MSc thesis in Geomatics

Building massing generation using ***GAN*** trained on Dutch 3D city models

Ondřej Veselý

2022

TU Delft

MSc thesis in Geomatics

Building massing generation using GAN trained on Dutch 3D city models

Ondřej Veselý

June 2022

TU Delft

The work in this thesis was carried out in the:

3D geoinformation group
Delft University of Technology

Supervisors:	Dr. Giorgio Agugiaro
	Dr.ir. Roberto Cavallo
Co-reader:	Jun.-Prof. Dr. Reinhard Koenig

Abstract

Despite being relatively novel, generative adversarial networks (*GAN*) have already been appropriated for application to several problems within the field of architectural and urban generative design. However, the preceding *GAN* based models for building massing generation make use of only simplified and two dimensional representation of the built environment.

This work improves upon the existing deep-learning-based methods for generation of building massings and building group layouts, by fusing high accuracy three-dimensional building models with site context derived from cadastral and topographic data, sourced from openly available datasets in the Netherlands. *Pix2pixGAN* implementation in *PyTorch*, trained on existing massing data encoded into images as heightmaps, is used to generate building massing geometry. Two methods for geometry extraction from heightmaps are introduced, voxelization and vectorization. The goal for the model is to maximize similarity of morphological traits of configurations generated by the model to the ground truth training data. The effects of multiple proposed training configurations on the resulting massings generated by the model are evaluated, together with visual assessment, using their *Spacematrix* mappings.

Three distinct models with specific goals are presented - parcel infill model, street block infill model, and urban fabric infill model. All three models show a capability to learn spatial traits of existing building configurations and transfer them into new situations not encountered in the training data, which is confirmed by the distribution of *Spacematrix* mapping of the generated results being similar to the distributions of the ground truth data.

The proposed methodology represents a novel approach to generating building massing configurations by autonomously inferring the rules of their composition from existing urban areas. The resulting models could be used to provide initial states in optimization-driven design approaches, or as smart massing suggestion engines, assisting architects and city planners during the early building design process.

Acknowledgements

First I would like to thank both my mentors — Giorgio Agugiaro and Roberto Cavallo — for giving me a chance to work on this research topic as my graduation project. I appreciated your constant support, guidance, and confidence. Your multidisciplinary perspective and critical feedback ensured that I stayed on track and continuously refined my arguments during the development of this thesis.

Special thanks belongs to my old mentor and co-reader — Reinhard Koenig — for his comments and most importantly for helping me discover my passion for computational design. Without your support I would probably never reach this point in my academic career.

I would also like to mention my mentors at City Inteligence Lab — Angelos Chronis and Theodore Galanos — to both both of you I give my gratitude for inspiring me to develop this work.

Finally, many, many thanks to my co-workers, friends, family, and — especially — İdil Gümrük for your support and patience with me during this difficult journey.

Contents

List of Figures

List of Tables

Acronyms

[1] 3D Register of Buildings and Addresses
[2] Current Elevation Data of Netherlands
[3] Register of Buildings and Addresses of the Netherlands
[4] Large Scale Topographic Map of the Netherlands
[5] Cadastral Register of the Netherlands
[6] National Roads Database of the Netherlands

"The question of whether a computer can think is no more interesting than the question of whether a submarine can swim."

Edsger W. Dijkstra

1 Introduction

1.1 Motivation

Context The proliferation of reliance on computers for the design and engineering of the built environment, and the increasing complexity of applying digital tools effectively to the design process gave rise to a new kind of design and consultancy practices. Companies where *"developers, and computer scientists work along with designers and engineers to offer ... bespoke digital solutions for architecture and the construction industry"* [Carta, 2021].

Under this new design practice paradigm, the increasing speed and reliability of design performance simulation models makes it trivial and therefore imperative to evaluate designs based on various technical, environmental, and socio-economic factors already from the initial stages of the design process [Caetano et al., 2020]. With improved ability to predict and evaluate the performance of any given design variant, generating and exploring large, and more importantly diverse, sets of variants becomes the priority goal of the early design phases.

Problem statement The development of computational design methods within the field architectural design has already brought to the market a multitude of digital toolkits and software packages focused on quickly generating variants of residential, office, hospital, or other building layouts based on a set of user-defined parameters (*e.g. TestFit*[1], *Kreo*[2], *SpaceMaker*[3]). These design toolkits use generative algorithms to offer large amounts of design options and evaluate them on a broad range of metrics, providing a range of options in the internal layout of the building.

However, when it comes to generation of building massing, the variety of solutions offered by these tools is much lower. Often just a limited collection of basic building shape archetypes and configurations is offered, which are then simply conformed to the shape of the plot. The general site context, such as the surrounding building forms, access routes or landscape features, are only considered during the evaluation of the variant, not its generation.

I aim to investigate whether we can build a deep-learning-based model capable of suggesting more diverse building massing designs, trained on previous design solutions present within available 3D city models. By defining the geometrical and semantic properties of the desired massing design *and* the site context it is within, the challenge I want to tackle with this work is how to synthesize new design options by inferring the solution from selected buildings and sites already present within the existing built environment in the Netherlands.

[1] https://testfit.io/
[2] https://www.kreo.net/
[3] https://www.spacemakerai.com/

Approach I propose a method based on the picture to picture translating generative adversarial network (Pix2Pix GAN) [Isola et al., 2017]. Pix2Pix GAN, as a type of deep-learning-based model for image to image translation, has already shown promising performance in similar applications. It is a neural network framework capable of learning complex relations between visually represented input (i.e. site plan) and output (i.e. building massing representation) from paired images within a training dataset.

Although a concept of a generative adversarial network (GAN) [Goodfellow et al., 2014] is already fairly well established in the field of deep-learning, its application to generation of architectural and urban designs is still a fairly novel idea. The first experiments of applying GANs to architecture and built environment related data gained traction with the introduction of the Pix2Pix GAN architecture in 2017, which allowed for training on picture base datasets in a way more approachable to designers or artists [Isola et al., 2017].

Previous work While 4 year ago machine learning methods still had yet to make a significant impact in the field of architecture and design [Khean et al., 2018], since advanced models like Pix2Pix GAN became readily available, multiple works in built environment design related fields explored their potential. The methods based on the Pix2Pix GAN framework have been applied to the design of residential housing units — their shape, internal structure, functional zoning, and furniture placement [Chaillou, 2020], or to topological layout of office buildings [Chang et al., 2021].

In the field of urban design, Pix2Pix GAN has been used to assess environmental performance of building groups on a scale of city block from the perspective of sunlight access and wind flow [Chronis et al., 2020; Mokhtar et al., 2020]. Pix2Pix GANs have been proposed as a support tool for planners, architects or developers by giving them insight about the possible solutions for building composition of a city, urban blocks and building groups [Fedorova, 2021; Yao et al., 2021; Tian, 2020].

The previous experiments with utilizing GANs for the purpose of generating building layouts show the potential of these powerful models. However, the work published to this date is reliant mostly on two dimensional representation of buildings as their footprints. I propose an improved method for training GAN models on 3D representations of building forms extracted from a complete 3D model data of Dutch building stock - the *3D BAG* dataset [3D Geoinformation research group, 2021].

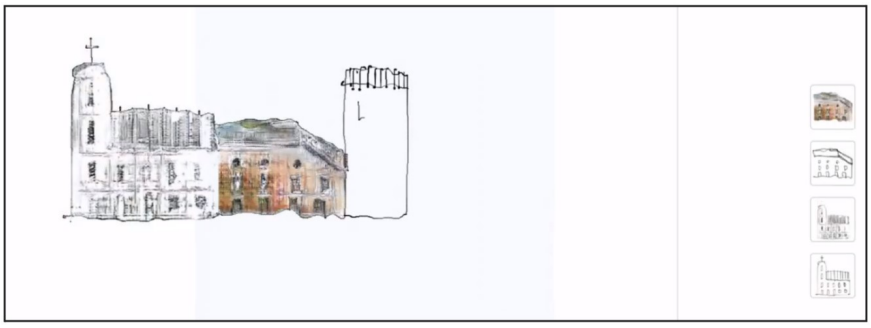


Figure 1.1: Sketcherbot uses Pix2Pix GAN trained on hand-sketches of trees and urban scenes to fill in designers sketches with detail (figure taken from [Alonso, 2017])

1.2 Research questions

The main research question for this thesis is:

> To what extent can one assist the process of building massing *(and by extension the built environment)* design generation by utilizing GAN model trained on existing building forms represented in Dutch buildin stock?

To explore this question, the following sub-questions are therefore relevant:

- At what scales is such model applicable (eg. single building, building group, street block)? Is a single general model sufficient, or does one need to train a separate dedicated model for each scale?
- Is it possible to steer the properties of the generated designs towards certain traits by curating the data the model learns from? What metrics should one use to measure these properties? How strong is the correlation between measurable properties of the designs in the training data and the generated results?
- How important is the level of detail (LoD) of the training data and what aspects of site context have to be included in the training data for the quality of the results?

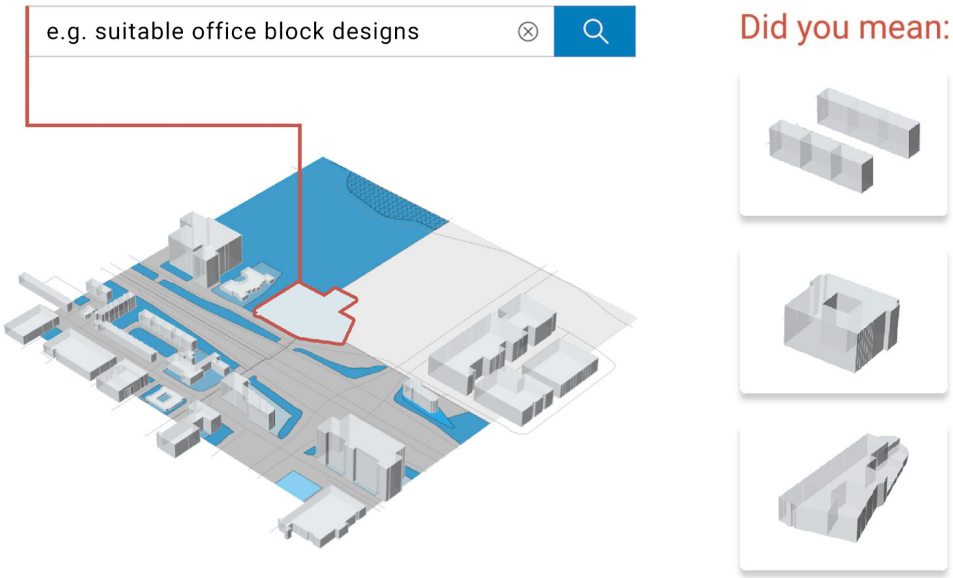


Figure 1.2: Concept sketch of the research goals in a nutshell - a building massing design suggestion engine

1.3 Scope and objectives

Scope The scope of my work is focused on training a GAN for generation of building and building group massing designs utilizing the updated version of 3D Basisregistratie Adressen en Gebouwen [1] (3D BAG) dataset (released in 2021). This open dataset containins highly accurate building models of almost all buildings in the Netherlands [3D Geoinformation research group, 2021]. My proposed GAN is trained using image pairs generated from the 3D BAG, cadastral data, road network data, topographic maps and other openly available datasets covering the built environment of the Netherlands.

An existing implementation [Zhu, 2017b] of Pix2Pix GAN [Isola et al., 2017] — implemented in PyTorch [Paszke et al., 2019] — is the primary GAN model used for the purpose of this research, with an option of exploring other similar GAN models for the sake of performance comparison as a possible additional experiment. The focus of this thesis is on exploring the applicability of existing state-of-the-art GAN frameworks and reimplementing significant parts of specific GAN architecture from scratch is not a part of this research. Scope of this work in relation to these existing GAN models is defined instead as finding the possible combination of training data and training hyperparameters to be used with the existing GAN implementations to help me explore and answer the posed research questions.

Goals From my review of existing applications of similar methods to related topics, I conclude that the current state-of-the-art research on utilizing GAN for generation of building or building group designs still has some shortcomings. The models are often trained on simplified building morphology, which omits many details of the general building form and roof-scape. The GAN methods are applied only on two dimensional representation of the build environment or lack the methods to convert the output raster imagery image back into 3D geometry. A rigorous study of quantitative and qualitative performance of the generated designs and how they relate to the training data are missing.

My goal for this thesis is to develop an improved method for training and evaluating GAN frameworks on the task of generation of building massing designs and compare performance of various approaches to how one defines the goals of the model and creates the training datasets. Ultimately I want to be able to discuss the usability of the variants generated by such models in the practice of design and engineering of the built environment.

1.4 Scientific relevance

By generating building massing suggestions based on rules and features inferred from existing solutions, instead of the user having to manually define and input these rules themselves in the form of a fixed algorithm, I hope to extend the possibilities for application of generative building design tools. The suggestions generated by the model could be used as initial states for further optimization and exploration. Using GANs to generate a larger and more diverse set of starting states for optimization algorithms to iterate over has been proposed as a method for avoiding finding local minima early in the optimization process [Chen et al., 2019].

[1] 3D Register of Buildings and Addresses

The proposed framework for suggesting building massing solutions is relevant outside of the domain of computational and generative building design as well. One of popular applications for picture to picture translating GAN models is using them as a base for suggestive drawing applications, where sketched input of the human user is completed and filled in by the generator network. This method has been explored on various drawing types including sketches of building elevations (see Figure 1.1) [Alonso, 2017] and floor plans [Swahn, 2019].

In this fashion, a generative model capable of suggesting building massing variations could assist architects, developers and city planners in design exploration in early architectural project stages when sketching out initial massing options (*SO - schetsontwerp phase* of Dutch construction design documentation process). Urban planners and municipalities could use the tool to explore and quickly evaluate densification potential of existing urban areas and use the tool to generate options for building infill scenarios within existing urban structures.

2 Theoretical backround

In this chapter I will elaborate upon the theoretical concepts introduced in the Chapter 1 Introduction to ensure basic understanding of the terminology used before I connect them to the methods proposed for the data collection, processing, training, post processing and evaluation in next chapters.

Since the work heavily relies on utilizing GAN, in particular the Pix2Pix GAN architecture and its derivatives, a brief introduction to the principles and theory behind it is provided to better understand the motivations upon which the further steps of my work are based. Additionally, I will introduce the concept of level of detail (LoD).

Finally, I will provide an overview of the related works on the topic of architectural and urban design generation using GAN models and provide my own conclusions on the limitations of existing research.

2.1 Generative adversarial network

GAN, as first described by Goodfellow et al. [2014], is a machine learning framework consisting of two mutually competing convolutional neural networks (CNNs) - generative network (Generator) and discriminative network (Discriminator). Both networks are trained using a collection of training data, often based on some real-world dataset.

The goal of the Generator is to synthesize new data samples with similar properties to the samples in the training dataset. The goal of Discriminator is to identify whether a data sample is a *fake* generated by the Generator, or a real-data sample from the training dataset. This zero-sum game of two competing networks creates a training feedback loop. As the Generator improves at generating *fake* outputs, the Discriminator has to improve at identifying them. I find a metaphor of GAN training as a game between the Generator as an criminal and Discriminator as a detective investigating the forgery case useful when explaing this basic principle (see Figure 2.1).

Pix2Pix GAN [Isola et al., 2017] applies this principle to images data and, as a type of a supervised conditional acgan, it can learn the relationship between pre-organized pairs of images. *Supervised* in this case means that the training data has to be preprocessed and organized into pairs of images and the model tries to estimate the general function that maps the content of image A to the content of image B.

'Conditional' in this case means that both Generator and Discriminator have access to the input A image. Therefore the Discriminator can evaluate not only how *real* the output image looks, but also how well it maps from the original input image. Both Generator and Discriminator as a type of CNNs can achieve a great level of accuracy as their internal data labeling and transformation can happen at the scale of individual pixels [LeCun et al., 2015].

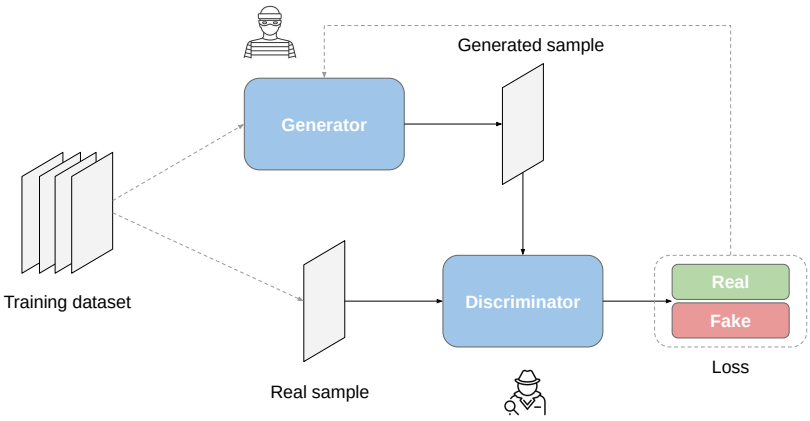


Figure 2.1: Basic principle of GAN

Pix2Pix GAN can only generate deterministic mapping from picture A to B, meaning that after training single image input will always yield the same single result. An extension of the original architecture was proposed and developed as BicycleGAN [Zhu et al., 2017]. The training still requires paired data, but after training, it enables mapping one input to multiple possible outputs. This is achieved by injecting a z-vector (noise) into the input layer. By manipulating the z-vector in small increments, it is possible to create gradual transitions between multiple output mappings (see Figure 2.2).

Figure 2.2: Example of BicycleGAN mapping single evening landscape into multiple daytime versions [Zhu, 2017a]

In the context of this work, I intend to use Pix2Pix GANs or BicycleGAN to map a set of site conditions in the input image to building massing plausible on the given site in the output image. The main objective of the model is to learn the rules guiding the placement and form of building massings in urban context from the training data and to generate new configurations based on rules inferred. To utilize these picture to picture translating models for this purpose, I need to generate training raster images by compressing both the site information and the building massing into images containing their graphical representation — similar in principle to architectural site plans (see Figure 2.3).

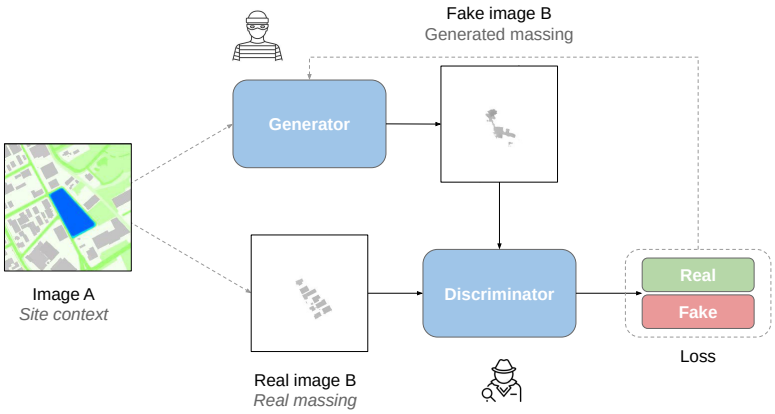


Figure 2.3: Proposed GAN model

2.2 Level of detail

One of my the proposed improvements compared to the previous applications of GANs to generation of building massings, at least the ones covered as part of the literature study (see next Section 2.3 Evaluation), is to use more detailed three dimensional building models to generate the training data for the model. When describing the level of accuracy of the building geometry modelled, I will use the term established withing the field of Geomatics — level of detail (LoD).

Definition To cover the LoDs of building models accurately, one can refer to CityGML 2.0 based classification of LoDs, or in particular the extended classification from Biljecki et al. [2016] (see Figure 2.4). Under this classification the LoDs are grouped into levels ranging from 0 to 4, with increasingly complex geometrical primitives used to describe increasingly complex features of the building. The expanded model adds further sub-levels to each of the 4 main LoDs. This reflects the fact that within one LoD, as defined by the CityGML standard, a building model can use the same rules and geometrical primitives while being modeled with various levels of accuracy, depending on specific implementation.

Based on the CityGML 2.0 standard, the extended definition by Biljecki et al. [2016] and how the terms are applied in common practice, I can summarize the LoDs as following;

LoD 0.1 – 0.3 Models buildings only as a planar polygonal geometry. In practice, it describes only the footprint (or the roof edge) of a building

LoD 1.1 – 1.3 Models buildings as extrusion-based prismatic geometry. In practice, it describes the building as a volume enclosed by a solid geometry obtained by extruding the footprint(s) by a single height value associated with each footprint.

LoD 2.1 – 2.3 Models building as complex surface or solid geometry types, with optional semantic properties attached to geometrical parts. In practice, it describes the approximation of the external envelope of the building.

LoD 3.1 – 3.3 Uses the same data model as LoD 2. Besides the geometry of the building's envelope, it also models other external features of the building, such as openings, external equipment and various facade features.

LoD 4 Uses the same data model as LoD 2. In theory LoD4 stores not only the building's external features, but interior features as well. But is rarely used in practice and therefore omitted from the extended definition by Biljecki et al. [2016].

LoDs in practice To put things into perspective, one of the more popular datasets to source information on building geometry from is OpenStreetMap (OSM). OSM is an open source global geographic dataset that optionally includes height information for the building footprints stored within it. Only a single height value can be associated with each single building. Therefore one can only talk about OSM data as a source of in bst case LoD 1.1 building models. In practice this height information is in many cases missing, or it is just inferred instead of being accurately measured for each building (see Figure 2.5).

While higher LoD 3D city models exist, they created on scale of individual cities, and in almost all cases at least parital rely on manual modelling of the individual buildings. A recent improvement in the context of Netherlands in this regard is the 3D BAG dataset [3D Geoinformation research group, 2021]. 3D BAG is an automatically generated collection of 3D building geometries for almost all buildings in Netherlands, modelled at multiple LoDs up to LoD 2.2 (see Figure 2.6). Therefore, I will be using 3D BAG as the main source of building geometry for the purpose of this research.

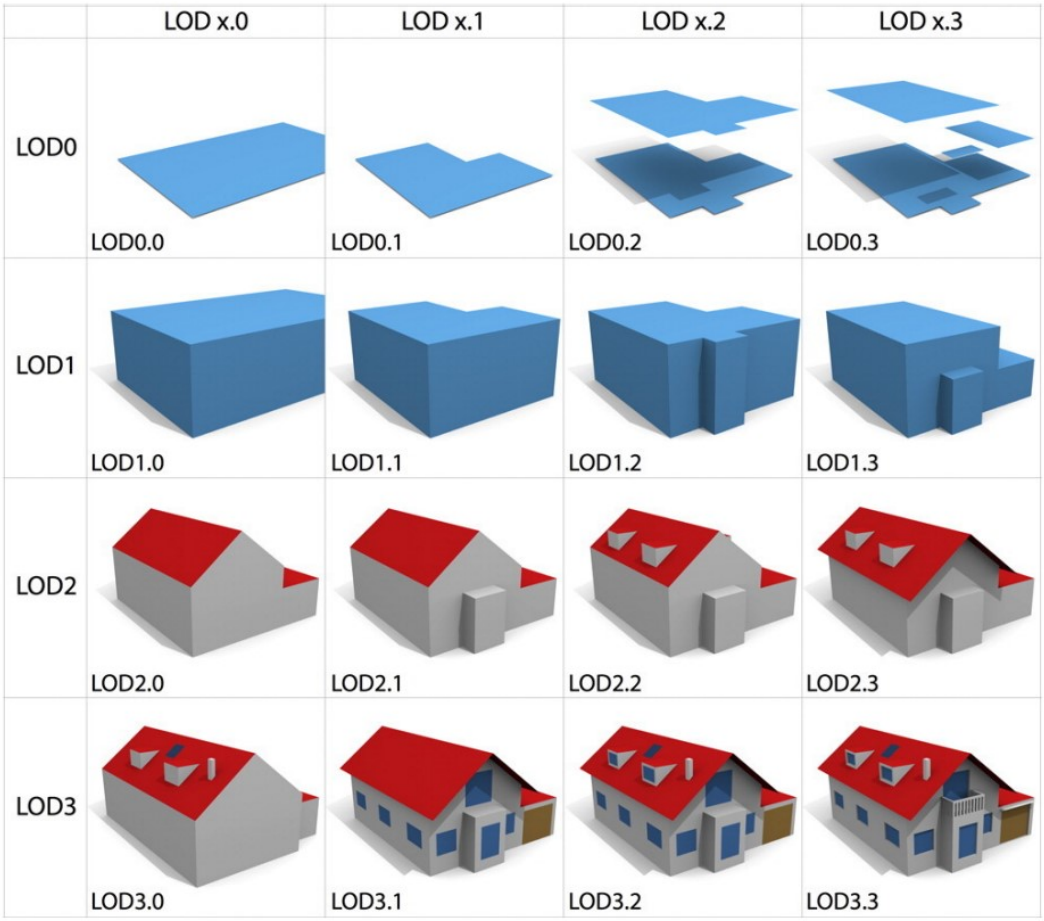


Figure 2.4: Improved LOD specification for 3D building models by Biljecki et al. [2016]

Figure 2.5: Mapbox web-maps use OSM data as source of its LoD 1.1 building models. In this screenshot, many building heights in center of Amsterdam are missing completely

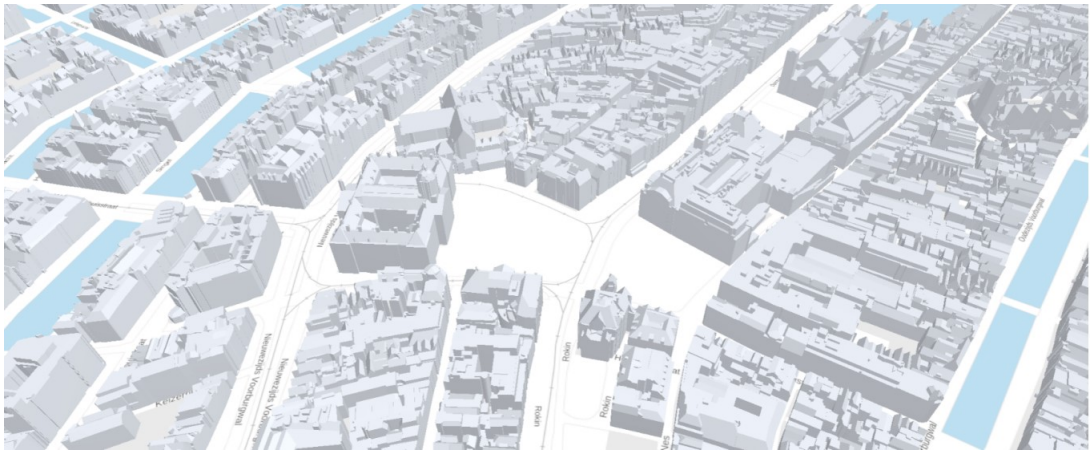


Figure 2.6: 3D BAG LoD 2.2 building models for the same area of Amsterdam as in fig. X. visualized in 3D BAG viewer

2.3 Related work

pix2pix GAN

Although the idea of the GAN was already introduced a few years prior by Goodfellow et al. [2014], the Pix2Pix GAN [Isola et al., 2017] introduced a novel architecture for using the GAN as a general-purpose model for image-to-image translation problems.

What the use of GAN architecture brought as a novelty into the field of deep-learning based image translation and generation, is that the model not only learns the general function for mapping from image A to image B, but also learns how to evaluate the quality of the results (which was previously a challenge in image generation).

The research demonstrated that this new approach is effective at synthesizing photos from label maps, reconstructing photorealistic representation of objects from sketches or colorizing images, among other tasks. Since the release of that paper, numerous artists and designers have tried to create their own experiments with the system, proving its ease of use.

A limitation of using Pix2Pix GAN in practice, similar as with many other architectures based around types of CNNs, is that it often requires large amounts of data with few outliers to produce accurate results [LeCun et al., 2015], with recommended dataset sizes approaching 10,000 image pairs.

