology, GH Python Remote. Food4Rhino (2017). <https://www.food4rhino.com/app/gh-python-remote>

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