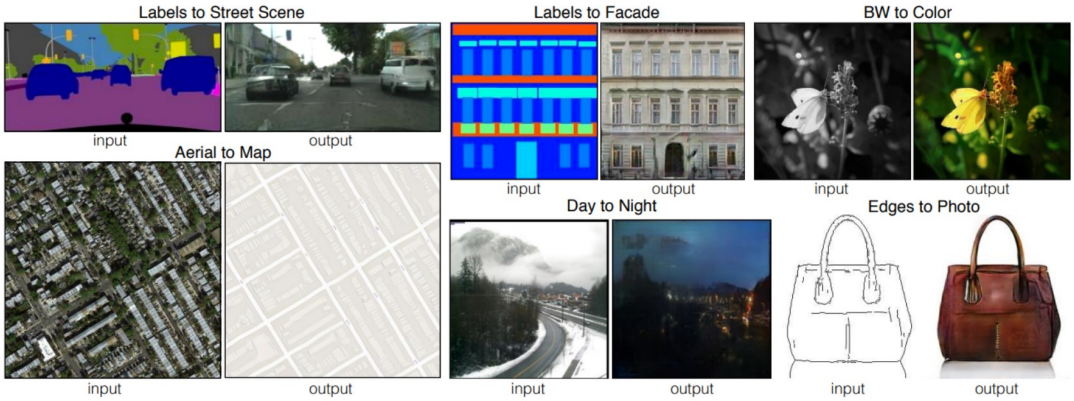


Figure 2.7: Results of the pix2pix GAN model applied to several picture to picture mapping problems [Isola et al., 2017]

ArchiGAN: a Generative Stack for Apartment Building Design

ArchiGAN [Chaillou, 2020] represents a thorough experiment on the feasibility of using Pix2Pix GANs for design of residential unit floor plans. The author proposes a stack of multiple pix2pixGAN, based on the original model [Isola et al., 2017], but implemented in the TensorFlow framework [Hesse, 2017].

Each layer of this stack of multip Pix2Pix GANs is taking care of a particular step of the spatial layout design process for a single family home. The output of one GAN then feeds into another in a sequence: *(I)* generating footprint shape, *(II)* generation room program partition, *(III)* generatiing furnishing details.

The research is one of the first works showing how a complete generative architectural design workflow using GANs could look like. Although the model is trained on a very homogeneous dataset of single family homes found in Boston, USA, the results show the ability of the model to generate diverse solutions for different inputs.

A limitation of the work is that methods for vectorization of the raster outputs generated by the Pix2Pix GAN were not covered by the research, therefore integration with of similar framewroks with common CAD modelling tools used by architects and planners remains a challenge for future development.

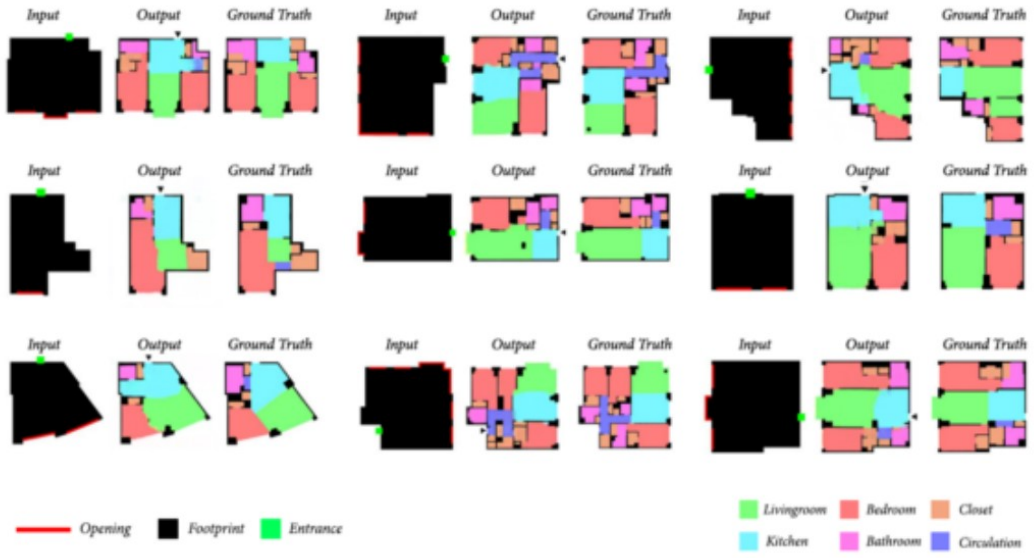


Figure 2.8: Room partition generated by pix2pix based on building's shape and fenestration [Chaillou, 2020]

Generative Adversarial Networks for Pedestrian Wind Flow Approximation

In both Conditional Generative Adversarial Networks for Pedestrian Wind Flow Approximation [Mokhtar et al., 2020] and *InFraRed* [Chronis et al., 2020] a Pix2Pix GAN (in its original implementation by Isola [2017]) was trained to predict urban microclimate metrics around a collection of building massings.

To input the geometry information into the model, the massing is converted into its height-map representation. The GAN model then learns to map the height-map of the geometry to the color-map of wind speed values (in case of both works) or solar radiation values (in case of Chronis et al.) extracted from precomputed simulations using state of the art engines.

Both these works show that the GAN models are able to learn some sort of internal representations of environmental phenomena affecting buildings and vise-versa of the ways the buildings affect their environment. For example, to be able to at least roughly predict the yearly solar radiation value, the model needs to internalize some representation of yearly sun paths and combine them with its internal representation of building shape to output solar radiation values in areas shaded by the building as a product.

It is necessary to state that these models not accurate enough to substitute the standard simulation models in engineering applications, but — comapred to more complex simulation engines — they can serve to indicate the expected performance of the building design much sooner in the design process.

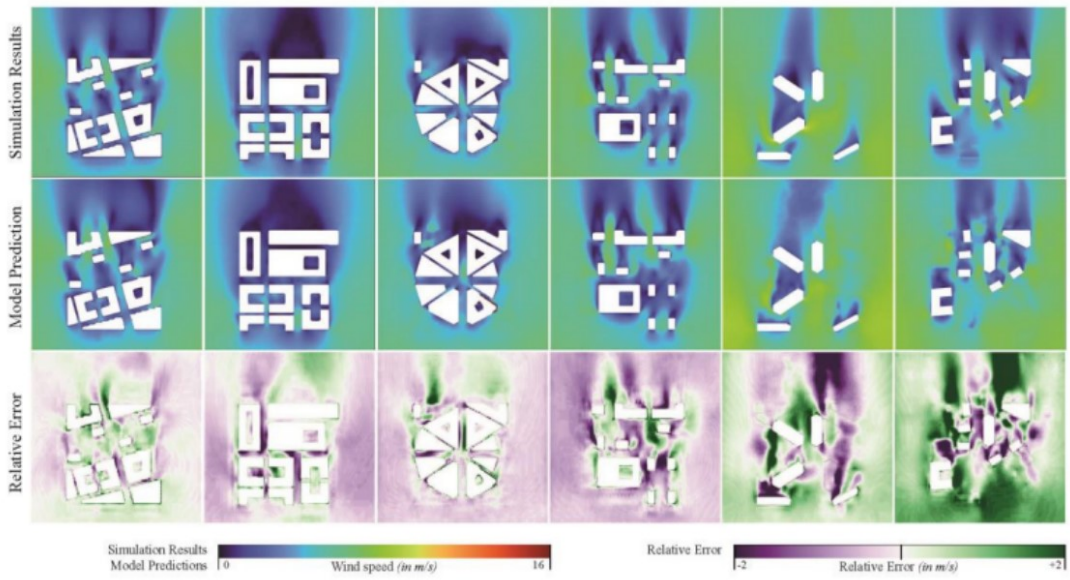


Figure 2.9: Sample of wind speed GAN model predictions together with their relative error compared to the simulation results [Mokhtar et al., 2020]

Generative Adversarial Networks for Urban Block Design

In their master thesis 2021, Fedorova explored the application of pix2pix GAN to the design of urban block configurations based on urban block archetypes collected from multiple cities. In their approach, no parameters of an urban block typical for a certain city are explicitly defined. Instead the algorithm learns them implicitly from the existing urban configurations present in the training data. This approach has then been applied to the multiple cities with different morphologies in order to evaluate how the model performs in different contexts and whether it is possible to transfer learned structures between cities.

This method shows the potential of GANs for learning the properties of urban fabric from the existing context without their explicit definition. However the model model does not seem to be taking building heights into consideration, working only with the 2D building footprints. The clarity of the generated shapes could be further refined with more training.

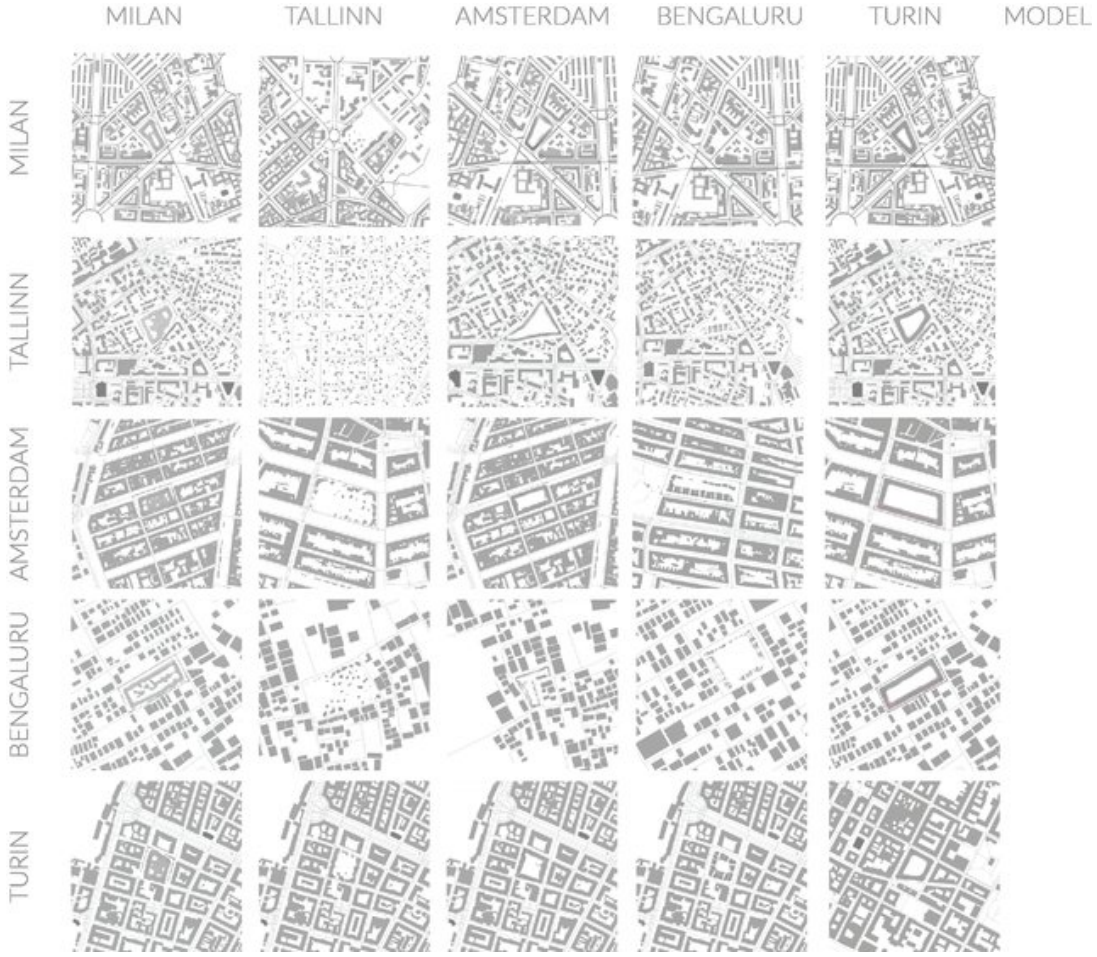


Figure 2.10: Building block designs generated by pix2pix GAN [Fedorova, 2021]

Generative Design Method of Building Group

Generative Design Method of Building Group [Yao et al., 2021] describes a method for the residential building group footprint layout generation. A Pix2Pix GAN was trained on the dataset consisting of footprints of residential building groups collected from Shanghai.

The total collected dataset was split into multiple training datasets based on the building groups density. As a result, networks trained on datasets with higher building density also created building groups with higher building density in their generated output images.

According to the authors, *"the generated results have a certain degree of ambiguity"* and additionally lack the building height information. The authors solved the former problem by manual vectorization of the output and the later problem using *Galapagos* software (evolutionary optimization engine) [Rutten, 2013] to dynamically assign building heights to generated footprints based on optimized solar performance.

This approach of using the output of generative adversarial network as a generator of initial solution that is further optimized using genetic algorithms shows promise especially in terms of increasing the diversity of results evolutionary algorithms would be able to generate if used on their own [Chen et al., 2019].

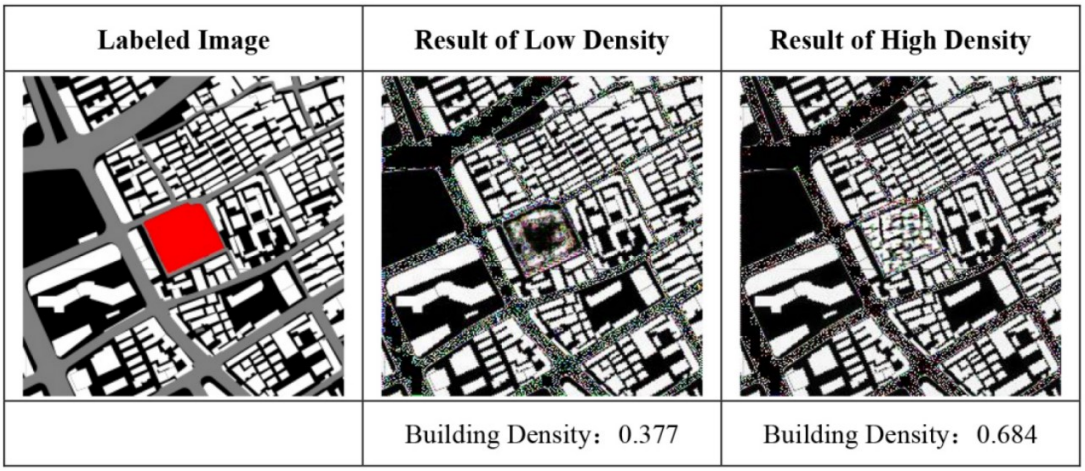


Figure 2.11: The generated results of the trained pix2pix GANs [Yao et al., 2021].

Suggestive Site Planning with Conditional GAN and Urban GIS Data

A Pix2Pix GAN based toolkit has been proposed, as a tool *"could be utilized both in the context of site planning but for morphological analysis for the city's urban fabric."* [Tian, 2020]. A model is trained on a dataset of building group footprints from the city of Boston.

Interestingly, the use category of each building is encoded as color information. The model learns to generate the building footprint shapes and assign building use to them based on the shape of the plot input into the model. The generated raster images are vectorized into geometrical features representing the building footprints.

From my review it seems that no other information, beyond the plot shape and the buildings placed on it, is provided in the training data. Therefore it is given that the model is not capable of reacting to the site context except conforming the generated building footprints to the boundary of the plot. Building height information is also omitted from the model, with the building features being extruded to uniform height, controlled manually by the user.

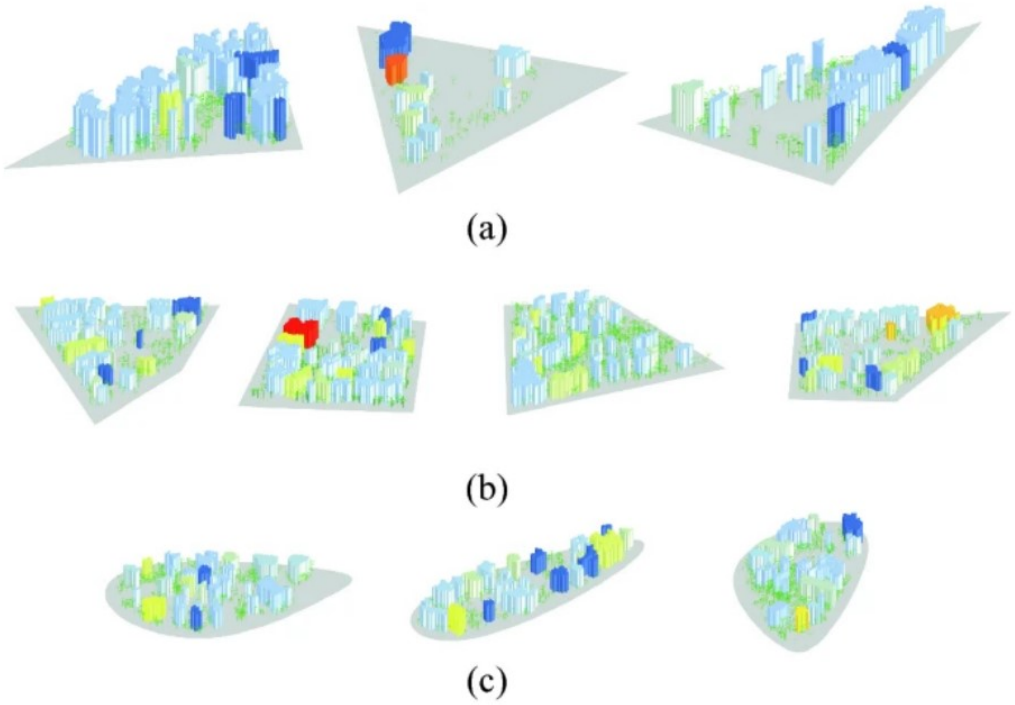


Figure 2.12: Site planning suggestion for (a) Triangle shape site boundary (b) Quadrangle shape site boundary (c) Spline shape site boundary [Tian, 2020].

2.4 Conclusions

Increased popularity and relative ease of use of open source image-based GAN frameworks in recent years has motivated architects and urban designers interested in computational design methods in finding plausible ways to apply them into the design process.

Limitations of existing work Existing reviewed methods for generating layouts of building groups are applied only to the problem of two-dimensional footprint arrangement and learn to generate their outputs from input data covering only limited number of site context aspects (e.g. only the plot shape and surrounding building footprints).

From the literature study, it became evident that existing work done on generating building or building group massings are applied only to the problem of two-dimensional footprint arrangement and learn to generate their outputs from input data covering only limited number of site context aspects (e.g. only the plot shape and surrounding building footprints). Most of the research on using GANs for this purpose was largely done either at the LoD 0.1 (effectively using the GAN to generate only the building footprints, not massings per se.) or at LoD 1.1, with the building models sourced form OSM dataset.

I recognized the availability of highly detailed building models as one of the most limiting factors to the maximum possible fidelity of generated results with similar methods. It is my assumption this is caused by difficulties in finding high quality data sources that describe the three-dimensional geometry of buildings together with the urban context in a dataset sufficiently large to be practical for training GANs.

Potential improvements By using a LoD 1.3 and higher source of building geometry to train the model, it should be possible to generate not only general footprints and maximum height of buildings, but also include some details of the roofscape for multi-level roofs. From LoD 2 and higher, the model can also observe the structure of the roof from buildings, e.g. with pitched roofs. One of such sources of building geometry is the 3D BAG dataset, which I will be exploring and using further in this research.

Additionally, my preliminary findings lead to the conclusion that existing models often lack the framework to accurately compare the similarity between the training and generated data using metrics more typically used in urban and architectural design. Without this, it is hard to evaluate the quality of their outputs and their usability in design practice. Therefore I aim to introduce more rigorous evaluation framework for evaluating the accuracy of the produced models.

3 Methodology

In the following chapter I will describe the general methodology, describe the datasets and methods used to build the training data, together with some information on how the datasets used are constructed and collected, to clarify any limitations that the choice of these data sources might have on the results, and explain my motivations behind my methods for manipulating these datasets.

Finally I will provide an overview of the methods used for building the training datasets, training the GAN model, pre-processing the output, and reconstructing its 3D representation, together with metrics and methods for evaluating the generated results.

3.1 3D building models

3D BAG is a fully automatically generated open dataset containing detailed 3D building models for 10 million buildings in the Netherlands. The dataset can be considered state-of-the-art, thanks to the building geometry being modeled with high LoD (up to LoD 2.2) compared to other large building geometry datasets of similar scope. To understand some of the decisions taken within the methodology of this thesis, some background and details of the methods and datasets the 3D BAG is based on will be presented first in the next paragraphs.

The automatic creation of the dataset is possible mostly due to the open availability of highly accurate elevation data in the Netherlands. The Actueel Hoogtebestand Nederland 3 [1] (AHN3) is a high-resolution point-cloud dataset acquired via airborne light detection and ranging (LiDAR). The latest surveying campaign was carried out during the years 2014 – 2019 and the resulting datasets made publicly available shortly afterwards. The resulting LiDAR point cloud has an average point density of 8 points per square meter across the whole country.

3D BAG combines the AHN3 data with building footprints acquired from Basisregistratie Adressen en Gebouwen [2] (BAG) and Basisregistratie Grootschalige Topografie [3] (BGT). For each building a subset of AHN3 points located within its footprint is isolated and a collection of best fitting planes is matched to this partial point-cloud using a RANSAC algorithm. Intersection lines of these planes are used to create a planar partition of the original footprint. Each vertex of the planar partition is then elevated back to corresponding height to form the final building roof geometry. Vertical walls and ground plane are then simply added based on the building footprint to form a watertight solid (see Figure 3.1 for graphical explanation of 3D BAG methodology by one the authors of the algorithm - Peters [2021]).

[1] Current Elevation Data of Netherlands
[2] Register of Buildings and Addresses of the Netherlands
[3] Large Scale Topographic Map of the Netherlands

This methodology has some implications which have to be considered. Since the geometry is based on elevation data collected in years 2014 – 2019, any buildings or changes to existing buildings more recent than 2014 can be potentially missing. The authors of the dataset additionally try remove potentially problematic building types, such as any buildings that show overlap with other buildings or major infrastructure, or greenhouses, which often include erroneous LiDAR scan points in their footprints due to issues the LiDAR scanning technology has with capturing reflective and transparent glass surfaces.

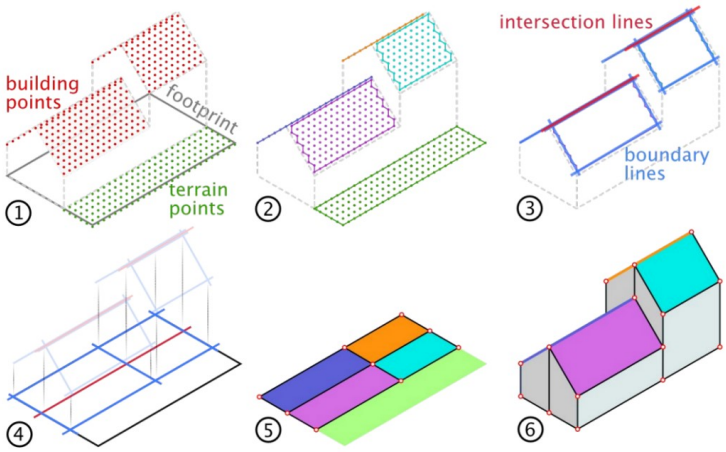


Figure 3.1: Process of building geometry reconstruction used by 3D BAG [Peters, 2021]

Due to the method used to generate the building geometry by raising vertices of planar partition, the algorithm is unable to generate any building geometry with overhanging geometry. A planar partition by definition cannot contain overlaps, therefore it is impossible for any roof parts to overlap in their ground projection. While this is certainly a limitation to the geometries 3D BAG algorithm can generate, it is favorable in the context of this research, since the intent is to translate the building geometry to a raster height-map before it can be consumed by the Pix2Pix GAN.

In a height-map, only a single height (Z value) is assigned to each pixel and by extension, a discrete XY coordinate pair. This means that no X,Y coordinate pair can map to more than 1 Z-value. This is usually a limitation, but in this case for the reasons above, it can represent the 3D BAG geometry fully, without any loss of information, except the loss of information fidelity caused by discretization of the geometry from vector to raster representation.

The 3D BAG dataset is provided for use in three LoDs - LoD 1.2, LoD 1.3 and LoD2.2 (3D geoinformation research group, 2021). The building features in the first two LoDs are available through a typical and widely adopted geospatial data transfer API - Web Feature Service (WFS). This is the case since in LoDs 1.0-1.3, any building geometry can still be described as a collection of planar shapes in the XY plane with an extrusion (Z-)height attribute associated with each of them, and therefore is fully transferable through protocol based around 2D shape data.

Since LoD 2.0 and higher allows for use of non-extrusion-based geometries for building features, the most detailed LoD available, LoD 2.2 is provided only through fully 3D enabled

data formats. Namely Wavefront OBJ, OGC Geopackage and TU Delft's GeoJSON, with only the later two preserving most of the semantic information associated with the building geometries. Additionally, the LoD 2.2 data is only available to be requested in larger chunks of pregenerated tiles (each tile consisting of max. 3500 buildings [Peters, 2021]).

For LoD 1.2 and LoD 1.3 building models, 3D BAG provides multiple height values to be used for each building or building part. Each provided height corresponds to a certain percentile of heights of all AHN3 points detected within the footprint. Since the AHN3 points tend to be evenly distributed and located mostly on the roof, one can think of the range of percentiles as an interval between the lowest and highest heights of the roof. From experimental research, it seems that LoD 1 models using the 50th percentile of roof heights as extrusion height have the smallest root mean square error (RMSE) of building volume compared to the LoD 3 (ground truth) models [Biljecki et al., 2014] (see Figure 3.2). Therefore the 50th percentile (median) height value is used for generating all LoD 1 models in this work.

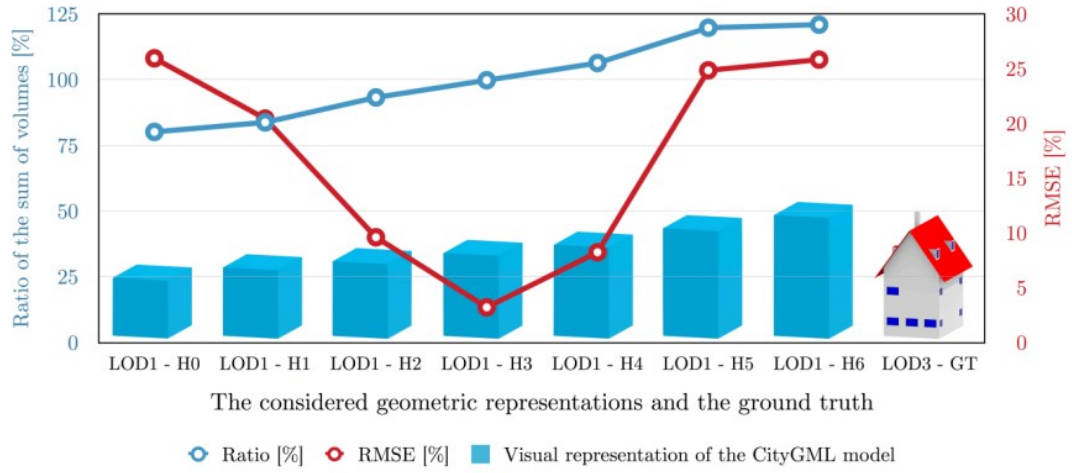


Figure 3.2: RMSE of volume of LoD1 geometry at different roof heights compared to ground truth [Biljecki et al., 2014]

3.2 Other input datasets

While the 3D BAG dataset provides me with high LoD building massing models, I further enrich my representation of urban environments the GAN learns from. To be able to fill in the building massing that appropriately reacts to its site context, the model benefits from additional information. Such information can be the shape of the plot the building is (or will be) located on, the main axes of both pedestrian and vehicular circulation in the area, the types of surfaces on the site, or information on nearby landscape features and greenery.

To acquire these data, I request and merge together data from several additional datasets. All of them are openly available data provided by the Dutch government via the PDOK platform [PDOK, 2022].

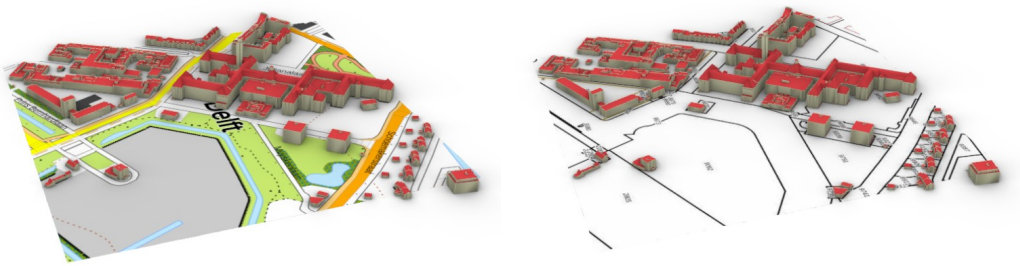


Figure 3.3: 3D BAG geometry overlapped over different datasets *(left: TOP10NL, right: BRK)*

Basisregistratie Adressen en Gebouwen [1] (BAG) The BAG is a centralized, openly available dataset of all buildings and addresses in the Netherlands. Each municipality is responsible for regularly updating the data as new buildings are registered, built or demolished.To include additional semantic data about each building in the area of interest, which were omitted from the 3D BAG dataset, additional data are requested from the BAG dataset via Web Feature Service (WFS) queries. In particular I'm interested in information about building construction age (*bouwjaar*) and its use (*gebruiksdoel*). BAG also stores information of the building area (*oppervlakte*), which can be used together with the construction year and use information to create database queries and selectively focus the data collection effort only on areas where buildings with specific properties are found.

TOP10NL The TOP10NL dataset is part of the TOPNL series of topographic maps, provided by the Topographic Register of the Netherlands. The TOPNL maps are provided at different scales, with TOP10NL being the most detailed one (1:5000 – 25000). The TOP10NL can be used in its default form as a simple basemap. Its constituent layers can be requested by modifying the parameters in the Web Map Service (WMS) requests. I use the TOP10NL dataset to gather information on land cover *(LC LandCoverUnit)*, natural landscape features *(Terrein vlak)* and extent of road *(TN RoadTransportNetwork)* and highway *(Wegdeel vlak)* surfaces in the area (see Figure 3.4).

Basisregistratie Kadaster [2] (BRK) The BRK dataset is the cadastral map of the Netherlands. It contains cadastral parcel boundaries, parcel number and the footprint of the buildings within the parcel. In this work, I use the cadastral parcels to define what is the plot belonging to the building, by creating a join of all cadastral parcels the building overlaps. Even though the existing cadastral parcellation is often changed to accommodate large new construction developments, historical parcellation found under current buildings gives a good indication of the project site boundary available for the massing design at the time the building was created.

[1] Register of Buildings and Addresses of the Netherlands
[2] Cadastral Register of the Netherlands

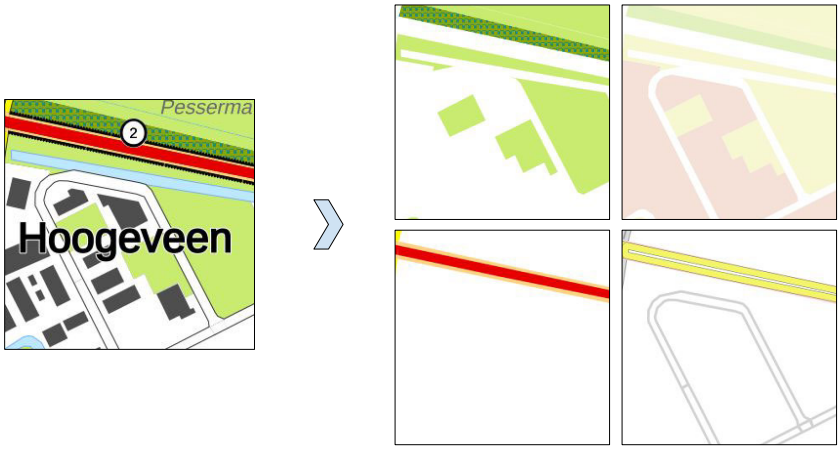


Figure 3.4: TOP10NL basemap and the layers extracted

Nationaal Wegen Bestand [1] **(NWB)** The NWB is a database of all public roads in the Netherlands that have a street name or road number and is regularly updated by national government, provinces, municipalities and water boards. In this work I use the NWB to extract the main axes of pedestrian and vehicular circulation around the site and extract the borders of separate city blocks out of the urban fabric. While the road surfaces are already present in the TOP10NL dataset, NWB can be used to extract the corresponding road centerlines as geometric features using WFS (compared to rasterized representation of roads available through the TOP10NL WMS).

3.3 Data collection

Resolution

To process the geometrical data and the site information into an image input consumable by the Pix2Pix GAN model, the features have to converted into a raster representation of constant resolution. The target resolution for the raster images used in training is decided on one hand by the architecture of the Generator network used by the GAN, since different architectures have different limitations on the resolutions they are capable of processing. On the other hand it is limited by the amount of memory available on the GPU.

For the Generator network two CNN architectures were considered during the course of my research. U-Net 256 [Ronneberger et al., 2015] requires the images to be at a fixed resolution of 256 by 256 pixels (or its multiples). Other architecture is ResNet9 [He et al., 2016], which requires the resolution of the images to be a multiple of 4. Therefore 256 by 256 was settled upon as a target resolution usable with both Generator models. Using images of higher resolution is possible, but with increased resolution the number of the model parameters increases exponentially both for Generator and Discriminator. This would make it not be possible to load the full model into memory of the consumer-grade GPU.

[1] National Roads Database of the Netherlands

Scale

The resolution of 256 by 256 pixels was settled upon as the ideal target for training images considering both its usability with the chosen GAN architecture and the limitations of the used hardware. While the resolution has to remain constant, the scale of the images, meaning the size of the geographic area that fits into this frame, remains variable. While a typical urban site plan will use scale ratio, for example 1:5000, in this work I will use scale measured in meters per pixel (m/px), as I am dealing with digital data not intended for print. It is however possible to convert per-pixel scale to scale ratio, assuming a/the canonical pixel density of a screen of 96PPI. In that case, a pixel resolution of 1px/m would convert to a scale ratio of 1:3820.

To decide on set of scales to be used for the data collection, I experimented with number of scales ranging from 0.5m/px to 4m/px, evaluating the amount of site context captured in a 256 by 256 pixel frame and considering the amount of detail that would be eventually recoverable from the output of the GAN model (ie. 4m/px model is only able to capture features larger than 4m in their footprint). Eventually two scales - 1m/px and 2m/px - were used for most of the training carried out during this research.

Figure 3.5: Scale test samples collected from area of Amsterdam Zuidas

Massing

In each frame, independent of its scale or content, I am interested in collecting several kinds of information. My intent is to collect as many data points as possible about each frame and store them in a local database. Later these data can be selectively retrieved based on the intended application or setup for a concrete test-case experiment.

The first and probably most important data type is the geometry of all buildings, which I obtain from the 3D BAG dataset. For all buildings in the frame of interest, their LoD 1.1 and LoD 1.3 geometry is requested using a WFS query. The geometry is then converted into its height-map representation.

The height information for each polygon representing a separate building part is encoded into a 8-bit pixel color value, remapping height values between 0 and 100 meters to 55-255 values (resulting in 0.5 height difference per pixel value step). The features are then rendered into a raster using a closest-neighbor interpolation (to avoid interpolation of height values across building footprint edges). This is effectively equivalent to voxelization of the geometry, with the voxel dimensions dependent on the image scale (see Figure 3.6).

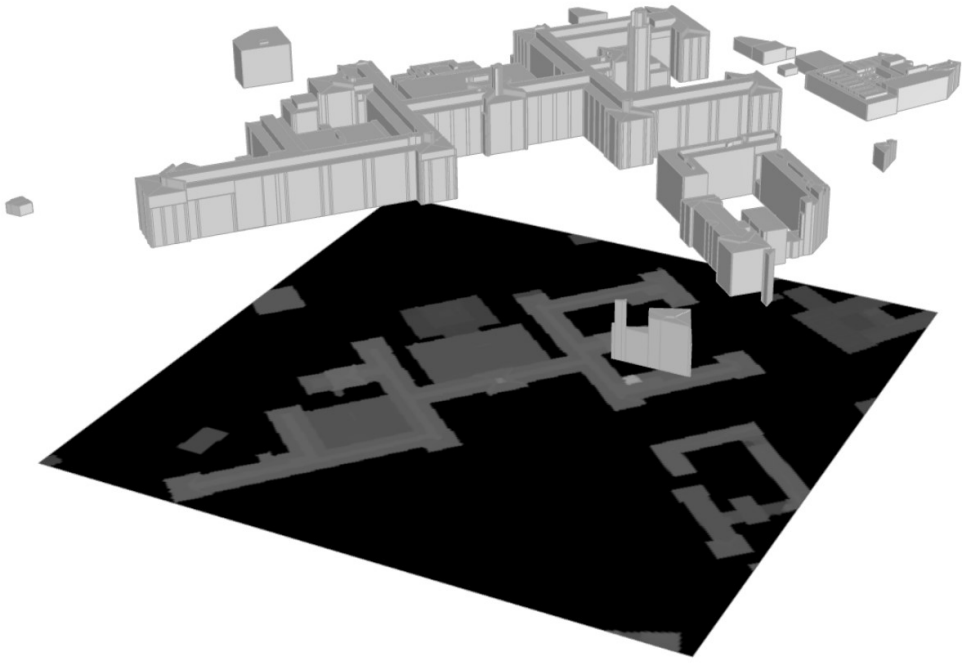


Figure 3.6: LoD 2.2 prismatic massing of TU Delft Faculty of Architecture with its height-map representation below it

Target building and plot

The general strategy for teaching the model to generate appropriate building massings is to remove a single or multiple buildings from the building geometries present within the frame and letting the model predict the features of the form of the features removed. A mechanism for selecting a set of buildings within a frame therefore is necessary. Additionally, if one wishes to limit the area where the model attempts to insert the missing massing, they should include an intended boundary for the model to fill in.

Parcel definition One of the strategies I propose to create this boundary, is to use the cadastral parcel the target building is located on. However, what parcel is the building located on is not always a clear cut question, since the parcelation rules are highly irregular (see Figure 3.7). Therefore, I consider the join of any parcels that is overlapping the building footprint as a the total building associated parcel, which is collected into the dataset.

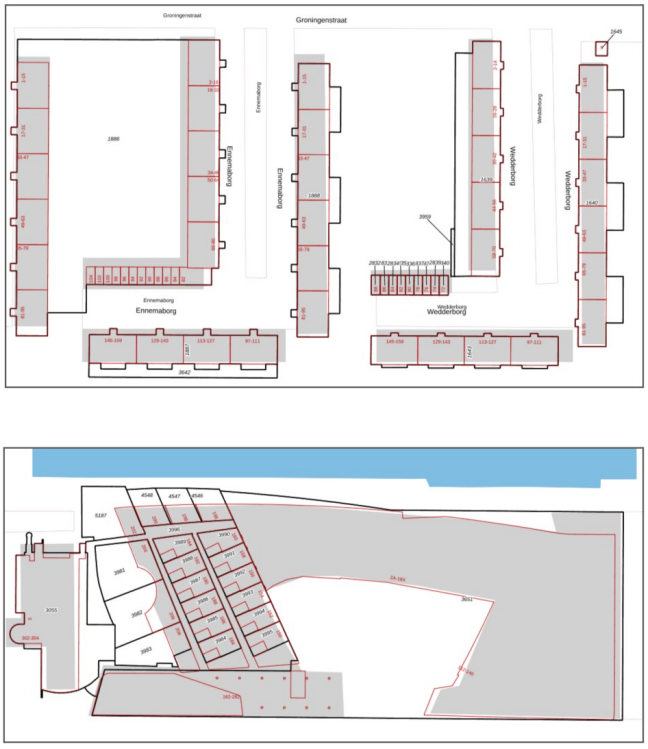


Figure 3.7: Examples of different parcelation types, top-left: one parcel to many buildings *(top-right: one parcel to one continuous building, bottom: many parcels)*

Block definition To test the ability of the GAN to generate massings for larger cohesive groups of buildings, I decided to include a boundary of a larger urban block, if present within the frame, as one of the data layers stored for each data sample. To define what constitutes a block, I make use of the street network features. An urban block in this case is defined as a continuous area completely encircled by streets.

I extract the polygons defined by the NWB road network curves (center lines of any named street or road in the Netherlands) and choose the one closest to the center of the frame as the "block" of a specific data sample. If no such polygon is found, no block is associated with the data point.

Properties of each block are stored, to enable further filtering of the blocks with specific traits to be used for training. Some of the stored properties are the number of buildings contained within the block, their volume and footprint areas, their functional uses and construction year(s) and whether the block is located within the central part of the frame (to avoid training on blocks with site context obscured on one or multiple sides). Finally, the block geometry is rasterized as a bitmask and stored to a local collection.

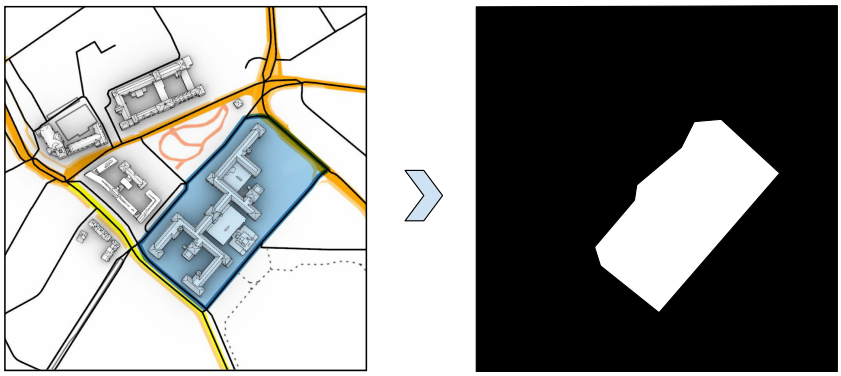


Figure 3.8: Example of a single *"street block"* defined as a continuous area separated from the surrounding by roads on all sides

Density reduction masks

To test the ability of the GAN model to recognize deficiencies within existing urban fabric which could be used as potential targets for densification, I decided to generate a series of density reduction maps. These are boolean masks masking out a progressively increasing amount of buildings randomly selected from all buildings present within the frame.

The goal is to use these to mask out a varying amount of existing buildings and let the model learn to identify the areas with gaps in the fabric of the existing build-up area. Based on the amount of removed building in the datasets the model is learning from, it is expected to have different thresholds for to which extent it learns to fill the gaps in the fabric. If a real site without any existing buildings removed is input into a GAN model previously trained to find the underdeveloped locations in sites with synthetically decreased density, I expect the model to find new unique locations where the site could be filled in with new development.

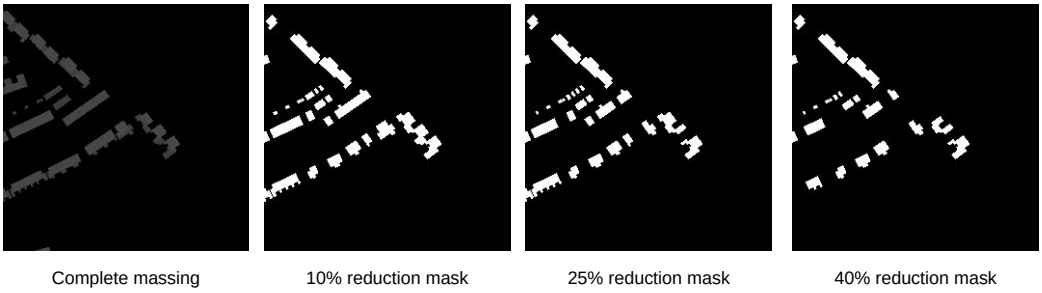


Figure 3.9: Series of density reduction maps

Site context

To provide additional information to the model to infer the building massings and placings from, I intend to include additional information to the model. I decided to focus on information on the land cover, the natural landscape features, the road surfaces and any major road links in the area. This information can be sourced from the TOP10NL topographic map model of the Netherlands. The data are requested via WMS as individual layers of the full map. To store the simplest possible representation of the information, without relying on the color or texture features of the images, a series of filters is applied to the requested data before converting them to a black and white representation (see Figure 3.10).

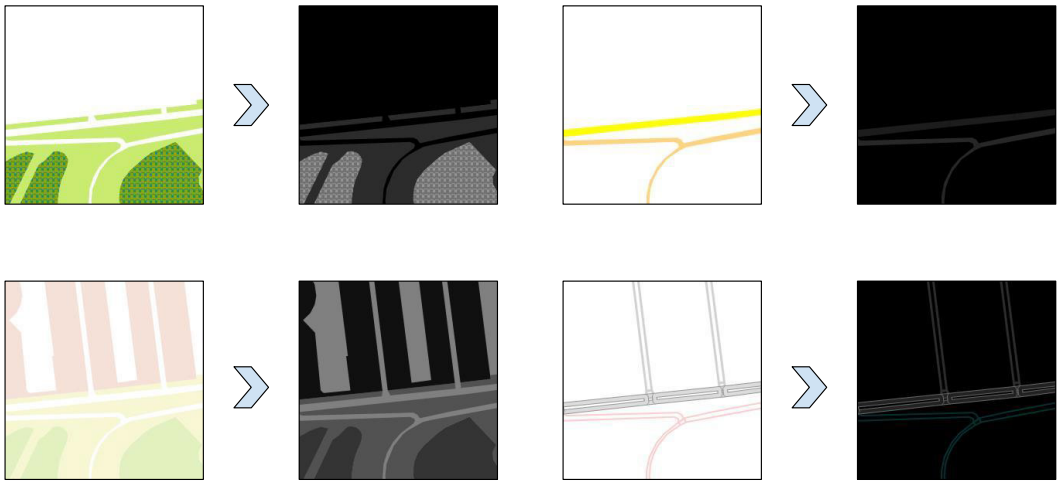


Figure 3.10: Site context map layers sourced from TOP10NL, converted to greyscale representation.

Data scrapping method

After deciding on the scale, the next challenge is how to iterate through the available datasets step by step to export the data points for the GAN model to eventually learn from. The first proposed approach is similar to the XYZ map tiling method, the same approach that web map providers such as OSM or Microsoft have adopted for breaking large map datasets into smaller more manageable map tiles [García et al., 2012]. For a chosen area of interest, the algorithm divides the location into a set of tiles of set size (ie. 256 by 256 meter) and then iterates through the tile grid, requesting and collecting data for each tile. In each tile, the building closest to the tile center is then considered the target building (see Figure 3.11).

However, after several iterations of building training datasets using the above-mentioned XYZ map tile methods, some disadvantages were identified. First of all, I observed that if I use an XYZ tiling of a bounding rectangle of larger urbanized area, not all parts are equally densely built up, with many tiles containing very few buildings if any at all - essentially creating a wasted data point not usable for training. Second, the method is fairly inefficient, since in one tile multiple buildings of interest may lie, but only one can be captured as the target building per tile and, by definition of the XYZ tiling, the same area cannot be contained within any other tile. This means that plenty of usable building data will be skipped. Finally, this method does not allow for specifically targeting certain types of buildings, which is useful for quickly building up a specific dataset. Instead, it gathers a random collection of building types roughly corresponding to the building type distribution in the tiled area area.

Figure 3.11: Algorithm to find a target building and a parcel within the tile.

To solve these issues, I eventually decided to use a different data collection strategy, utilizing the information on individual buildings available in the BAG. By means of XML-encoded filters included in the database request, one can search the BAG database for specific kinds of buildings, either in the area of interest defined by a bounding box, or in the whole dataset (see Figure 3.12). The maximum number of features returned via WFS request to BAG database is 1000, but by using pagination parameters it is possible to quickly scan the database for larger amounts of buildings.

Once the properties of buildings to be added to the local database are defined, the data collection pipeline iterates through the features returned by the BAG WFS service. The frame of interest, sized depending on the scale, is centered on the mid-point of the building

```
<Filter>
  <And>
    <PropertyIsEqualTo>
        <PropertyName>gebruiksdoel</PropertyName>
        <Literal>woonfunctie</Literal>
    </PropertyIsEqualTo>
    <PropertyIsGreaterThan>
        <PropertyName>oppervlakte_max</PropertyName>
        <Literal>200</Literal>
    </PropertyIsGreaterThan>
  </And>
</Filter>
```

Figure 3.12: Example of a simple XML Filter used return all residential buildings with footprint larger than 200m^2

of interest, defined as a center of each building's bounding rectangle. Therefore, all the above-mentioned image-based data layers are collected by requesting the geometry features and raster map layers with this generated frame of interest used as a boundary. The images are then converted to a single-channel (black and white) representation before being stored (see Figure 3.10). This is done to make eventual image manipulation and data merging easier when manipulating the raster layers into the representations used for the GAN training.

Additionally, I compute and collect a number of additional statistics that can be used later to filter the collected data when creating tailored training datasets. See Table 3.1 for an overview of the collected variables. These variables represent an elementary but complete enough set for us to derive a number of spatial statistics per target parcel, block or the complete area of interest (I will expand on these in Section 3.6 Evaluation on page 41). This is then particularly useful when building training datasets targeting specific types of building configurations.

Variable	Collected for
BAG Pand ID	main building
Coordinates (EPSG:28992)	main building
Area	main building, parcel, block
Area of buildings contained within	parcel, block, frame
Footprint perimeter length	main building
Volume	main building
Count of buildings within	parcel, block, frame
Volume of buildings contained within	parcel, block, frame
Building age statistics	main building, parcel, block, frame
Building use statistics	main building, parcel, block, frame
Proportion contained within frame	main building, parcel, block
Proportion contained within subframe	main building, parcel, block

Table 3.1: Overview of main variables collected per collected data point

3.4 Training data

With the data collection pipeline established , one is able to generate a large number of data points for which multiple layers of rasterized spatial data can be retrieved. These include; information on building forms in the area captured, stored in its height-map representation at multiple LoDs, information on the axis of pedestrian and vehicular network in the area, and information on the natural features land cover and the extents of street surfaces (see Figure 3.13).

I also collect a number of rasterized binary masks, encoding different areas of interest in the frame, which can serve as targets for the GAN model to fill in; collections of buildings in the frame, single parcel extents or single city block as defined by street network (see Figure 3.8).

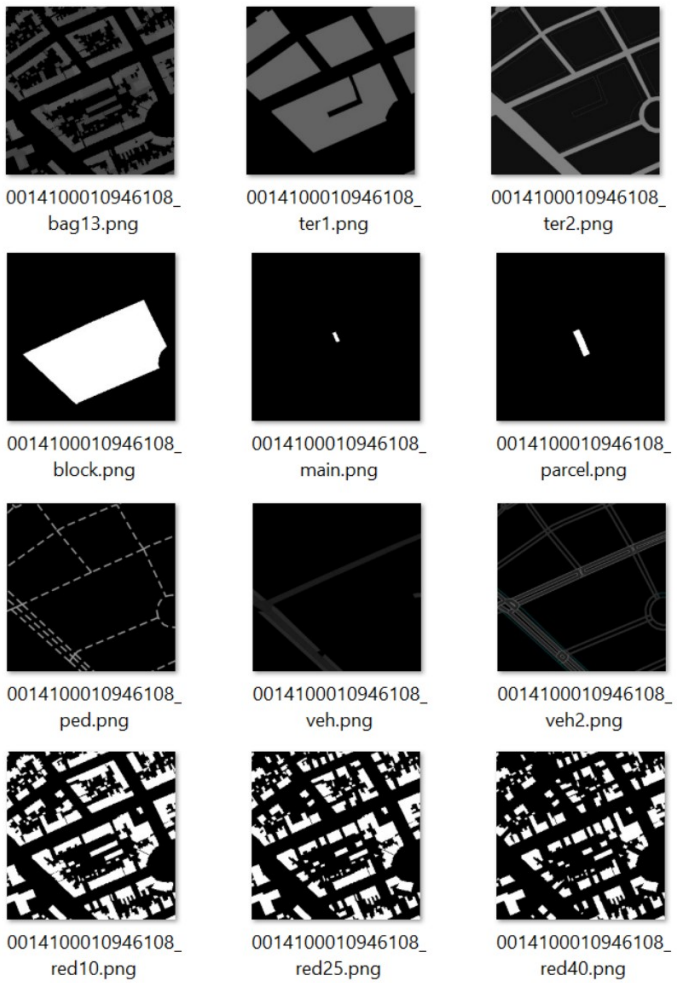


Figure 3.13: All raster data layers extracted per data point; *starting from top row - 3D BAG height-map, natural landscape features, land cover, block footprint, target building footprint, target parcel footprint, street axes, major road links, all road surfaces, set of 3 density reductions (10-40%)*

Additionally, a set of both numerical and semantic properties associated with each collection of rasterized spatial data are stored. This data are derived before the rasterization from its original feature representations captured within the frame of interest. This includes information like built-up area, gross floor area (GFA), volume statistics, building age statistics or building use information. Most of these statistics are stored multiple times, per each target feature mask available as a raster.

Collected statistics are used when building training datasets are targeted at specific type configurations or buildings with specific properties. For each data point a number of spatial indicators [Berghauser Pont and Haupt, 2009] can be derived, defining the characteristics of each area of interest (AoI) - e.g. a cadastral plot, a street block, or their surroundings.

These include the floor space index (FSI) (see Equation 3.1) and describing the general intensity of use in the AoI;

$$\text{FSI} = \frac{\text{GFA of all building in AoI}}{\text{Total area of AoI}} \tag{3.1}$$

ground space index (GSI) (see Equation 3.2) describing the density of built up areas in the AoI;

$$\text{GSI} = \frac{\text{Total footprint area in AoI}}{\text{Total area of AoI}} \tag{3.2}$$

or open space ratio (OSR) (see Equation 3.3) describing the intensity of use of open space in the AoI;

$$\text{OSR} = \frac{\text{Total non-built-up area in AoI}}{\text{GFA of all building in AoI}} \tag{3.3}$$

For example, if one wishes to build a GAN model trained on specific types of densely built residential blocks built between 1970 and today, one can introduce a series of query filters when retrieving data points from my local database to retrieve all matching samples.

1. FSI in image frame ≥ 0.3
2. FSI inside street block ≥ 0.9
3. mean construction year inside street block ≥ 1970
4. use inside street block contains '**woonfunctie**' ?

Using such queries I can swiftly design and implement experiments to train the GAN model on various datasets, controlling the type of site contexts and building typologies present within them, and observe how the generated solutions output by the model differ depending on the kinds of samples the model was trained on.

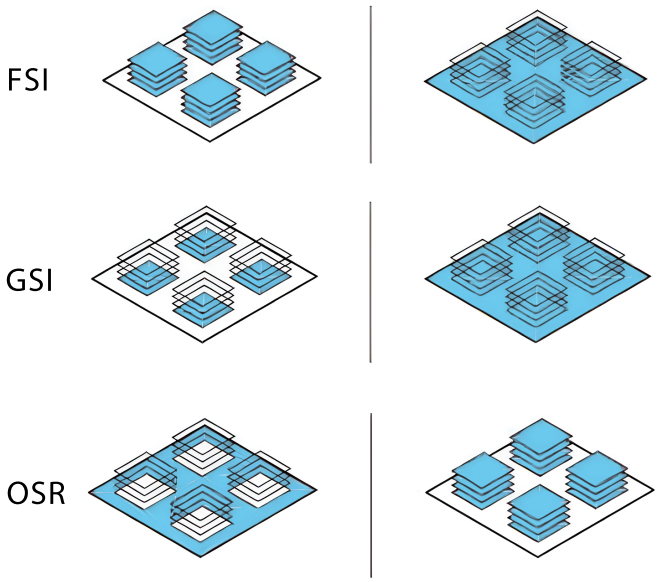


Figure 3.14: FSI, GSI, OSR (redrawn from [Berghauser Pont and Haupt, 2009])

With the data of interest selected, the next step is composing the structure of the image pairs the model will train on. One image pair represents an example of the desired mapping from image A to image B that one intends for the model to learn. The Generator tries to find a general function that maps the contents of each image A in the training dataset to an image as similar as possible to its corresponding image B. In parallel, the discriminator learns to classify whether an observed image pair contains a *real* image B from the training dataset, or a *fake* image B generated by the Generator.

Thereafter, I will summarize my method for composing these image pairs for the purpose of training GAN models capable of generating new building massing designs within existing site context as follows; Image A contains the geometry of the buildings in the frame, with a subset of buildings within the area of interest removed. Instead, this removed subset is contained in the image B, since it is the goal of the model to find a function that can generate this target subset of building geometries. Additionally, I include additional information in image A, that I expect the model can benefit from when inferring the form of the missing massings.

The standard implementation of Pix2Pix GAN architecture uses 3-channel *RGB* images with 8-bit color depth to train the model. I make use of these channels to separate various primary data layers - geometry, site content, and area of interest (see Figure 3.16 *RGB* encoding). This makes later data retrieval easier and reduces ambiguity in areas where the data layers overlap, potentially improving the learned mapping of the GAN. Using this framework I can then experiment with which data layers I use to compose each channel (see Figure 3.15).

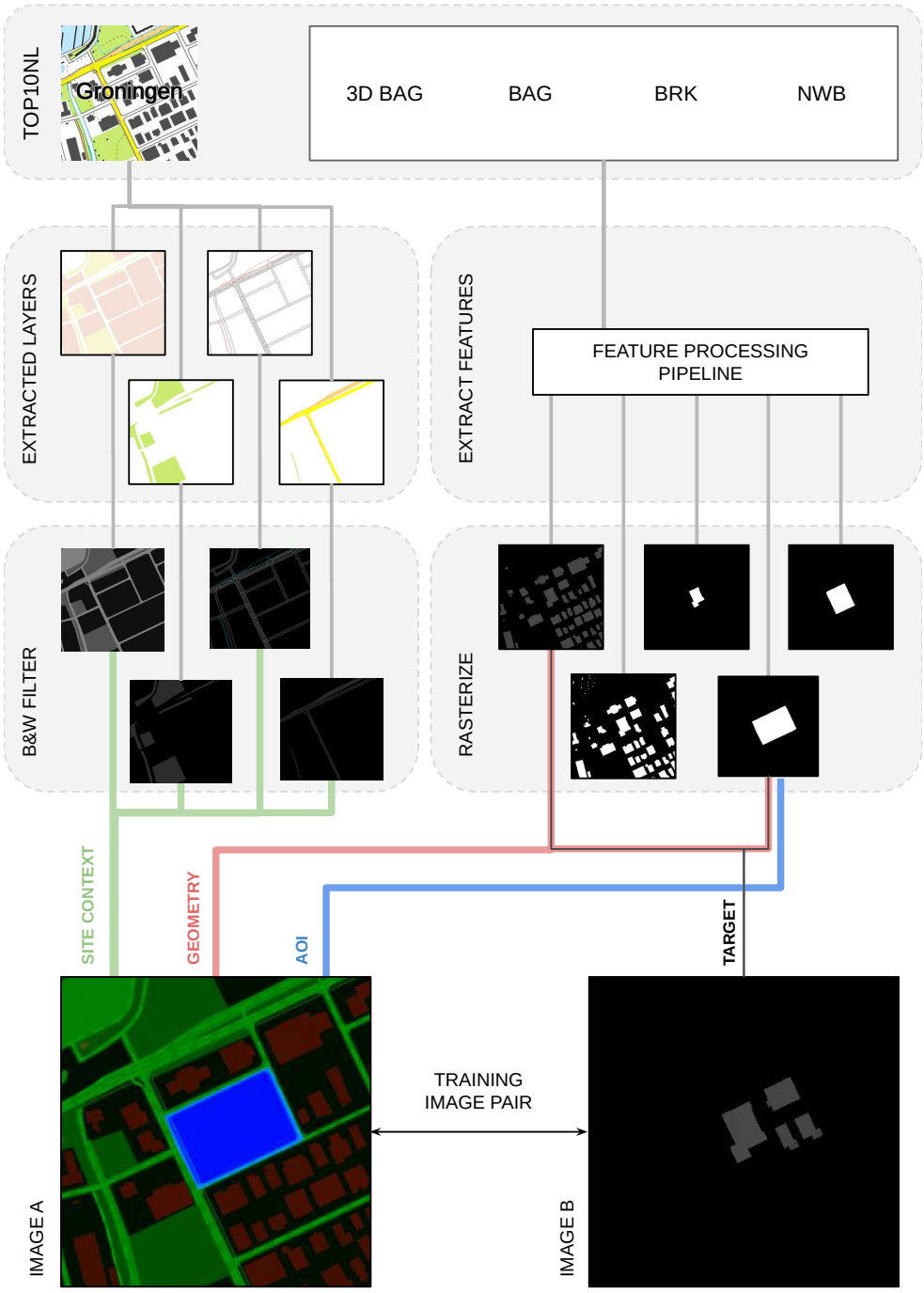


Figure 3.15: Overview of a pipeline for composing spatial data into specific training image pairs

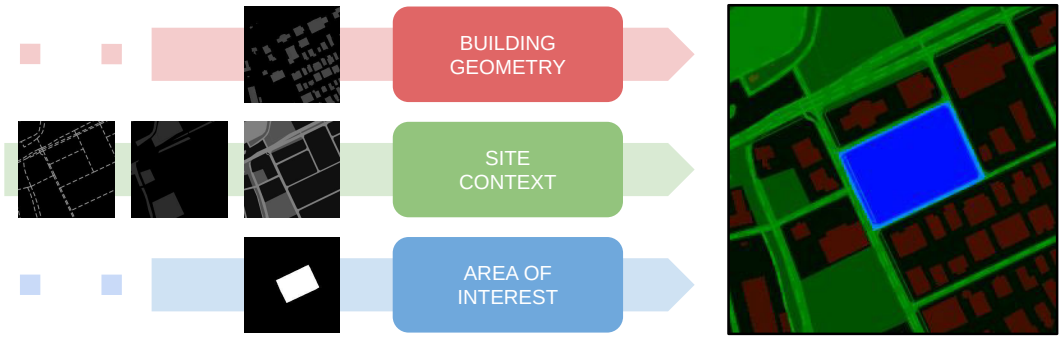


Figure 3.16: Proposed encoding of different data layers into separate *RGB* channels

3.5 Geometry extraction

The last step of the massing generation pipeline deals with the extraction of the massing geometry suggested by the GAN network from the output images. The building geometries are converted into a height-map before being used as training data for the network. Therefore, one needs a method to retrieve the resulting 3D geometry from the height-maps the model learned to generate.

It is good to note that due to the limitations of the methods used to create the 3D BAG dataset, which acts as the primary source of building geometry data for this work, the original building geometries contain no overhangs. This means the amount of information lost during the translation between 3D geometry and its height-map representation is limited only by the resolution of the height-map.

Considered approaches Multiple methods can be considered to reconstruct the geometries. First is voxelization, a method where a number of voxels, based on the pixel's value, is constructed above each pixel of the height-map. Voxels, equivalent to pixels in 2D space, represent values in discrete 3D space. The output layer of the GAN framework is a two-dimensional array with the third dimension encoded in each field. Therefore using voxelization effectively reconstructs the representation that the GAN model is outputting without any additional bias.

The second method to be considered is vectorization based on the *3Dfier* algorithm [Ledoux et al., 2021]. This is the same method used to generate the 3D BAG's LoD 1.1 geometry. Footprint of each building is extruded to a single height, defined as a percentile of all heights found within the footprint. Using the *3Dfier* method has the advantage of generating simple geometries with identical data-model to the original geometrical data used.

On the other hand, the *3Dfier* method requires a vectorization of the building's footprint as a feature to be generated. Reliable vectorization of raster images and in particular outputs of GAN networks is still an open problem [Dong et al., 2021], as it generates features that are often too complex to be practical.

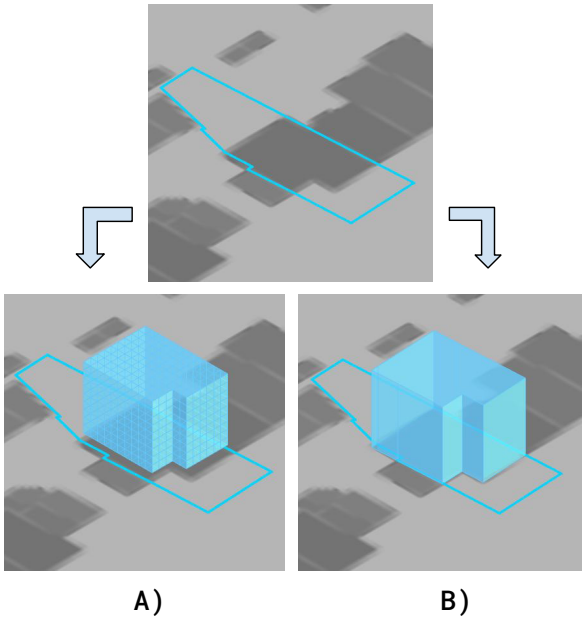


Figure 3.17: Methods of extracting 3D geometry from 2.5D height-maps; *A: voxelization; B: 3Dfier method*

Simplification of the vectorized features is possible, but inherently introduces some bias to the output generated by the GAN model. Additionally, using the *3Dfier* bases approach requires using a single height value for the whole building footprint, effectively omitting any height differences the GAN model generated. Essentially this corresponds to reducing the LoD of the output to LoD 1.1.

Voxelization For the above-mentioned reasons, the voxelization of the output height-maps as the primary method to evaluate the 3D geometry generated by the model. The vectorization of the geometry based on the *3Dfier* method is used only for visualization purposes.

The voxelization of the height-maps begins by converting the GAN output to its true gray-scale representation by averaging the *RGB* channel values to avoid influence of any color noise introduced by the GAN. Optionally, the image is downscaled from 256x256 pixels to 128x128 pixels as part of preprocessing pipeline to lower the amount of voxels generated and improve computation speed on large datasets. Finally a raster smoothing filter is applied if noise is still present within the height-map generated.

The image is deconstructed into an array of height values. Since I originally mapped a range of building heights from 0–100 meters to *RGB* values (55,55,55) to (255,255,255), a single integer step in gray scale value corresponds to a height difference of 0.5 m. The dimension of voxels in the X and Y axis depends on the scale of the original image.

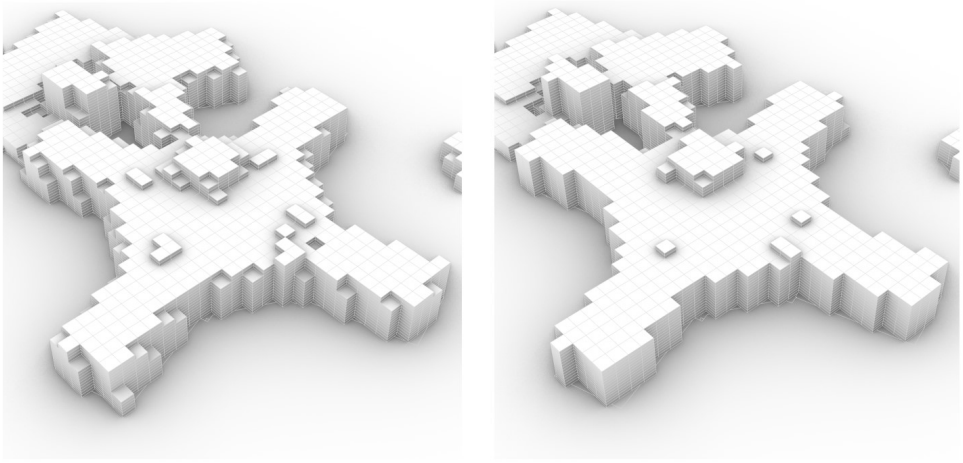


Figure 3.18: Results of voxelization of a ground truth building feature rasterized using bilinear interpolation (*left*) and nearest-neighbor interpolation (*right*)

Raster interpolation It is good to note here that during implementation I discovered the importance of taking into account the image rasterization methods used to generate the raster images in the training dataset in the first place. Most of the modern rasterization libraries use by default some sort of edge smoothing algorithm to avoid *"staircase"* effects created when pixelating non-orthogonal edges (also known as *aliasing* in the field of computer graphics).

Popular interpolation methods are e.g. bicubic or bilinear interpolation, which introduce a gradient of 'in-between' values around the edges of the shape. In my case, these interpolated values are undesirable, since they introduce building heights not present in the real dataset into the picture and 'smoothen' out the building edges. Instead, nearest-neighbor interpolation is preferred when rasterizing the building features (see Figure 3.18).

Vectorization For extracting the LoD 1.1 building geometries using the approach based on the *3Dfier* method, the outline of the shapes present in the raster output is extracted using the *Potrace* algorithm [Selinger, 2003] and, optionally, the complexity of the generated shapes is reduced using the *Ramer–Douglas–Peucker polyline approximation* [Ramer, 1972]. The value of the pixels contained within each shape is remapped to their original height values and the 50th percentile of heights contained within each footprint is computed. Since one can think of the height-map as a representation comparable to a point cloud, this method is basically equivalent to the implementation of the method used to generate LoD 1.1 3D BAG data.

These combined methods ensure that the building geometries used for the training can be well represented in the rasterized format and that, after the model is trained, the geometries can be recovered again from the GAN models output.

Recovering the 3D geometry is crucial for visually evaluating the height differentiation of the resulting massing geometries, since most humans struggle with reading height-map encoded information reliably. Additionally, it is useful for results analysis, since I can apply the same methods used to evaluate the 3D spatial features in the training dataset to the features generated by the network.

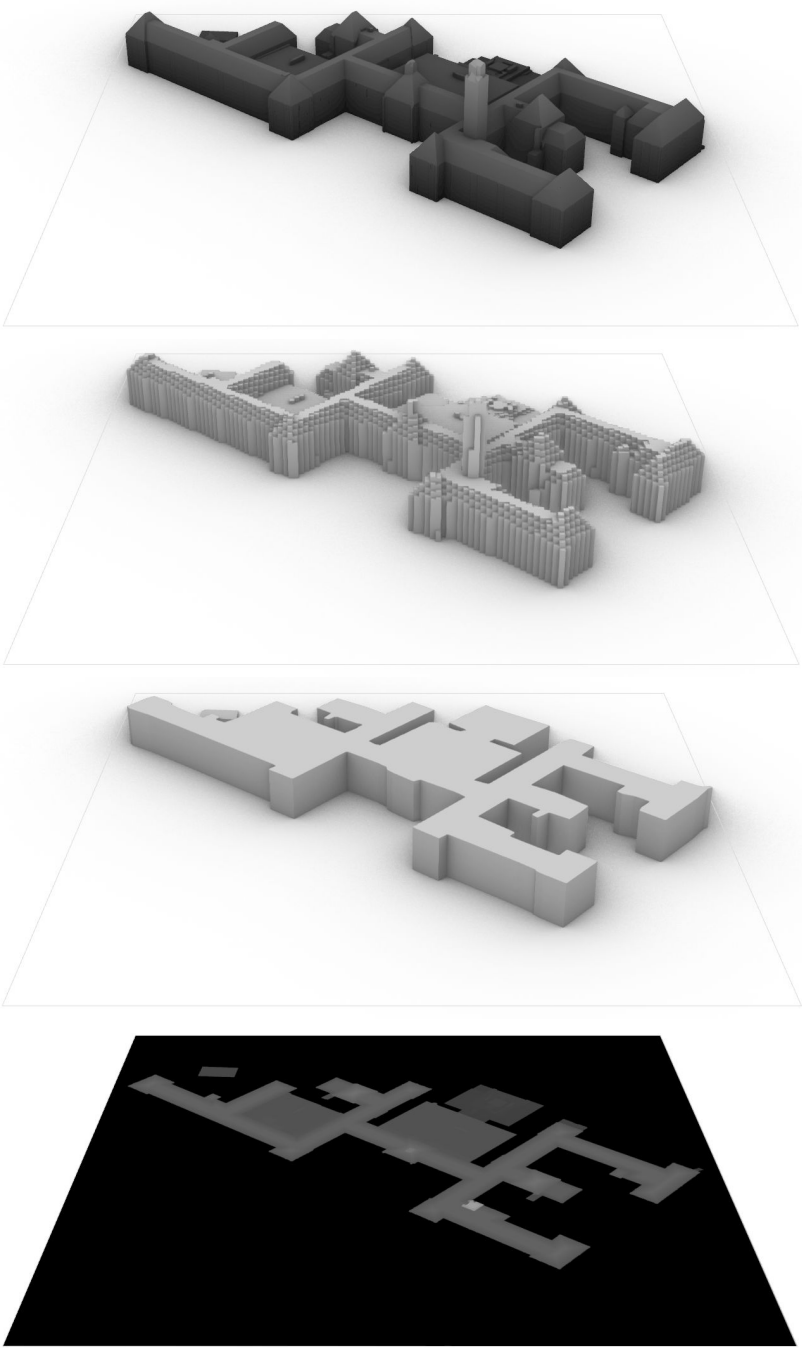


Figure 3.19: Results of geometry extraction pipeline vizualized on LoD 2.2 model of TU Delft, Faculty of Architecture [*from top to bottom: 1. Original 3D BAG model; 2. Reconstructed geometry using the voxelization method; 3. Reconstructed geometry using the* 3dfier *inspired vectorization method; 4. Height-map representation of the geometry*]

3.6 Evaluation

To evaluate the quality of the generated output, I need to define a similarity metric between the training data and the generated results. In field of deep-learning, such similarity metric is commonly referred to as loss function. Defining a universal loss metric for pairs of images (or image encoded geometry data in our case) is still a difficult and open problem in the field.

Traditional error metrics such as RMSE do not work well for image similarity evaluation, since they *"do not assess joint statistics of the result, and therefore do not measure the very structure that structured losses aim to capture"* [Isola, 2017].

GAN Loss In the GAN models, the inclusion of the Discriminator network partially circumvents this problem, thanks to Discriminator effectively learning its own loss function when training to classify between generated and ground truth data. The Discriminator driven loss function is referred to as GAN loss. But since the GAN loss characteristics are a direct result of the training, one needs to adopt additional evaluation metrics to evaluate the training results holistically [Zhu, 2017b].

One approach is to evaluate the results based on a perceptual visual similarity, which can be done using perceptual studies with real human testers such as *Amazon Mechanical Turk*, or CNN based image similarity models such as *Fréchet Inception Distance* [Heusel et al., 2017]. The second, recommended, evaluation approach is to evaluate the model directly in the context of its final application it is intended for [Isola, 2017].

Objectives My goals for evaluation of the model's output in the context of its application are;

- Evaluating and quantifying the spatial characteristics of the building massings generated by the model
- Proving or disapproving the hypothesis that if the model trained on buildings with certain traits, these traits will be maintained in the generated output

Spacematrix *Spacematrix* method [Pont and Haupt, 2007] is a framework for embedding urban morphological typologies into two-dimensional space, based on quantifiable variables describing the urban form, such as density, compactness, amount of open space or building heights. These variables are represented by a combination of 3 metrics, already introduced in the previous chapter - FSI, GSI, OSR, (see Equation 3.1, Equation 3.2, Equation 3.3) and additionally Levels (L), describing the mean number or floor levels in an urban area.

The *Spacematrix* embeddings are shown to correlate well with general recognized classifications of building layout typology [Berghauser Pont and Haupt, 2009] (see Figure 3.20) and can be used as a quantitative method for classification of the urban morphology of existing urban structures [Ye and Van Nes, 2014].

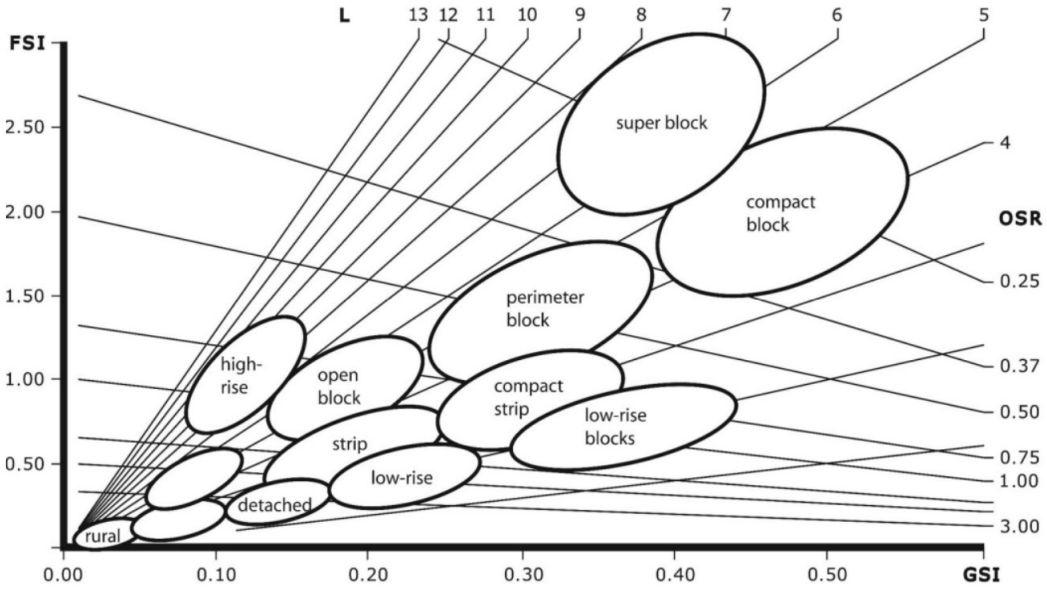


Figure 3.20: Urban morphological types mapped as their Spacematrix encodings (image from Pont and Haupt [2007] The relation between urban form and density)

Conclusions To evaluate and describe the traits of building massings present both in the training data and massings generated by the GAN model, the *Spacematrix* embeddings of both the ground truth and generated data are computed and compared. The goal is for the trait from training data to be replicated in the generated data.

Since the chosen evaluated metrics depend both on the building massing itself, and its relation to the site context, this way such evaluation can validate or disprove the hypothesis that the model is able to react to the site context and generates solutions that respect the massing's general site relations to the surrounding buildings.

Strong correlation between the *Spacematrix* variables for the training and generated data would mean the model is able to infer the spatial properties of buildings it was exposed to. Therefore the massings model generates should be steerable towards certain morphological characteristics by curating the input designs it is trained on.

3.7 Flowchart

Figure below contains a diagrammatic overview of the proposed methodological framework.

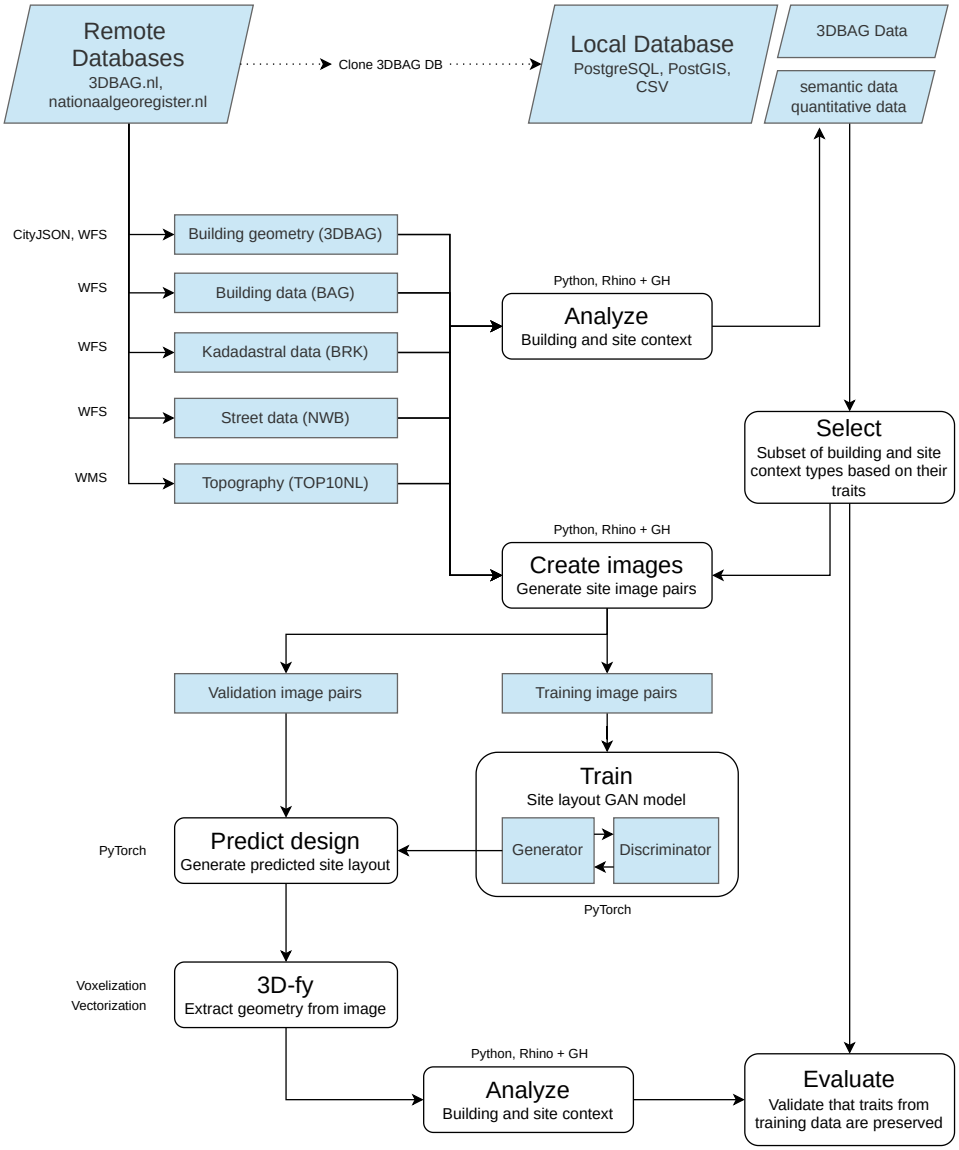


Figure 3.21: Flowchart diagram of the proposed methodology

4 Implementation and experiments

In this chapter I will further elaborate on the implementation details on the above described methodology, introduce any new findings or learning I made while implementing the methods and developing this research work, and present the overview and a description of the various experiments I designed and executed using my methodological framework.

4.1 Tools used

GAN The GAN model is implemented in Python using PyTorch library, a deep-learning framework allowing for GPU accelerated training [Paszke et al., 2019]. The code for the Pix2Pix GAN model is mostly based on the implementation developed as part of the CycleGAN/Pix2Pix project by Zhu [2017b]. I also experimented with using BicycleGAN [Zhu et al., 2017], a derivative model adding the option for one-to-many input-to-output mapping to the original Pix2Pix implementation, using the implementation provided by [Zhu, 2017a].

Feature processing The data retrieval, feature pre-processing and geometry reconstruction was implemented in *Grasshopper*, a visual programming extension for *Rhinoceros 3D* modeling software. Most of the nodes were developed during the course of this research as my own custom scripts written in *IronPython*. These include the methods for querying databases via WMS and WFS, and parsing the *GeoJSON* and *CityJSON* responses into *Rhinoceros 3D* geometry types. The rasterization of spatial features and raster postprocessing was implemented using the *Bitmap+* [Mans, 2021a] and *Graphic+* [Mans, 2021b] libraries for *Grasshopper*.

Data processing The processed data were collected in CSV files, which were linked to a structured collection of PNG files containing raster data layers associated with each data point. The collected training data exploration and filtering tools were implemented as *Python Notebooks*, using *Pandas* library for tabular data processing and *Pillow* library for image manipulation.

Hardware As a point of reference for computation time mentioned in the following sections, all data processing and GAN training has been run on a Windows PC with following configuration.

1. *CPU:* Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz (4 cores)
2. *GPU:* NVIDIA GeForce GTX 1050 Ti MaxQ (4GB VRAM)
3. *RAM:* 16 GB

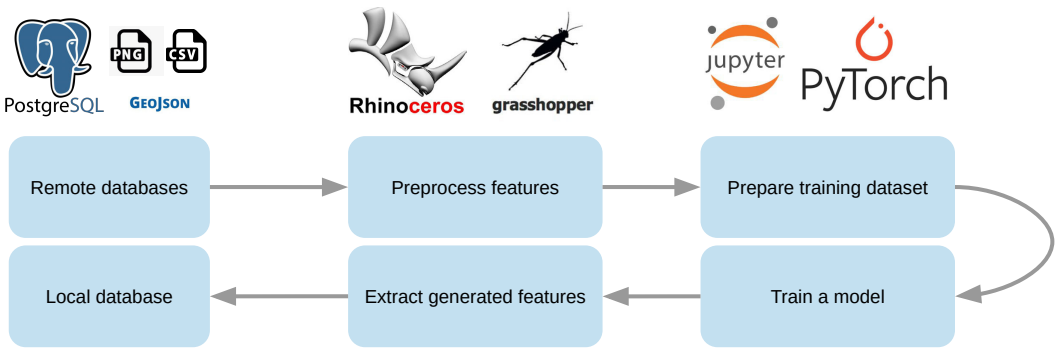


Figure 4.1: Overview of software stack used at different steps of the process

The code and scripts developed during the course of this project are available on Github at `https://github.com/ondrej-vesely/massingGAN`.

4.2 Data collection

APIs and formats Data on current building stock in the Netherlands were collected from multiple open data sources using multiple standards and protocols (please refer to Section 3.1 3D building models and Section 3.2 Other input datasets for overview of used datasets). While the 3D BAG geometry could be downloaded from the 3D BAG APIs using i.e. CityJSON [Ledoux et al., 2019] or OBJ encoded tiles of ca. 3500 buildings, to make the process of retrieving the LoD 2.2 building models faster, I store the building geometries inside a local copy of the 3D BAG PostgreSQL database instance and retrieve them as GeoJSON encoded features via local SQL queries. The LoD 1.2–1.3 3D BAG is requested GeoJSON encoded features via WFS from 3D BAG servers directly. A WFS is used as well for retrieving data openly available in a feature (vector) format, such as cadastral plot or street network geometry. Additionally, WMS are used to retrieve data available only in the map (raster) format, such as topographical maps. I implemented the methods required to query and parse such data in the software without the native capability to do so (i.e. in *Rhinoceros 3D*).

Local database and filtering As a primary key for storing collected data points in the database I use the *BAG Pand* number, an unique identifier assigned to every building object in the Netherlands. When adding data points to the local database, I begin with querying the BAG database for features matching the desired properties. This is done using XML encoded filter queries, based on the Open Geospatial Consortium (OGC) *Filter* standard, which can be passed to the BAG service as part of the WFS request in the Filter parameter. The OGC *Filter* standard allows use of numerical or lexical comparison operators on the feature property fields and spatial operators on the feature associated geometries to select the kind of features to be returned.

The BAG service returns features based on their order in the database, with the database itself being ordered in ascending order of the BAG identifiers. Since the lower BAG numbers

are usually assigned to the buildings in the northern parts of the Netherlands, one can notice a bias in the number of data points collected increasing towards the north (see Figure 4.2). This is indeed the case, since I often collected data only for a part of the returned BAG feature lists, unintentionally collecting northerly data points and omitting the more southern regions. This bias can however be eliminated when creating the final training dataset by specifying the spatial extents of specific areas from which I intend to sample the data points.

Results During the course of the research, I collected data for 67,576 unique BAG features, with some of them being collected multiple times at different scales. On average, I stored 12 raster maps and 30 additional statistics per data point (see Figure 4.3). I was able to scrape around 500 data points per hour, with slight variations depending on the size of the area of interest. That corresponds to approximately 135 hours of data collecting time to build the final dataset.

4.3 Training and postprocessing

GAN architecture During the course of the research, I experimented with two specific open-source implementations of the image translating GAN frameworks - Pix2Pix GAN [Zhu, 2017b] and BicycleGAN [Zhu, 2017a]. I was interested in training BicycleGAN on the same training data and comparing the results to my pix2pix implementation. BicycleGAN is a GAN model partially derived from the original Pix2Pix GAN code. Its goal is to allow for mapping single input to multiple outputs, instead of the deterministic one-to-one mapping generated by Pix2Pix GAN. It achieves this by injecting a latent code (random z-vector) into the generator.

While I managed to get BicycleGAN behave as expected during training on the example datasets provided by the framework's authors, for my own datasets I observed almost no changes to the generated outputs while traversing the latent space (see Figure 4.4). I therefore postponed further research into this model, but plan to revisit this architecture at a later date as a potential topic for future research.

Generator architecture I experimented with different CNN architectures used as the generator network in the Pix2Pix GAN model, the U-Net 256 [Ronneberger et al., 2015] and ResNet-9block [Hesse, 2017]. When comparing the result obtained using the two Generator architectures, I found that while ResNet in some cases generates more regular and "orthogonal" layouts compated to U-Net (see Figure 4.5, row [a]), in other cases the ResNet model 'collapses' and tends to fill the whole plot with a single massing lacking any defintion (see Figure 4.5, row [b]).

While I haven not analyzed the reasons for this behaviour further, these findings due not align with the findings of a similar research project — GANMapper by Wu and Biljecki [2022] — where the two generator architectures are compared in more detail and a conclusion is reached that ResNet is clearly superior for use on geographical data. Therefore more research is needed to trully evaluate which Generator architecture performs better. For the purspose of this thesis I decided to use U-Net 256, since is the architecture that Pix2Pix GAN by Isola et al. [2017] was originally implemented with.

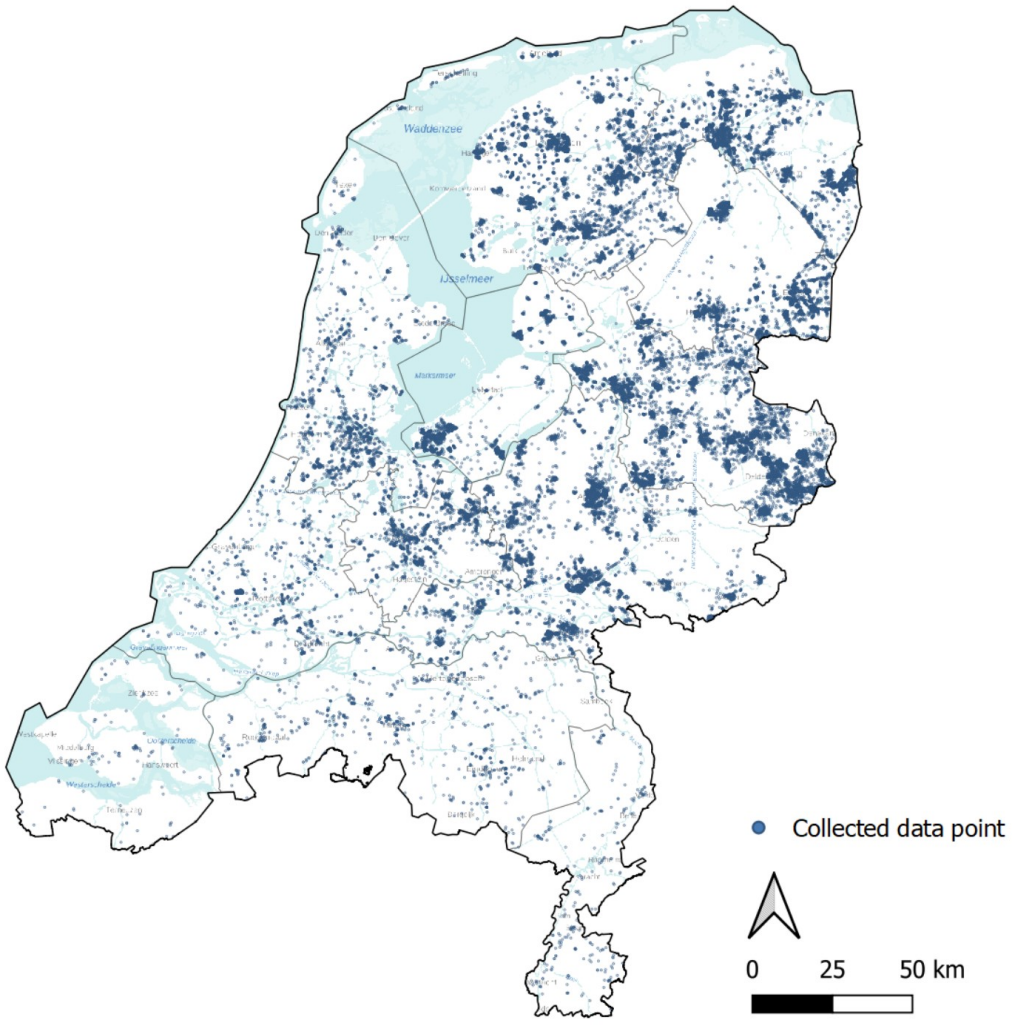


Figure 4.2: Plot of locations of 67,576 data points collected during the course of the research.

ID	0106100000008139
X	235679
Y	559714
MainBldgArea	95.7709
MainBldgVolume	577.293
MainBldgPerimeter	48.3347
MainBldgAge	1993
MainBldgUse	woonfunctie
MainBldgDakType	slanted;horizontal
ParcelArea	309.608
InParcelArea	115.822
InParcelVolume	597.387
InParcelCount	3
InParcelAge	1993-2003-2013
InParcelUse	woonfunctie
BlockArea	10021
InBlockArea	2864.54
InBlockVolume	16027.2
InBlockCount	41
InBlockAge	1991-1998-2018
InBlockUse	woonfunctie;winkelfunctie
InSubframeParcelAmount	1
InFrameBlockAmount	1
InSubframeBlockAmount	0.76
InFrameArea	10946.7
InFrameVolume	60155.4
InFrameAge	1991-1997-2021
InFrameUse	gezondheidszorgfunctie;woonfunctie;winkelfunct...
InFrameParcelAmount	1
IsMainBldgInFrame	1
AllImagesValid	True
Path	C:\data\training-images\v4\bloemkool\raw\
Size	256

Figure 4.3: Database entry for a typical data point

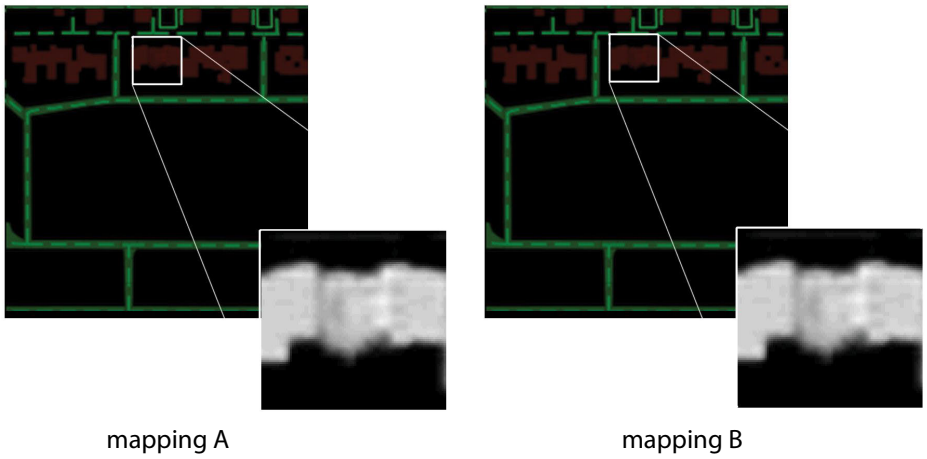


Figure 4.4: Results of the experiment with using BicycleGAN

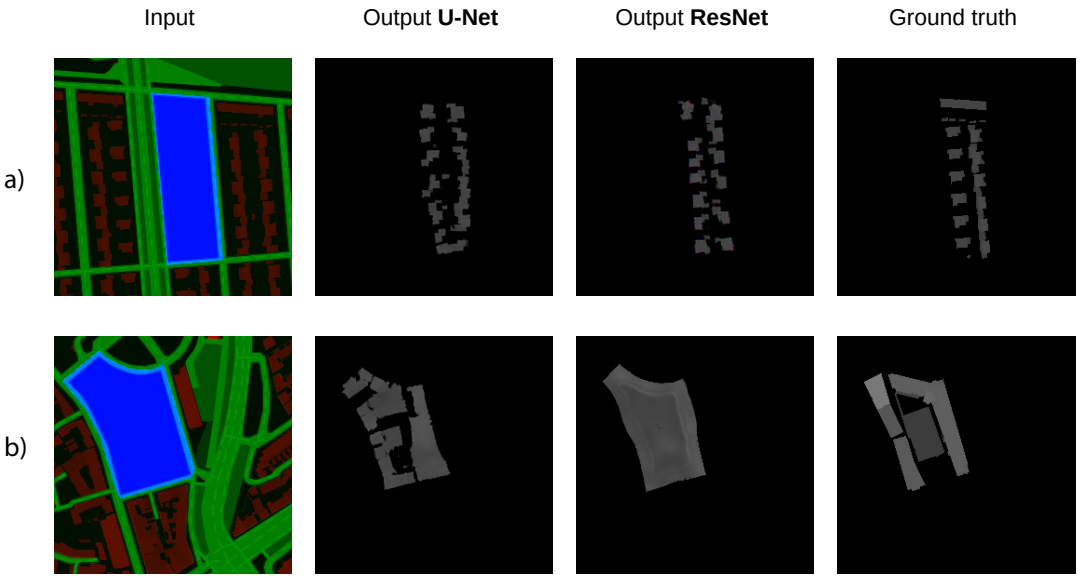


Figure 4.5: Comparison of results obtained with different Pix2Pix GAN Generator CNN architectures

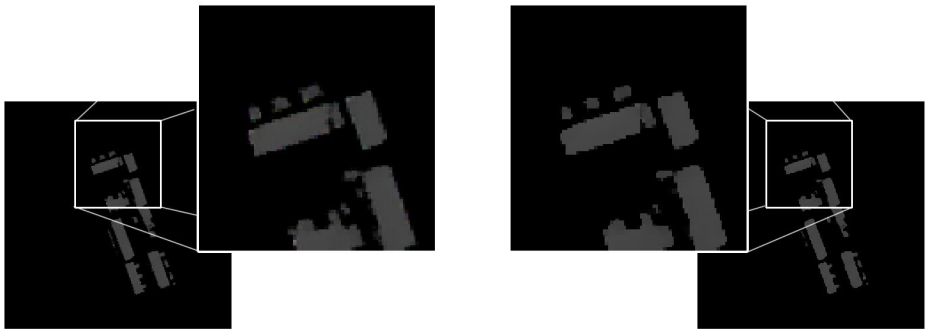


Figure 4.6: The example of the effect of the adaptive smoothing filter on height-map noise removal

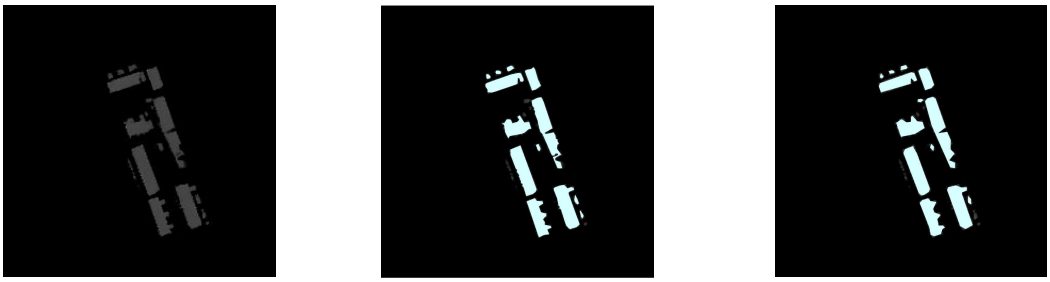


Figure 4.7: Example of the vector feature reconstruction pipeline used on one of the samples

Noise removal To remove some remaining noise present in the generated images, I decided to use an *AdaptiveSmoothing* image filter from *Accord.NET* computer vision and image processing library. The filter is based on a method presented by Matalas et al. [1997] and aims to remove image noise while trying to preserve any edges detected (see Figure 4.6). This filter is only implemented as part of the output processing pipeline used before the generated output is voxelized or vectorized, and is not included in the images presenting the examples of the the raw height-maps generated by the GAN model in the GAN output figures in this work. Additionally, any height-map accuracy calculations and statistics are calculated without this filter applied.

Vectorization For the 3Dfier process, the feature contours are traced with a *Potrace* algorithm [Selinger, 2003], used in *Grasshopper* via the *Bitmap+* image processing package. I further simplify the generated feature polygons using the *Ramer–Douglas–Peucker* polyline approximation [Ramer, 1972], with a tolerance value of 1.0 m. This number has been found to be effective at removing redundant vertices from the image tracing algorithms output in most of the cases encountered during experimentation.

Training times As a general size of training datasets and training time, I aimed at generating at least 1000 training images per experiment, if possible, and letting the model train for 200 epochs (epoch is a measurement of how many times the model has visited each training sample during its training). On the used hardware, such training with RGB image pairs at 256x256 resolution completes in under 11 hours. With larger datasets, I observed that one can proportionally lower the amount of epochs the model trains for to get reasonable results, although letting the training for the full 200 epochs can be still beneficial.

Inference times After the trained model is loaded into memory, the inference time for generating a single output for a given input image on the GPU is around 100ms. This means the time required for generation of an output is within the necessary speeds for real-time application without a user experiencing significant latency. The current bottleneck in my implementation is the geometry extraction pipeline, which takes around 0.5-1.0 seconds for each 256x256 pixel image - both for the voxelization and the *3Dfier* extrusion method. I believe this part could be optimized for much higher computation speeds - although this is currently outside the scope of this research.

4.4 Experiments

In this part I will introduce several selected experiments I implemented using the developed methodology, and will present an overview of results for each of them. The selected experiments differ in the kind of information the model receives in the input training images and what information it learns to fill in from the desired output images, and the kinds of urban situations and morphologies captured in the training data.

The goal of these experiments is to compare the performance of the model in various applications and help answer the research questions of at what scales, levels of detail and kinds of context the proposed model is applicable.

By querying the local database built during the data collection process, all collected spatial data can be selectively retrieved based on its scalar and semantic properties, which allowed me to design a diverse set of test cases to validate my methodology on.

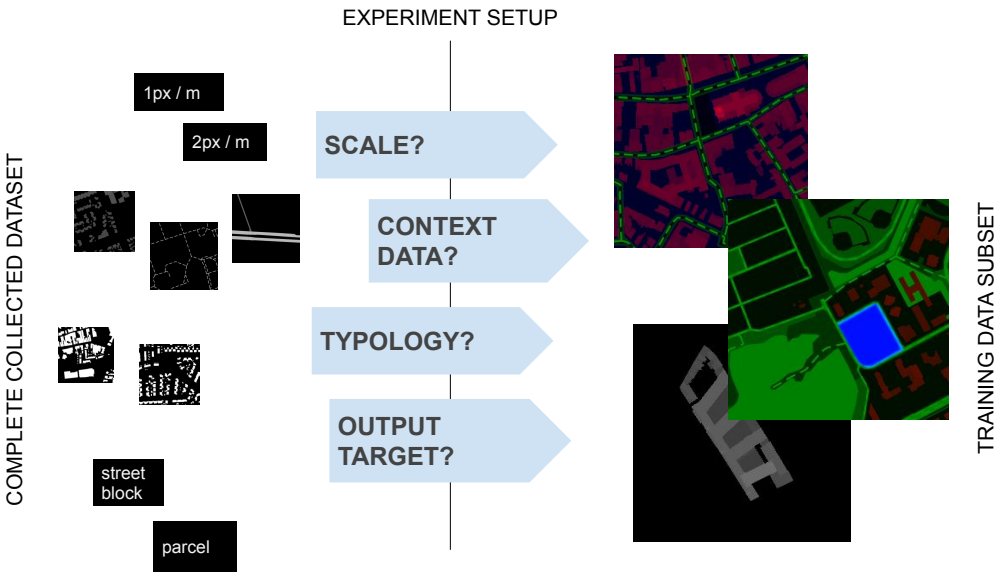


Figure 4.8: Conceptual overview of the data experimentation framework

4.4.1 Single residential building generator

In this initial experiment, I wanted to establish a baseline for further tests. I focused on minimizing the problem space and reducing the number of variables. Therefore I decided to focus on the smallest and most granular scale captured in the database: a single massing of a building using its parcel as an area of interest for the model to fill in.

The input images included only the spatial information on the parcel where the buildings should stand on, the surrounding building massings, and the street network in the area.

To capture as many data points in the training dataset as possible, I aimed to define the target building as a well represented feature in the collected data. The selection criteria is designed to target single mid-size residential buildings, defined as follows;

1. Only residential use (*'woonfunctie'*)
2. Built between 1980 – 2010
3. GFA between 200 and 400m^2

I found around 27,674 data samples fulfilling the criterion above, with 150 samples removed from this number as evaluation samples that the model does not have access to during training. The training was stopped after 25 epochs. With such a large dataset the training took around 1h 20mins per epoch. Longer training with a similar dataset might be done at a later date to test if the results can further improve. See Figure 4.10 on the next page for an overview of typical results.

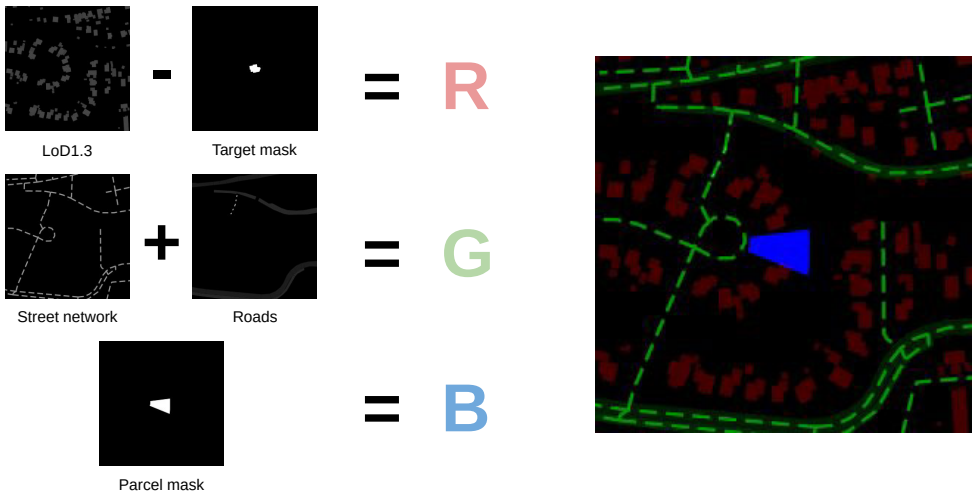


Figure 4.9: The composition of the training input images for Single residential building generator

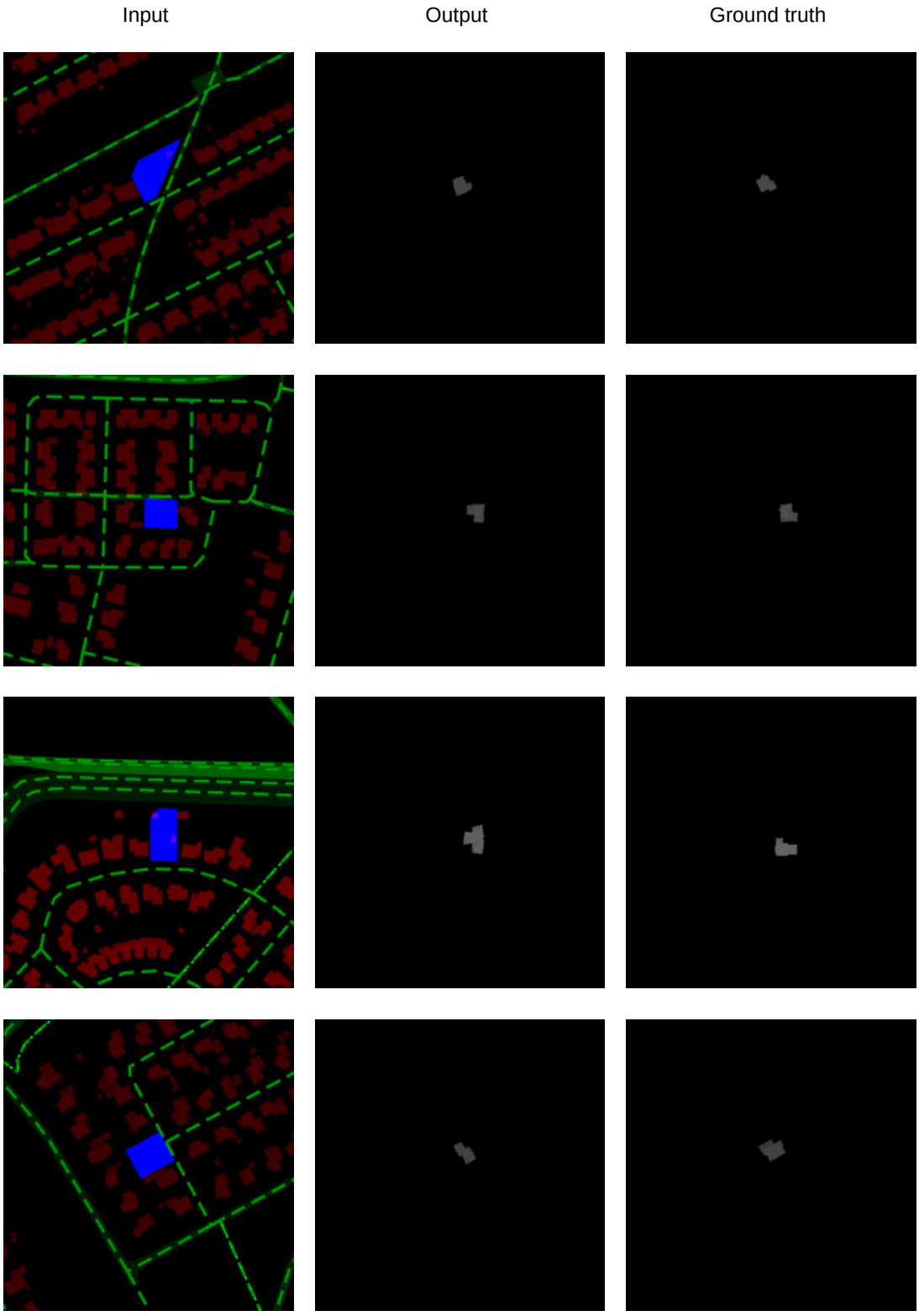


Figure 4.10: Examples of generated outputs for validation samples from Single residential building generator

4.4.2 Residential street block generator

The purpose of the second experiment is to test whether I could replicate the results of the first case study on a scale of large building configurations. As target data I chose the same kind of building typology, a residential mid-sized house, but this time in a situation where the buildings form into a distinct street block. Selection criteria are defined as follows;

1. Only residential use (*'woonfunctie'*)
2. $\geq$ 16% of the whole frame covered by buildings
3. Block built between 1990 – 2010
4. Block buildings have surface between 200 – 1000m^2 (*'oppervlakte'*)

The model receives all the information that I used in the previous experiment, but this time I added further site-context in the form of land-cover, natural landscape feature data, and road extents data. And, crucially, in the input image's blue channel used for encoding the area of interest I replace the parcel boundary with the boundary of one of the street blocks detected.

I generated exactly 1,000 data samples fulfilling the requirements described which were rasterized into 256x256 images at scale of 1 m/px, with again around 5% of additional data points added as evaluation samples. The model was trained for 200 epochs. See Figure 4.12 for an overview of typical results.

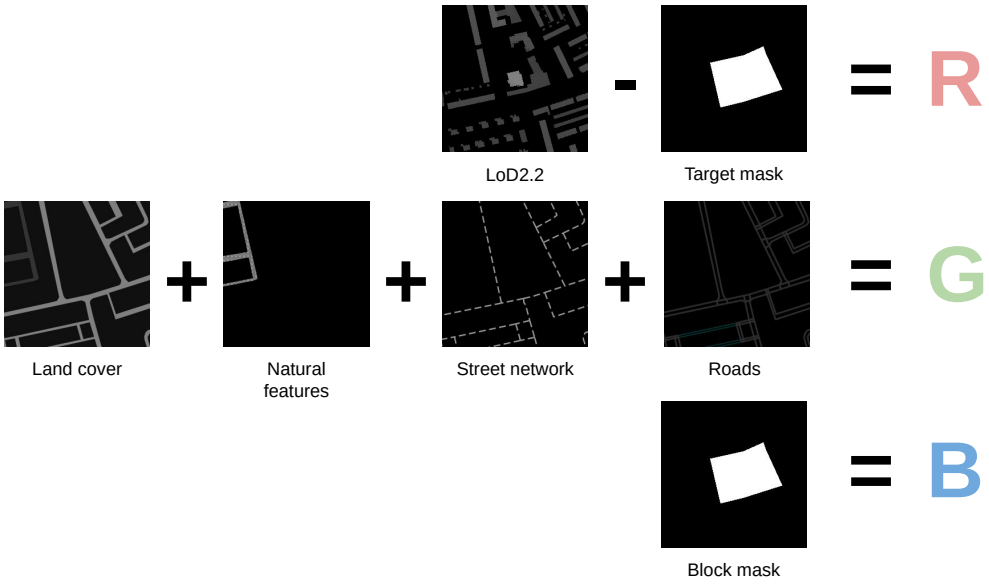


Figure 4.11: The composition of the training input images for Residential street block generator

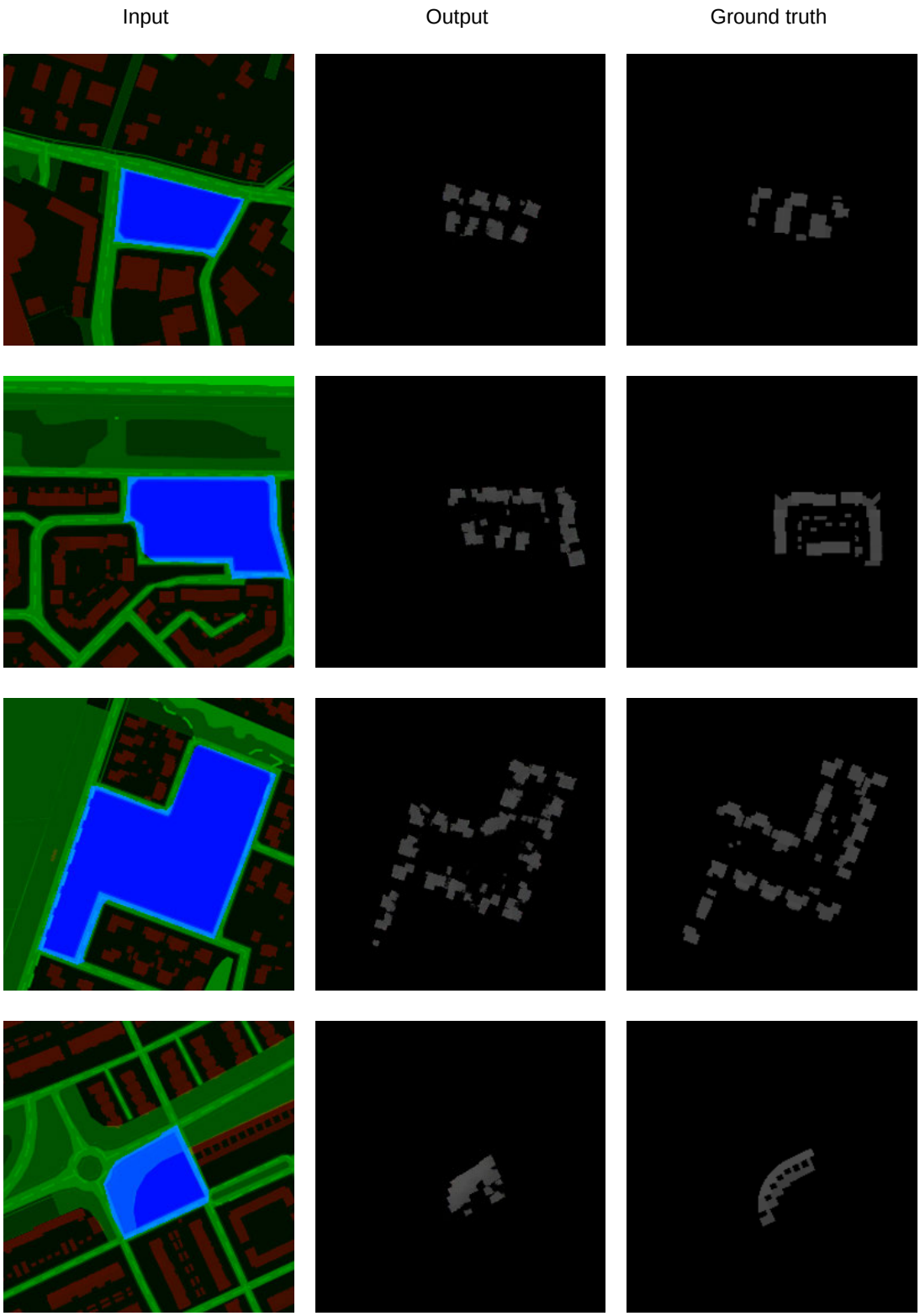


Figure 4.12: Examples of generated outputs for validation samples from Residential street block generator

4.4.3 Bloemkoolwijk street block generator

With the third experiment I planned to explore the diversity in morphological typology the model is able to generate, based on which kind of specific archetypes it is trained on. I decided to train only on examples of specific late-20th–century types of residential neighborhoods, familiarly known as *Bloemkoolwijken* in Dutch, or "cauliflower" districts in English. These neighborhoods, built mainly between 1970–1990, have a distinct structure to their street network which gives them their name.

I defined the target data as residential building street blocks similar to the previous generic residential building dataset, with the exception of spatially limiting the area from which data are retrieved to extents of typical *Bloemkoolwijken - Kronenburg* in Arnhem, *Peelo* in Assen and *Holy-Noord* in Vlaardingen (see Figure 4.13). Same as in the previous experiment, I generated exactly 1000 training data points, with additional 50 reserved for evaluation. The selection criteria are:

1. Within AoI (see above)
2. Block contains residential use (*'woonfunctie'*)
3. Area encompassed within the block $\geq$ 2000m^2
4. FSI inside the block $\geq$ 1

In Figure 4.14 you can see results generated by both the *Bloemkoolwijk* model and the generic residential model applied to the sites never seen before by either of the models, but matching the training data in density and type of use.

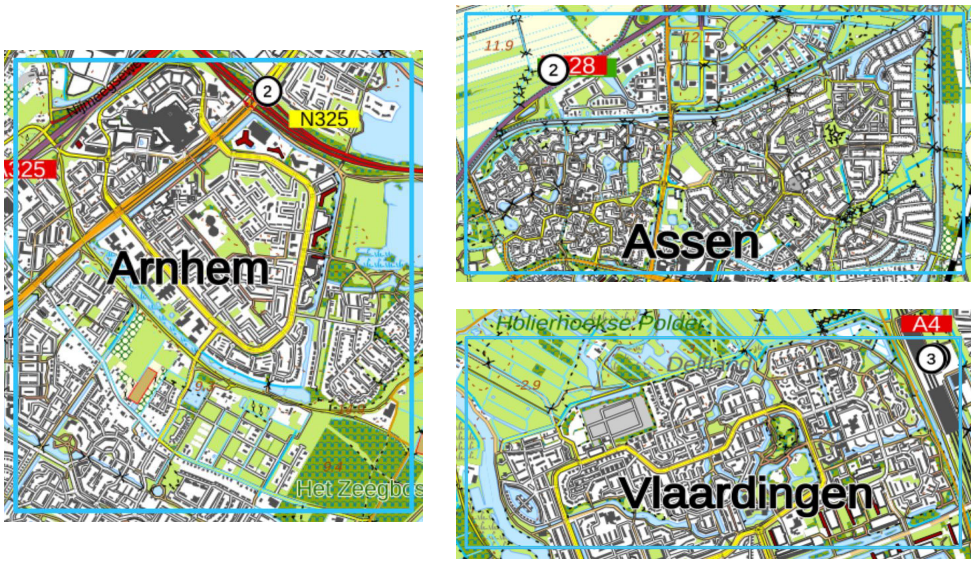


Figure 4.13: Spatial extents of areas explored in Bloemkoolwijk street block generator

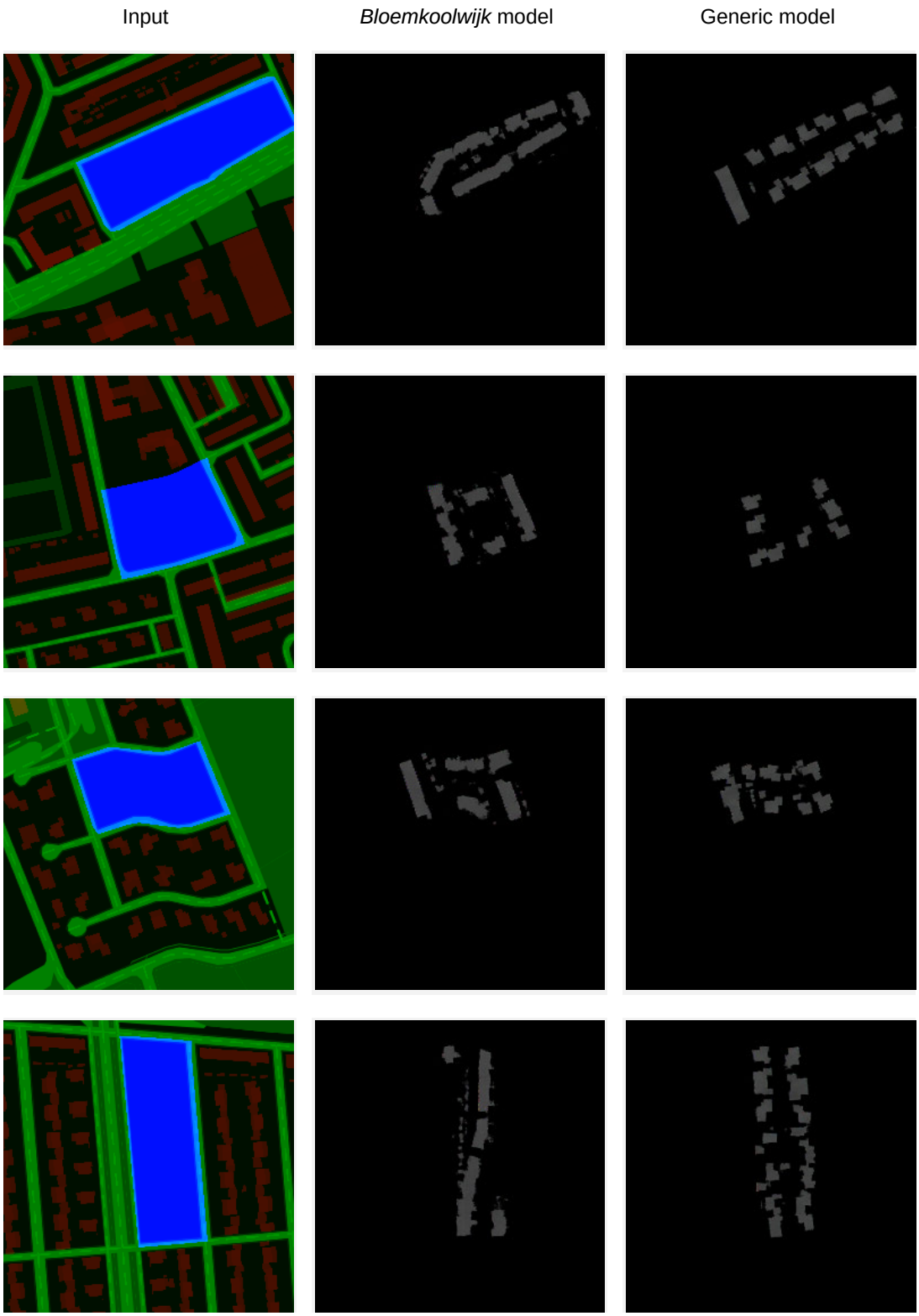


Figure 4.14: Examples of generated outputs for validation samples from Bloemkoolwijk street block generator

4.4.4 Urban fabric densification

The purpose of the fourth experiment is to evaluate the model's ability to identify patterns in urban fabric and to fill the existing gaps, with a goal of training a model capable of proposing changes to density of an area by a given ratio. I have defined as target data for generation of training data samples as dense urban centers, defined as;

1. Containing buildings with GFA between 500 – 1000m^2
2. Containing buildings built between 1990 – 2010
3. One of the uses must be office (*'kantoorfunctie'*)
4. FSI of whole area ≥ 0.5

I mask a certain amount of buildings within the area using the density reduction masks generated during feature preprocessing. As already mentioned in chapter XX, the density reduction mask is a binary mask of a randomly selected subset of all buildings, with the subset having reduced FSI-measured density compared to the original layout - with the density reduced to 90, 75 and 60% of value. In the results below, I used the 75% density reduction mask.

Other than the existing massing, the model also receives the land cover, street surface, natural landscape features and street axis data as part of the input raster data. See Figure 4.16 for an overview of representative results for this model.

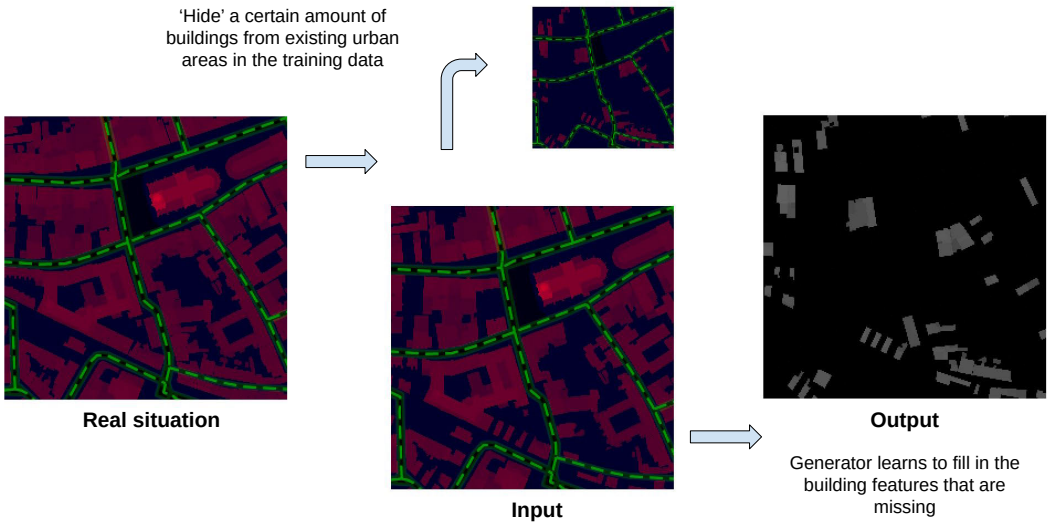


Figure 4.15: Visual overview of the concepts behind the Urban fabric densification

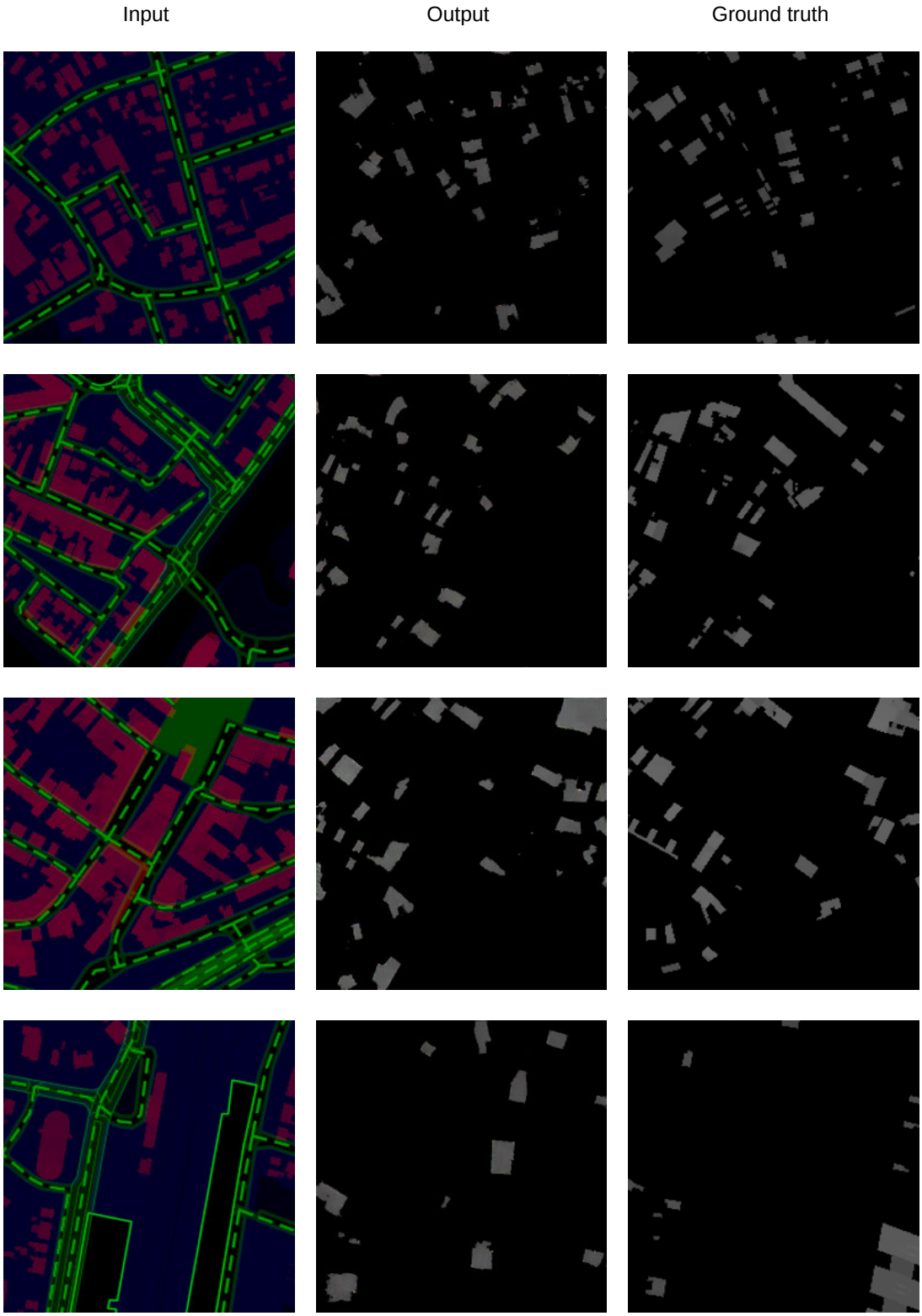


Figure 4.16: Examples of generated outputs for validation samples from Urban fabric densification

4.4.5 Study on scale

In the fifth experiment I wanted to focus on a more spatially intensive building typology than in the previous test block generation experiments, to test how the model copes with larger building scales. I chose large office park blocks as a target context. The goal of this experiment is to compare how the model behaves with lower feature resolution. The target sites are defined as any block with the following selection criteria;

1. Containing building with surface $\geq$ 1000m2
2. Containing building built between 1990 – 2014
3. One of uses is office ('kantoorfunctie')
4. Total area encompassed within the block $\geq$ 4000m2
5. FSI inside the block $\geq$ 1.0

Both models are trained on exactly identical 1000 data samples with raster maps exported at resolution of 256x256pixed, the only difference consisting in the scale of the data capture, with lower scale version at 512x512m (2m/pix) and higher scale at 256x256m (1m/px) (see Figure 4.17).See Figure 4.18 for an overview of representative results, with the results for both scales included above each other, first for 2m/px scale, followed by 1/px scale.

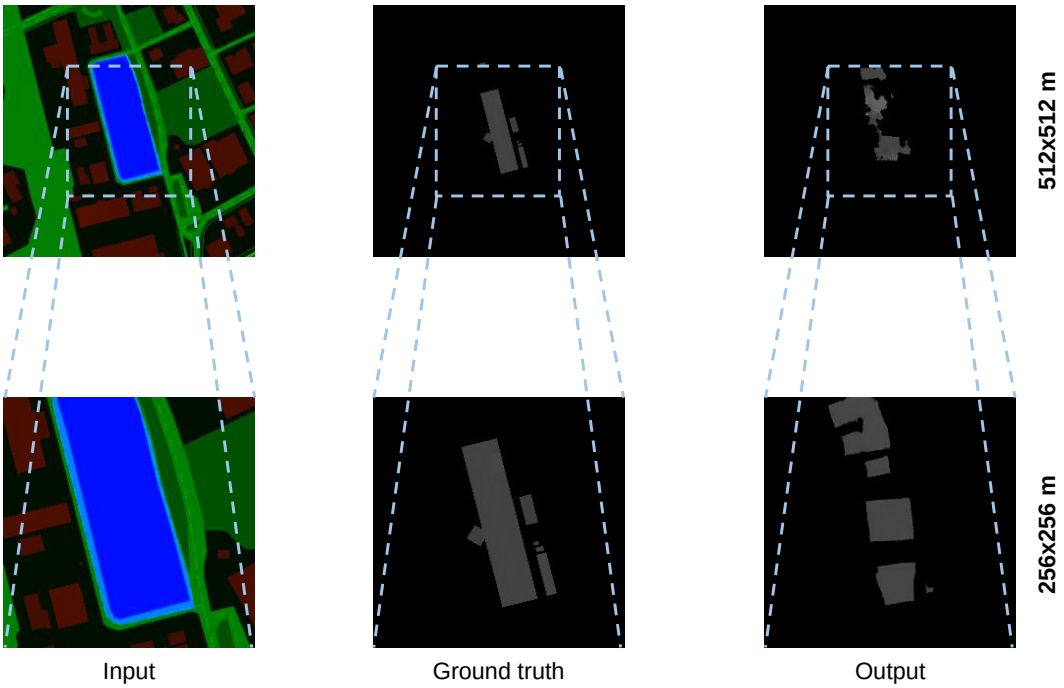


Figure 4.17: Same locations generated at two scales for the purpose of Urban fabric densification

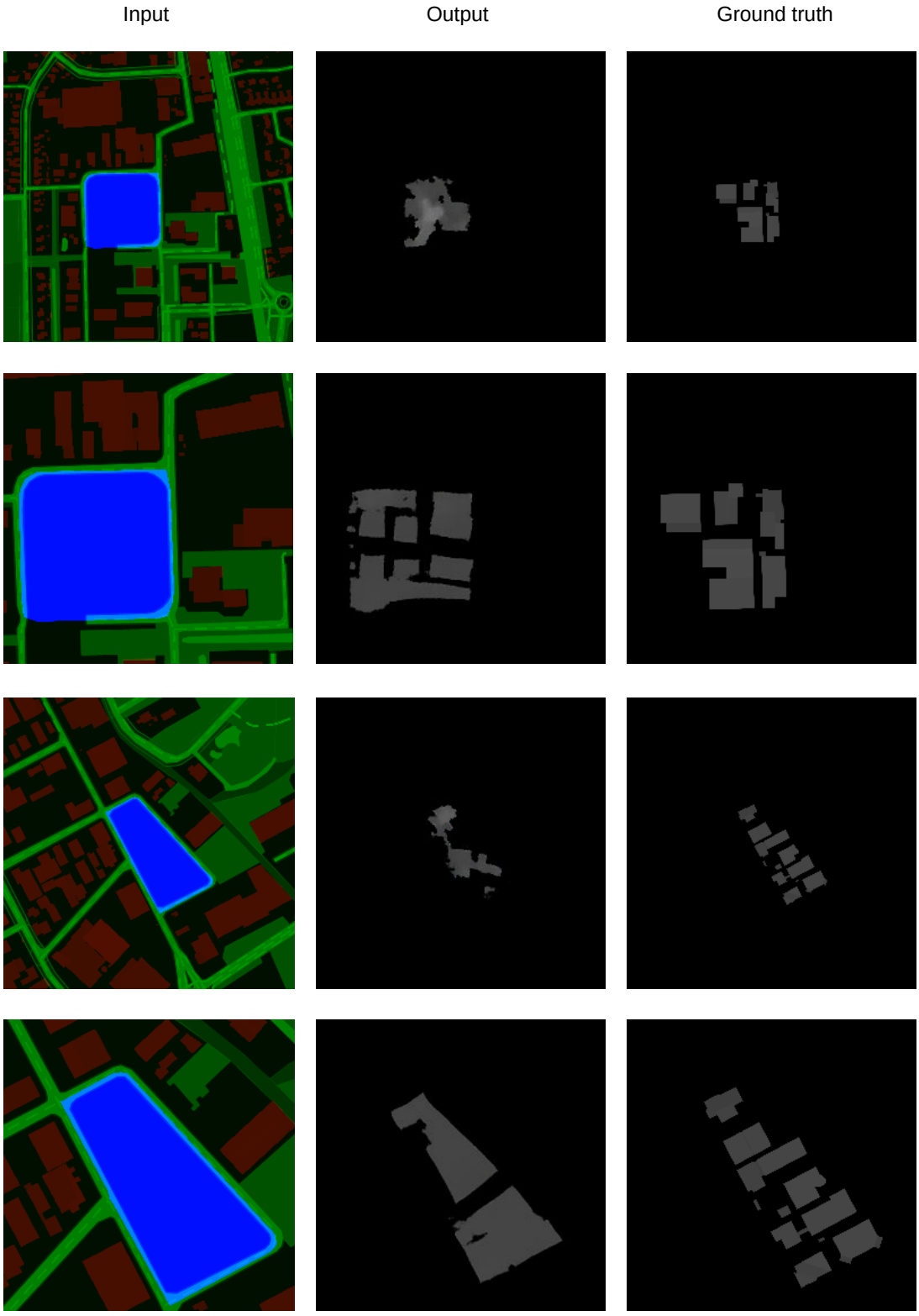


Figure 4.18: Examples of generated outputs for validation samples from Urban fabric densification

4.4.6 Study on level of detail

In the sixth experiment I aim to evaluate the importance of the level of detail of the building 3D geometries used in the training and as the models input data. The type of situations the models were trained on were identical to the residential street blocks used in Residential street block generator on page 56, but in this instance, each data point is generated at all 3 different LoD available through the 3D BAG service - LoD 1.2, LoD 1.3 and LoD 2.2.

The goal of this experiment is to help answer the question of what LoD of 3D building geometry is necessary to replicate the results presented in this work and thus, in what contexts can such models be applied.

To ensure the comparability of the results, the three models were using the identical training data sets of 1000 images, each image containing the exact same location and information, with only difference being the LoD of the building massing height-map. All three models were trained for 200 epochs. However, it should be noted that training of a Pix2Pix GAN is a non-deterministic process due to the random initialization of the network weights at the start of each training. Therefore, even when comparing two Pix2Pix GAN trained for an identical number of epochs on fully identical datasets, one can expect at least slight differences in the outputs of such two models.

In Figure 4.19 you can see the overview of the differences between the three LoDs and in Figure 4.20 an overview of representative results for each model is provided.

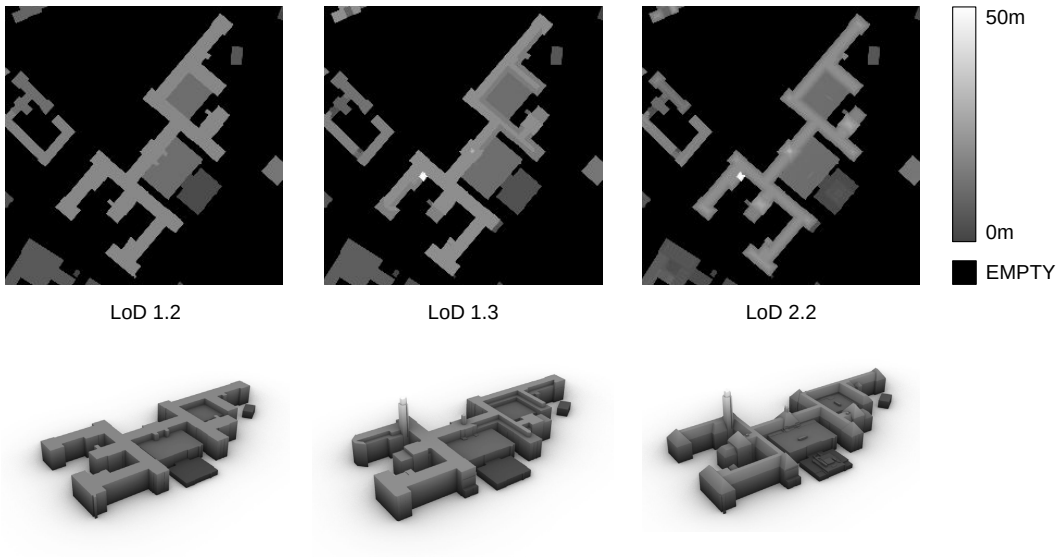


Figure 4.19: Height-maps for the same location (TU Delft, Faculty of Architecture) generated at different LoDs (color grading exaggerated for clarity - in the real training data, white corresponds to 100m height)

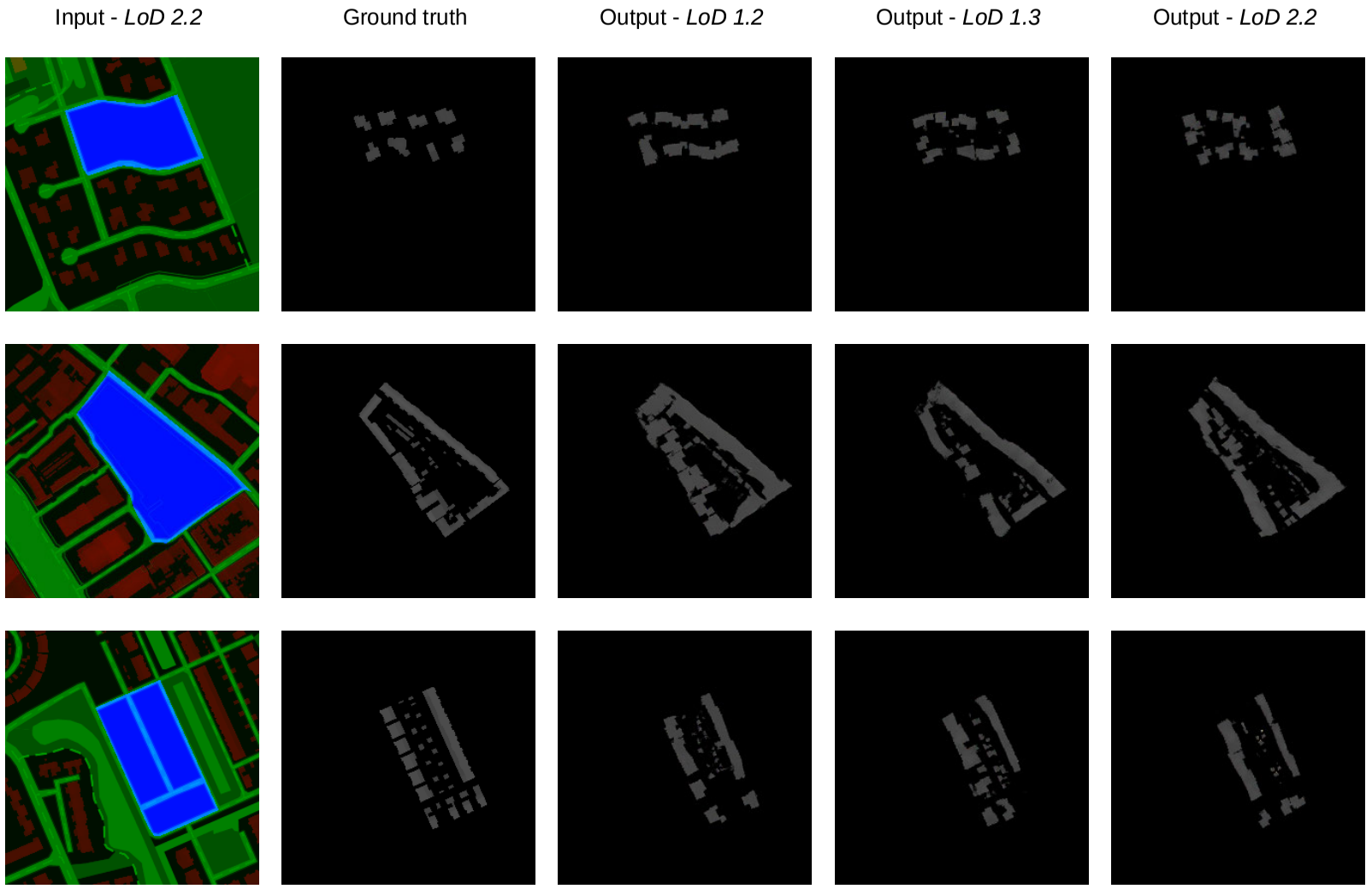


Figure 4.20: Examples of generated outputs for validation samples from Study on level of detail

5 Results

In this chapter I will summarize my observations and interpretations of the results from the experiments conducted and conclude some general findings about results generated using my implementation of the proposed methodology. The findings mentioned in the Section 5.1 - Visual analysis are based on my intial visual analysis of the generated results and interpretation of the behaviour of the models based on the experience with methods gained during the experiments. More detailed statistical analysis of the results is included in the next Section 5.2 - Similarity statistics, where I apply statistical methods to the generated outputs to quantify the similarity of the generated outputs to the training data.

5.1 Visual analysis

Single residential building generator The model trained for generation of a single massing on a single parcel (page 54) was trained on the highest amount of training samples of all models trained during this research, which is probably the reason why can one obsere better definition in the shapes generated by the model compared to the other experiments. The model seems to be capable of generating properly sized massing for the context they are in and properly keeps setbacks from other buildings.

One limitation of the results generated bt this model is that — since the target building in the training data was located in the center of the frame — the model learned to always predict the generated volume in the center of the frame as well. This reduces the diversity of the results. The model has less freedom over the placement of the building massing inside the boundaries of the plot. On the other hand, this can be used to actually steer the model towards placing massing at a specific position, which can be seen as a useful feature.

Although it is hard to argue as to why the model generated these specifying massing in each case, the results look convincing and exchangeable for ground truth at first observation. Even when the generated height structure is noisy, in general the model manages to match the heights of neighboring buildings quite well. I will elaborate upon this point in the next Section 5.2 - Similarity statistics, when analyzing the height distributions of the massings generated by the model.

Residential street block generator The second experiment (page 56) was focused on extending the typology from the previous experiment to full street blocks. The street block in this case is defined as any area border from each side by a named street or road, although a criterion on a minimum density of such block was placed on the data extracted for the training dataset. The resulting model trained on only 1000 data samples (without any data augmentation) seems to be capable of generating realistically looking building arrangements, respecting the boundaries of the site and the scale of surrounding buildings.

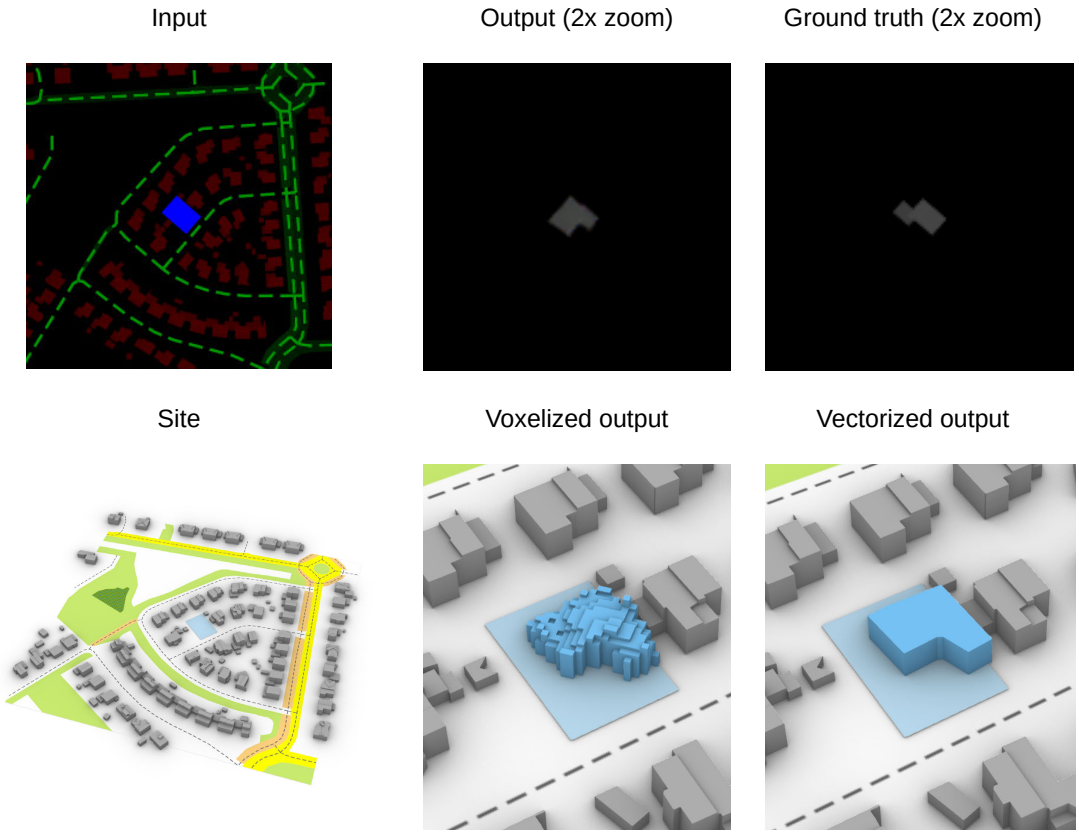


Figure 5.1: Massing generated for a single parcel from validation dataset for Single residential building generator

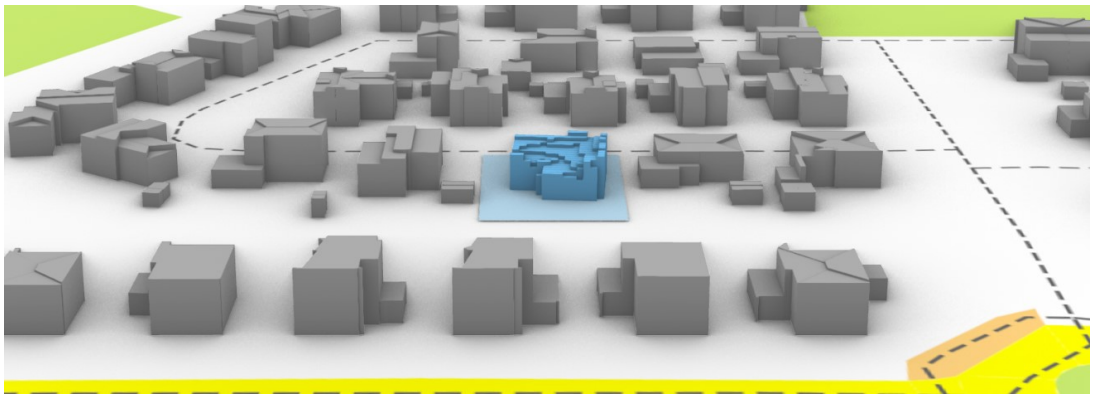


Figure 5.2: Rendering of voxelized result from Single residential building generator

The model also seems to be able of observing massing patterns present on the site and replicating them (see Figure 5.4). However, I could definitely observe that the model has a tendency to generate free standing buildings, even in situation when a the surrounding blocks on the site all have linear facades with low porosity (see Figure 5.3).

The tendency to generate a specific typology can be considered as a limitation of this model if one wishes to create a universal model capabable of replicating street block style of an urban area. At the same time this property can be beneficial in case one desires to generate repeatable results with constant block style. With a collection of similar models, with each one reliably generating trained typology independent of the context, one could use these models almost like a library of solutions, ready to be plugged into any context.

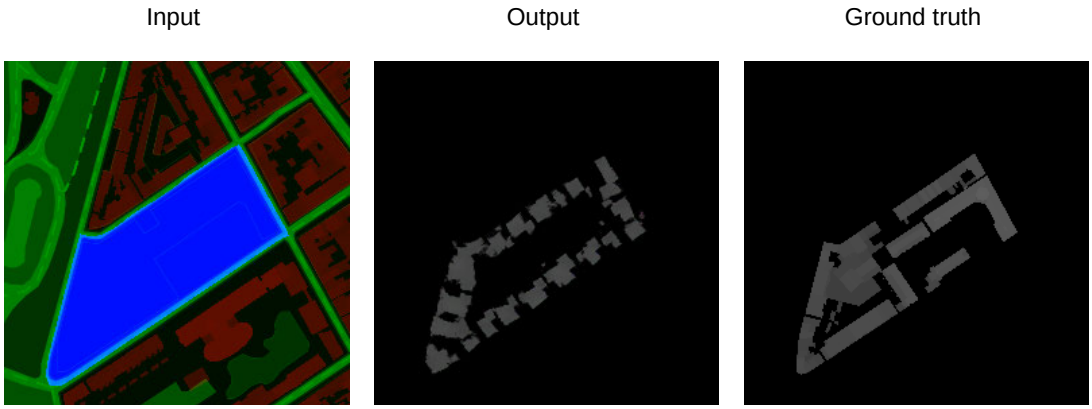


Figure 5.3: Height-map generated for a residential street block from Residential street block generator

Bloemkoolwijk street block generator The third experiment (page 58) was focused on comparing the results of a model trained on residential housing typology of Dutch *Bloemkoolwijken*, to a model trained on the generic uncurated residential neighborhood samples. The results show that the model is at least to a certain extent able to extract specific morphological traits out of existing urban configurations and apply them to new situations.

The model couldn't capture the most defining feature of *Bloemkoolwijken*, which is their sprawling tree-like street network, since the street patterns are already defined in the input data. But, instead, it learned another morphological trait one can observe when studying these neighborhoods, which is the very linear form of row houses built along the streets. The forms generated by the model trained on these neighborhoods are, at least by my visual evaluation, indeed distinctly more linear than for the generic model, with less setback and gaps between buildings.

When testing both generic and 'cauliflower' models on site never seen before in the training data, both models preserved their traits which were observed during training This tendency to generate linear forms becomes especially evident when comparing the results with the results generated for the same sites using the 'generic' residential block model (see Figure 5.5).

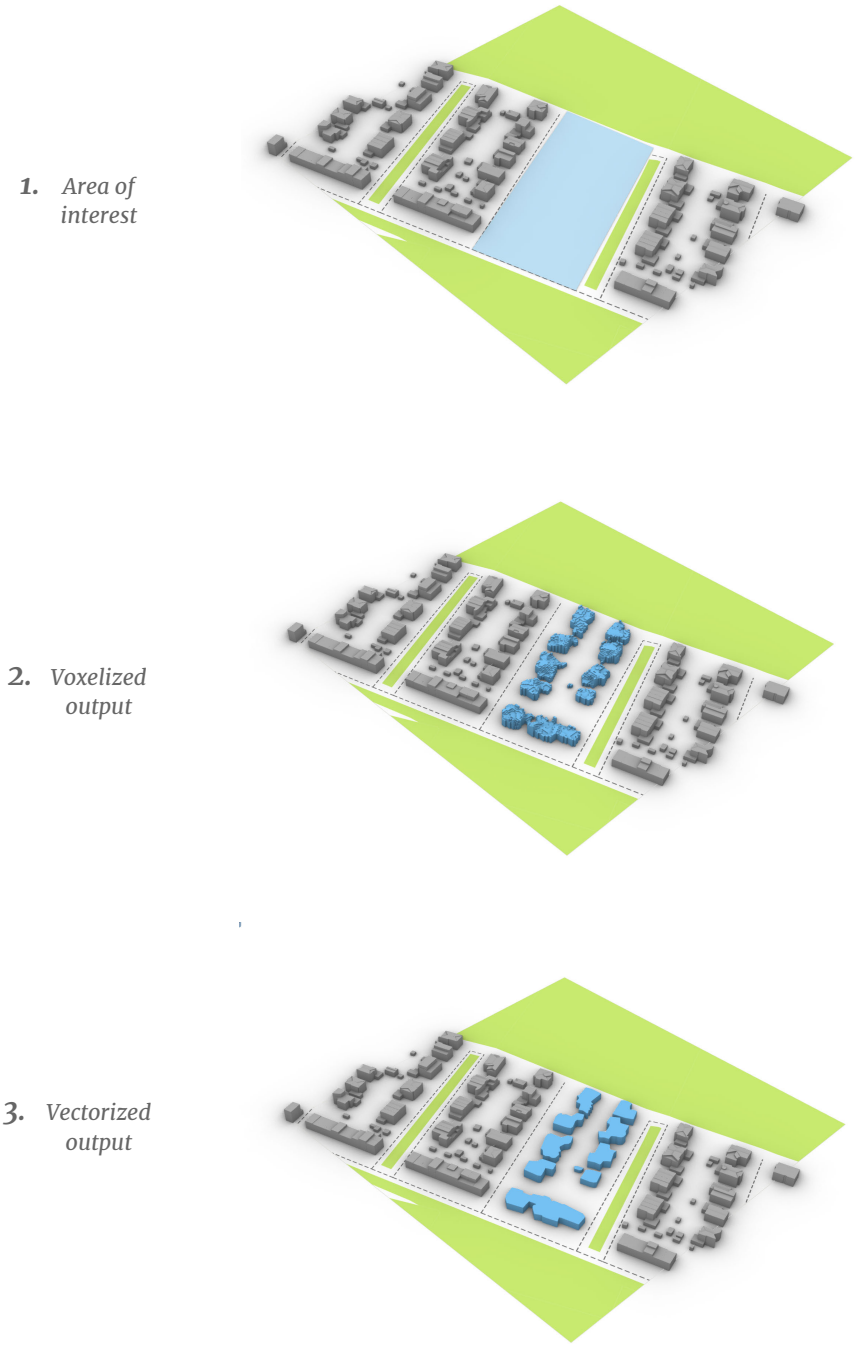
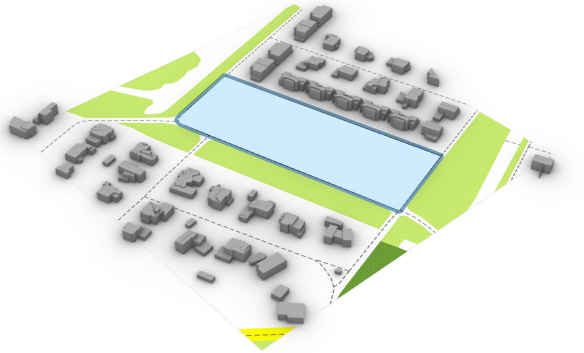
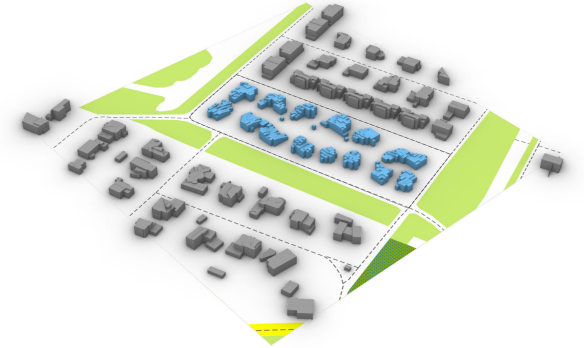


Figure 5.4: Rendering of the results from Residential street block generator

1. *Area of interest*



2. *Generic residential model*



3. *Bloemkool-wijk model*

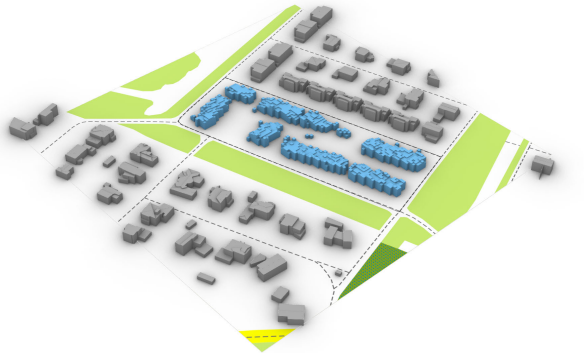


Figure 5.5: Rendering of the results from Bloemkoolwijk street block generator compared to the results of the model from Residential street block generator using an identical input

Urban fabric densification The fourth experiment (page 60) represents a method for training the model to autonomously identify gaps in existing urban fabrics and generate built volumes that could potentially fill them. If the model trains on synthetically "reduced" urban structures and learns to fill them in a way that resembles the original fabric, I hypothesized the model would be able to find real gaps in the urban areas, even in datasets that had no existing building intentionally removed.

From the results of the experiment, the model seems to indeed behave this way. Even in areas never seen before in the training data, with already a fairly dense structure, it manages to find reasonable ways to complete the shapes of the urban forms.

Certainly, some of these interventions would not be possible due to existing infrastructural or ownership limitations, land encumbrances or other boundary conditions not represented in the input data. But the model seems to respect the land-use limitations (blue input channel) and street boundaries (green input channel) and leave these areas empty (see Figure 5.6).

In my opinion such a model could be quite useful as a "smart" infill tool for explorative densification studies. The ability to quickly highlight gaps in current urban areas with potential for infill could serve as a "quick-scan" tool for municipalities and city planners, helping them assess the densification potential of various parts of a city fabric (see Figure 5.7).

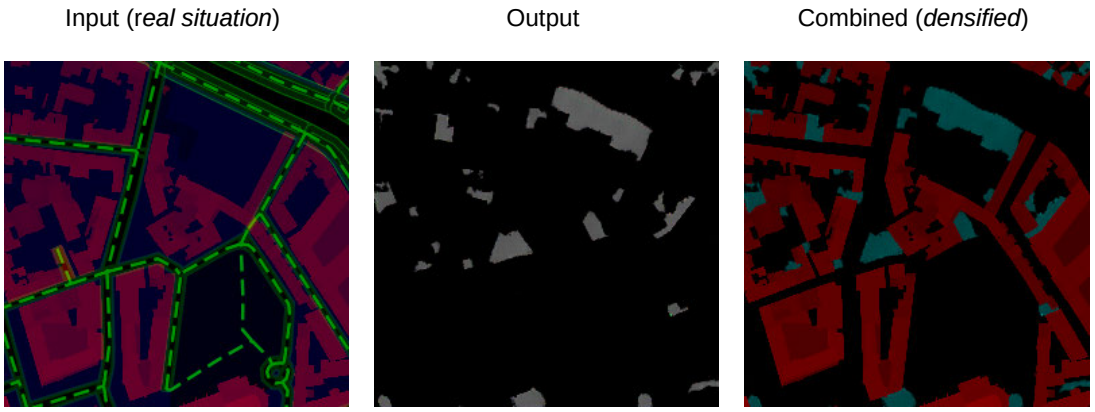


Figure 5.6: Height-map of additional massing added to urban area from Urban fabric densification

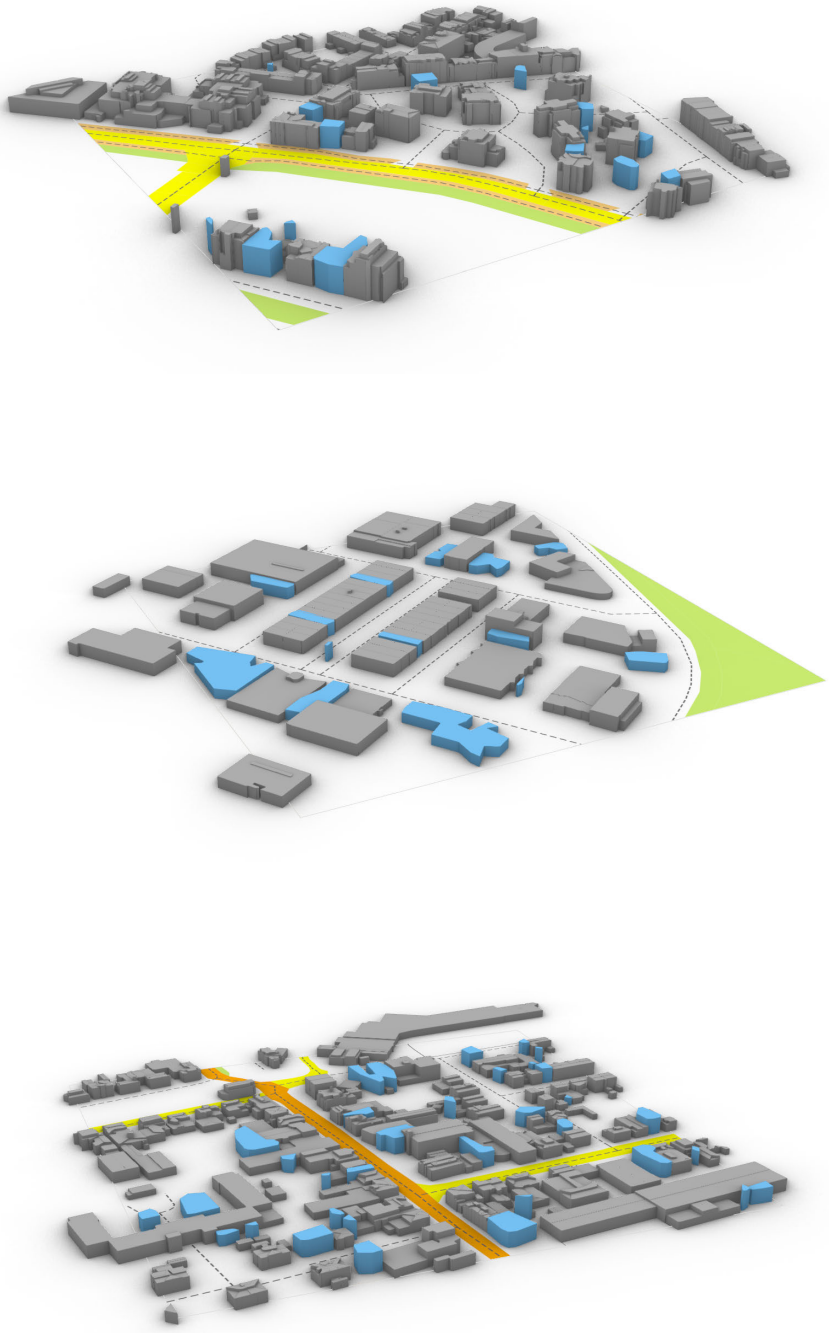


Figure 5.7: Rendering of the results from Urban fabric densification for different areas from validation dataset

Study on scale The fifth experiment (page 62) is centered around testing the influence of the image scale on the training result. I can conclude from the results that the scale is indeed an important parameter to consider when creating the training datasets and it can have a decisive influence on the clarity of the generated shapes.

The results of the model trained at 512x512m frames are clearly less detailed compared to the 256x256m results. The additional site context visible to the model at higher scales does not seem to outweigh the reduced amount of information dedicated to the target feature itself. The 1px/m seems as a good middle ground for a scale to use, although additional experiments with even more details could yield interesting results.

Noteworthy as well are the results of the model trained on large office blocks from 1990 and later, with generated forms distinctly different from the previous models (see Figure 5.8).

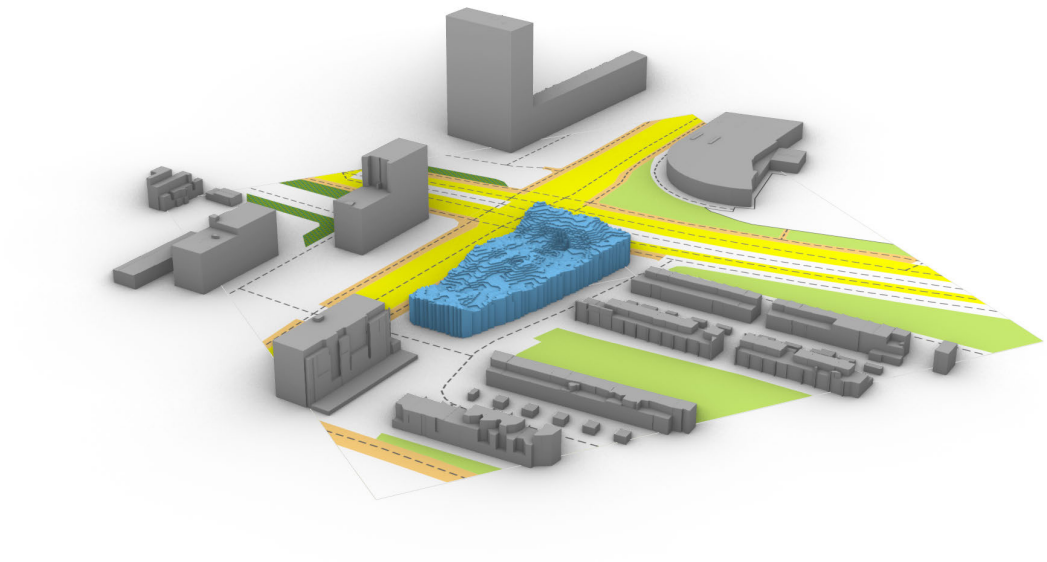


Figure 5.8: Rendering of one of the results from Study on scale

Study on level of detail The sixth and last experiment (page 64) aims to evaluate the importance of the level of detail of the building 3D geometries used in the training. The three models were trained on identical datasets, with the only difference being the LoD of the height-maps used to represent the building geometries.

When comparing the height-maps generated by each of the models, it seems that the LoD 1.2 and LoD 1.3 trained models tend to generate buildings with larger footprints compared to LoD 2.2. This could be caused by the footprint discrepancy between the footprint geometries provided for different LoDs in the 3D BAG dataset, where in some cases part of the BAG footprint of a building are omitted in the LoD 2.2 models (see Figure 5.10).

Additionally, one can observe much less noise in the height information generated by the lower LoD models (especially LoD 1.2) compared to higher LoDs. This results in cleaner roof geometry, requiring less drastic smoothing of the generate heightmaps to get reasonable results (see Figure 5.9).

My interpretation of this issue is that since LoD 1.2 models use only single height for the whole footprint, the GAN model learns to output a single greyscale value for each generated shape to match the ground truth data. It is my hypothesis that the Generator network trained on higher LoD models attempts to mimic the complexity of the roofscapes captured in e.g. LoD 2.2 and tries find a much more complex function allowing for proper roof geometry generation. This reduces the training bandwith of the Generator for output smoothness optimization and results in increased noise, especially at lower training data counts. I will focus on this issue more in depth in the next Section 5.2 Similarity statistics.

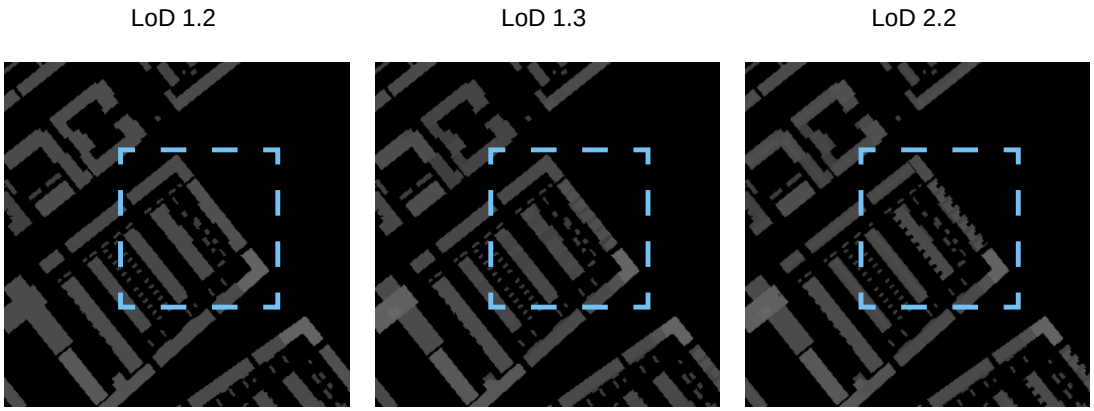
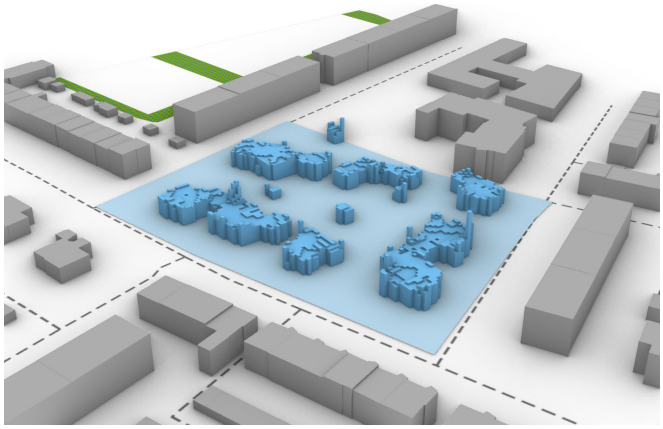
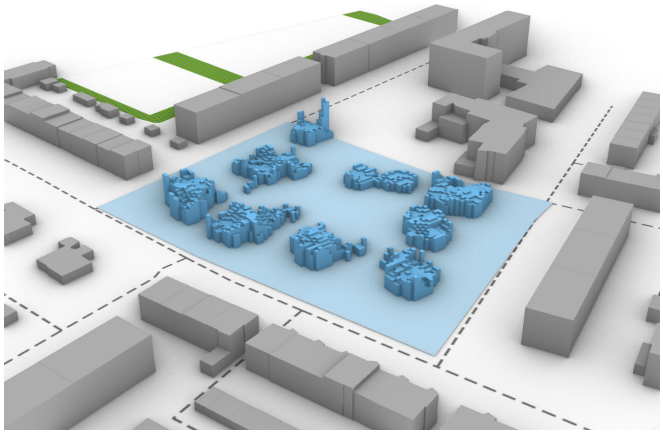


Figure 5.9: Height-map of ground truth input height-maps in Study on level of detail

LoD 1.2



LoD 1.3



LoD 2.2

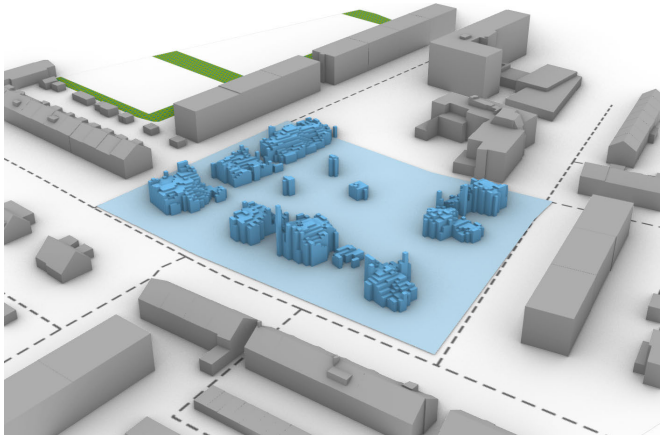
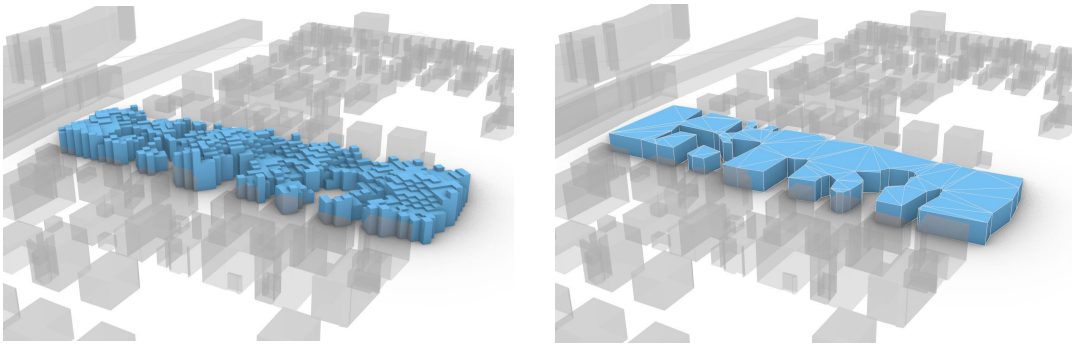


Figure 5.10: Rendering of the results from Study on level of detail for each of the LoDs the models were trained on

Conclusions From the visual evaluation of the generated solutions, both in their raw representation and in the three-dimensional representation extracted using voxelization or vectorization, I find the footprint shapes generated by the GAN model seem quite convincing and there seems to be a correlation between the typologies the model was learned on and the generated results. I have to note here that the model seems to struggle with defining the height values of the massing. The height-map values are often quite noisy, requiring at least one iteration of image smoothing to get reasonable results in the z-axis after voxelization. The LoD 1.1 vectorization of course takes care of this issue, but on the other hand it omits any height details included inside a single footprint (see Figure 5.11).

This is problematic, since for non-solitary standing buildings, the shape detection struggles to separate the overall shape into multiple parts. Therefore in densely built areas with little to no separation between buildings, the vectorization approach is quite destructive with respect to the information generated by the model. I will further elaborate on the behavior of the model when generating height information by means of numerical analysis in the next chapter.

Overall, the results show that the applied methodology framework is quite flexible in allowing one to generate various models tailor for specific building types and scopes. Each of the models trained in previous experiments generates its own distinct massing typologies. In the next chapter I will present my analysis on how similar these typologies actually are to the data the models were trained on.



Voxelization
**heightmap preprocessed by adaptive smoothing filter*

3Dfier height extrusion
**footprint extruded to 50th percentile of roof height*

Figure 5.11: Comparison of the two geometry extraction algorithms applied to the same output

5.2 Similarity statistics

The goal for the statistical analysis of the results is to quantify the similarity of the generated outputs to the training data. This is done to test the hypothesis that if the model trained on buildings with certain traits, these traits will be repeated in the generated outputs.

Metrics To evaluate the similarity of typological properties of the generated building massing, I quantify the volumetric properties of the generate massing as comparable quantitative metrics based on the *Spacematrix* method [Berghauser Pont and Haupt, 2009] — FSI, GSI, and OSR (see Section 3.4 Training data and Section 3.6 Evaluation for further explanation of these metrics).

I compute the similarity of these metrics for the out-of-sample data points (data points not seen by the models during the training) using root mean square error (RMSE) and normalized root mean square error (nRMSE). RMSE is a is a standard method to measure the error of a model in predicting quantitative data, defined as;

$$\text{RMSE} = \sqrt{\sum_{i=1}^{n} \frac{(x_i - y_i)^2}{n}} \tag{5.4}$$

Lower the RMSE, the more accurate is the model to the ground truth data. However, RMSE is a scale-dependent statistical measure — meaning its value depends on the general scale of the values in data it is applied to. While this means it can be still used to compare accuracy of different models applied to the same dataset and variable, it cannot be used to compare accuracy across different datasets. One can normalize the RMSE to make it scale-independent. Since there is no single established choice of normalization method, I chose to normalize using the ground truth mean;

$$\text{nRMSE} = \frac{\text{RMSE}}{mean(y)} \tag{5.5}$$

Together, RMSE and nRMSE, can be used to evaluate the ability of the model to match the typological properties of the generated massings to the ground truth. Both for multiple models applied to the same dataset (see Table 5.4) or for evaluating accuracy of models trained of different dataset (Table 5.1, Table 5.2, and Table 5.3).

	FSI	**GSI**	**OSR**
Ground truth (mean)	0.532	0.2461	1.65
Output (mean)	0.445	0.2591	1.873
RMSE	0.156	0.0489	1.102
nRMSE	0.293	0.198	0.667

Table 5.1: FSI, GSI, and OSR statistical similarity measures computed for all out-of-sample massings from Single residential building generator experiment

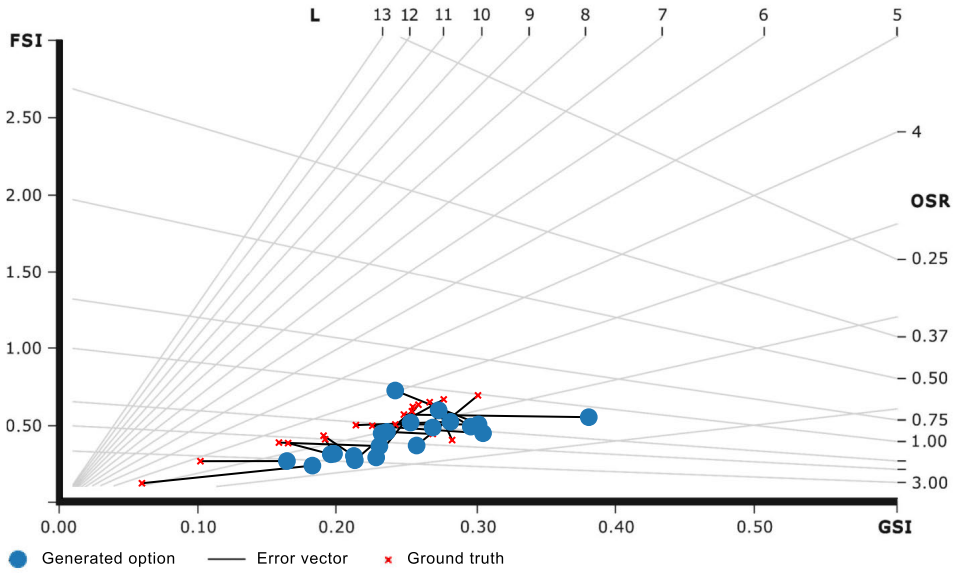


Figure 5.12: The out-of-sample outputs generated by the Single residential building generator mapped to Spacematrix (plot template redrawn from [Pont and Haupt, 2007])

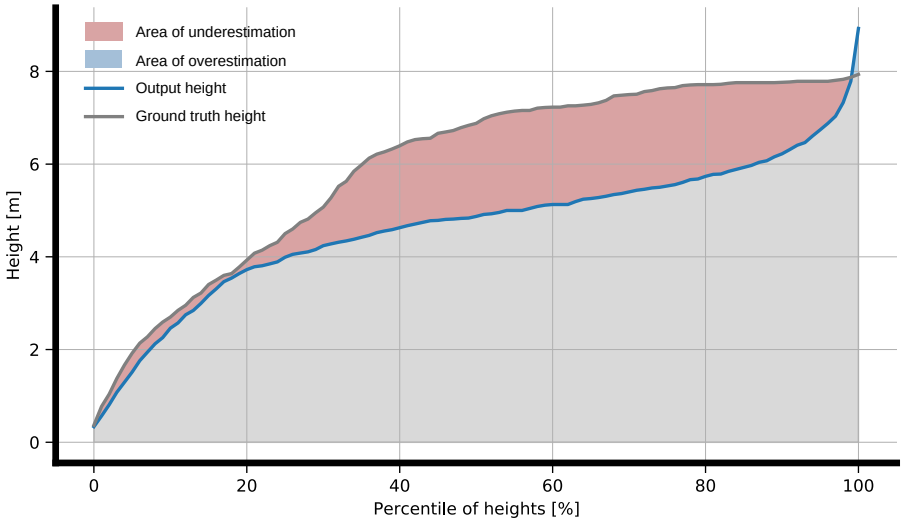


Figure 5.13: Comparison of cummulative height distributions for the out-of-sample outputs generated by the Single residential building generator compared to the ground truth

	FSI	GSI	OSR
Ground truth (mean)	0.77	0.2926	1.204
Output (mean)	0.53	0.2424	1.721
RMSE	0.456	0.1002	0.827
nRMSE	0.592	0.342	0.686

Table 5.2: FSI, GSI, and OSR statistical similarity measures computed for all out-of-sample massings from Residential street block generator experiment

	FSI	GSI	OSR
Ground truth (mean)	1.407	0.3383	0.947
Output (mean)	1.181	0.3246	1.081
RMSE	0.799	0.1201	0.666
nRMSE	0.567	0.355	0.703

Table 5.3: FSI, GSI, and OSR statistical similarity measures computed for all out-of-sample massings from Study on scale experiment

Plots To help visualize the distributions of the typological traits for both generated and ground truth building massings, I plot each variant from the validation datasets as a data point on a *Spacematrix* chart. According to Pont and Haupt [2007], specific areas of *Spacematrix* chart can be correlated to specific urban typologies (see Figure 3.20). This means that in the generated plot, one can observe that further the mapping of a generated volume is to the ground truth point, the larger the difference between the typology the model generated at a given location compared to the existing building (see Figure 5.12, Figure 5.15, and Figure 5.17).

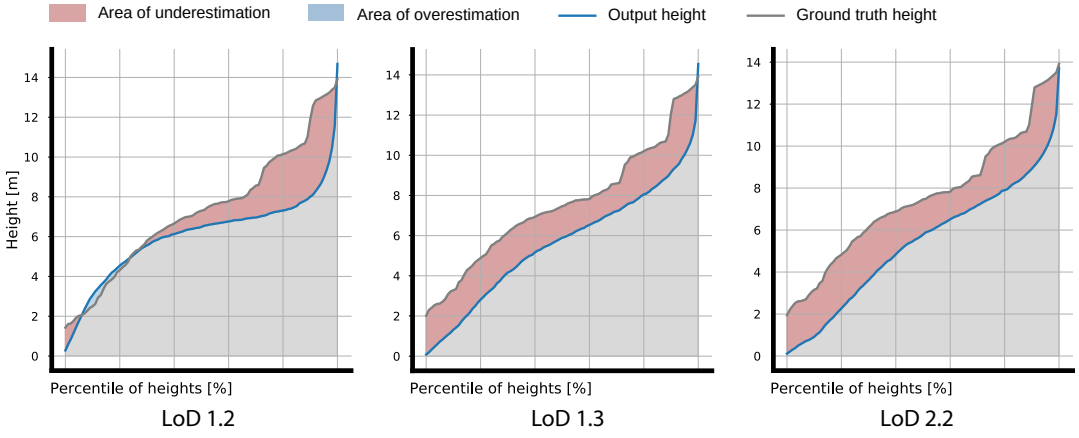


Figure 5.14: Comparison of cummulative height distributions for the out-of-sample outputs generated by the models trained with different LoDs from Single residential building generator compared to the ground truth (LoD 2.2)

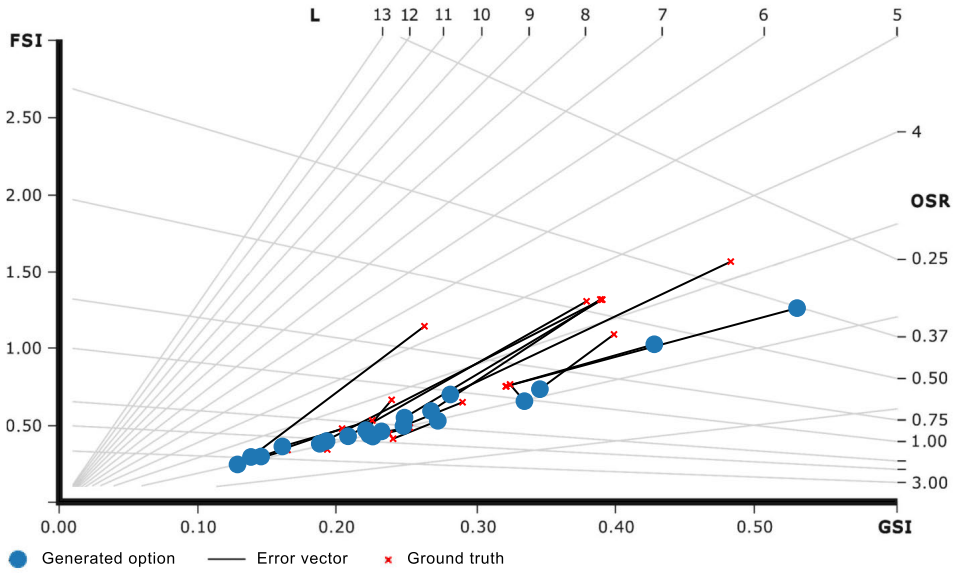


Figure 5.15: The out-of-sample outputs generated by the Residential street block generator mapped to Spacematrix (plot template redrawn from [Pont and Haupt, 2007])

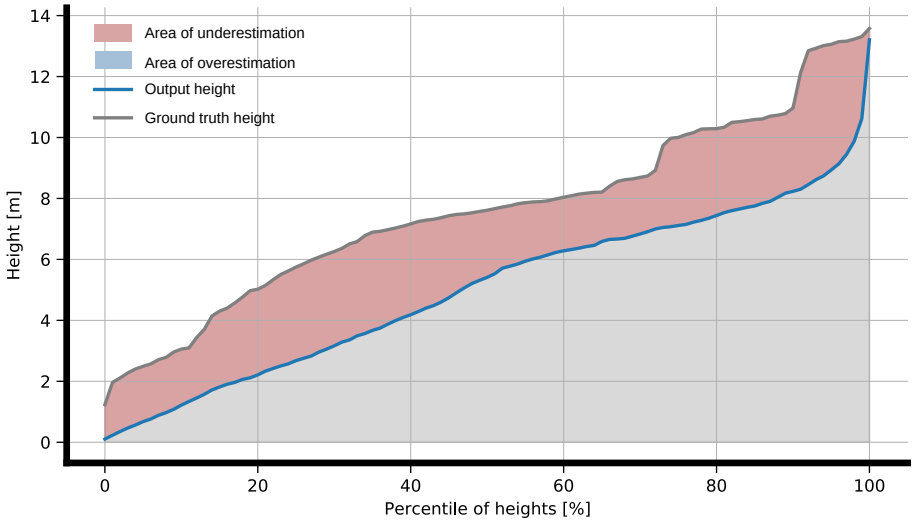


Figure 5.16: Comparison of cummulative height distributions for the out-of-sample outputs generated by the Residential street block generator compared to the ground truth

Additionally, I plot the cumulative distributions of heights generated by the model and compare to the ground truth. The heights are measured per pixel of the massing height-map, which means that the area under the cumulative distribution curve is proportional to the total volume bound by the building massing. Figure 5.13, Figure 5.16, and Figure 5.18 show the mean cumulative distribution functions for all out-of-sample cases in their respective experiments. Figure 5.14 compares the height distributions for models trained using different LoDs of massing height-maps.

Conclusions Even though the error vector between the generated and ground truth typology in the *Spacematrix* mappings of typological traits is often substantial, the general distributions for both the generated and ground truth data are quite similar. The models do not generate in every case exactly the same typology as the one already built at a given plot, but the distribution of their generated outputs matches the general distribution of typologies present in the their training data. This can be interpreted as the model being capable of learning the general typological traits present in the data and being able to generalize these learnings to new situations in a way thats does not necessarily match the ground truth.

Out of the three models compared (Single residential building generator, Residential street block generator, and 256x256m model from Study on scale), the Single residential building generator shows the highest accuracy in terms of FSI, GSI, and OSR. This can most likely be attributed to the fact, that the model was trained with the largest dataset — 27,674 compared to 1,000 for later models.

However, for all models one can observe a tendency to underestimate the total building volume. This is noticeable in both the FSI and GSI means being lower for generated massings than ground truth massings and in the height distributions. This corresponds with the model behavior observed for Residential street block generator in Section 5.1 Visual analysis, where the model tended to generate free standing building even in areas with much denser street block structure, but for other models the reasons for these results are less clear.

	FSI	GSI	OSR
Ground truth (mean)	0.716	0.2734	1.291
LoD 1.2 (mean)	0.593	0.2597	1.546
LoD 1.3 (mean)	0.495	0.2296	1.743
LoD 2.2 (mean)	0.516	0.2321	1.797
LoD 1.2 (RMSE)	0.336	0.0698	0.569
LoD 1.3 (RMSE)	0.332	0.0621	0.651
LoD 2.2 (RMSE)	0.364	0.0711	0.788

Table 5.4: FSI, GSI, and OSR statistical similarity measures computed for all out-of-sample massings from Residential street block generator experiment

Surprisingly, the last experiment shows that the model trained on LoD 1.2 massing data is most accurate to the ground truth (modeled as LoD 2.2 massings). It achieves the lowest RMSE and closest mean values for all *Spacematrix* metrics (FSI, GSI, and OSR) (see Table 5.4) and has the best fitting height distribution curve (see Figure 5.14). To explain this behavior, further research is needed.

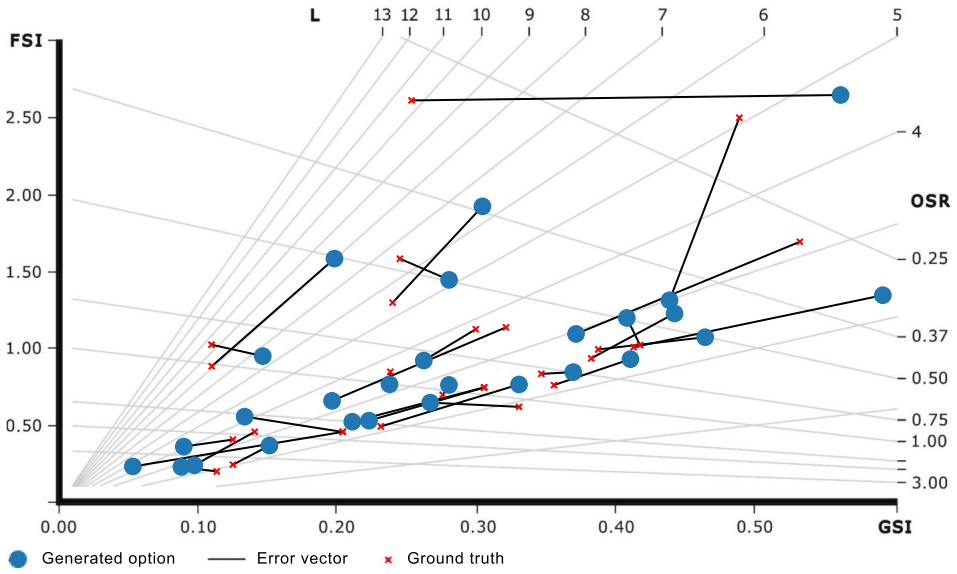


Figure 5.17: The out-of-sample outputs generated by the 256x256m model from Study on scale mapped to Spacematrix (plot template redrawn from [Pont and Haupt, 2007])

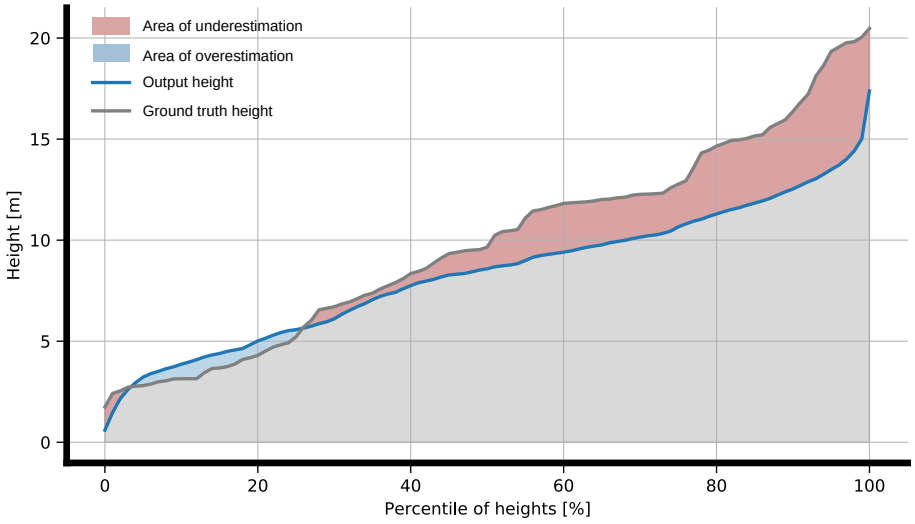


Figure 5.18: Comparison of cummulative height distributions for the out-of-sample outputs generated by the 256x256m model from Residential street block generator compared to the ground truth

6 Conclusions

In this final chapter, I will recapitulate the goals and scope of this work and list the chosen methods of approaching the research problem and their implementations. I will answer the research questions posed at the start of this report and state the limitations of the chosen methods. Finally, I will conclude the chapter with a discussion on the reasons behind some of the choices taken during the course of the development of this thesis and conclude with an overview of potential future research topics.

6.1 Research overview

My goal for this research was to expand upon existing methods for generation of architectural and urban massings by utilizing deep-learning models. I found that even though multiple works on this topic already exist, they make use of only simplified two dimensional representation of the building geometries and omit most of the site context the buildings designed are to be located in.

To elevate these issues, I proposed a methodology for generating building massings using Pix2Pix GAN model trained on highly accurate 3D building geometries of the buildings in the Netherlands obtained in the form of 3D BAG dataset. Additionally, I enriched the data model using spatial, qualitative, and quantitative information obtained from open datasets provided by the Dutch geospatial data services [PDOK, 2022].

The results generated by the model are evaluated for their similarity to the data the model was trained on by evaluating the models ability to match typological properties of the ground truth data. *Spacematrix* framework and cumulative height distribution analysis is used for this purpose.

Research sub-question 1

> *At what scales is such model applicable (eg. single building, building group, street block)? Is a single general model sufficient, or does one need to train a separate dedicated model for each scale?*

Three different applications of the Pix2Pix GAN at different scopes and scales were presented — single building generator (see Section 4.4.1), street block generator (see Section 4.4.2 and Section 4.4.3) and urban fabric infill generator (see see Section 4.4.4). Each model targets a specific use case, and while they allow for certain flexibility regarding the types of environments and sizes of plots they can be applied to, they cannot be used interchangeably. Especially since each of them have different sets of goals they were trained to fulfill — placing a single building on a parcel, filling an open plot with a building block, or finding locations for possible building infill in existing urban areas.

Research sub-question 2

> *Is it possible to steer the properties of the generated designs towards certain traits by curating the data the model learns from? What metrics should one use to measure these properties? How strong is the correlation between measurable properties of the designs in the training data and the generated results?*

Pix2Pix GAN models trained during the course of the research show an ability to replicate the urban typological traits observed in the training datasets. Therefore it is possible to control the type of massings the models produce by curating the data samples the models are trained on. This is confirmed by mapping the distributions of the traits in ground truth and generated massings using the *Spacematrix* method. While the models do not always generate the identical typology for a given plot, as in the ground truth data, the general outputs of the model match the typologies present in the training samples. Models trained with larger data sets show higher correlation of the measured metrics (FSI, GSI, and OSR) to the ground truth.

Research sub-question 3

> *How important is the LoD of the training data and what aspects of site context have to be included in the training data for the quality of the results?*

Counterintuitively, the experiments have shown that, at least for smaller training data sets (1,000 samples), the model shows higher ability to match the typological properties of ground truth building volumes (modeled as LoD 2.2 building geometry) when trained of LoD 1.2 data, compared to models trained using higher LoD massing models (LoD 1.3 and LoD 2.2). However, further research is required to understand the reasons for this behavior and to test whether it is particular to specific training dataset sizes.

Including site context information is critical to ensure the results of the models can be used in practice, since only sufficient site information can prevent the model from generating building volumes in areas that are unbuildable due to infrastructural, legal or environmental limitations. When these limits are included in the input data, such as street space extents, the models seem to follow these limitations strictly, avoiding potential design conflicts.

Main research question

> *To what extent can one assist the process of building massing* (and by extension the built environment) *design generation by utilizing GAN model trained on existing building forms represented in Dutch buildin stock?*

The resulting models show an ability to infer rules of building massing design from the situations observed during the training. By curating the datasets that the models are trained on, it is possible to create models targeting different specific situations encountered in building design practice. The experiments conducted show that the developed methodology can be used to generate GAN models with various design goals — whether that goal is generating design of a single building, a building group, or open ended exploration of possible densification targets in urban areas.

Figure 6.1: Rendering of a block massing generated by the 256x256m model trained as part of Study on scale

Reflection on scientific contribution

Generative design Although a similar application of Pix2Pix GAN to generation of building forms has been explored in multiple previous works (covered as part of the literature review in Section 2.3), this thesis presents a novel research thanks to training the model on high fidelity 3D building models combined with multi-layered site context information. The developed methodology is applied to multiple use cases, applying the model at different scales and scopes, presenting a unified framework for using GAN models in architectural and urban design generation.

3D city models The use of large scale 3D city models for the purpose of informing a generative design framework is a relatively novel use of this type of data, which shows the potential of such datasets not only for city governance, management, analysis and visualization, but for design and exploration of future urban interventions as well.

Quantitative evaluation The evaluation of the results using the *Spacematrix* framework — borrowing methods usually applied to analysis of urban fabrics to evaluate the urban typological traits of generated building massings — connects this work to the field of quantitative urbanism. Using such metrics provides a better picture of the accuracy and usability of the results in the field of urban and architectural design, compared to relying on more typical accuracy measures used in the field of deep learning, such as per pixel error distance.

Implementation From technical perspective, the the most noteworthy contribution are the IronPython scripts allowing for interfacing with geospatial APIs and formats, such as WFS, WMS, PostGIS, or GeoJSON, while using the *Rhinoceros 3D / Grasshopper 3D* CAD platform which does not natively support working with geospatial data. All code is openly available on GitHub[1] (see Section 4.1).

6.2 Limitations

Building typology I found it quite difficult to selectively define and filter distinct types of building configurations using openly available datasets. Queries on existing properties defined in the open datasets provided by the Dutch spatial data infrastructure [PDOK, 2022] are insufficient for this purpose, since the offered data on building national stock do not include useful semantic information beyond building use and construction age. The building typology has to be either inferred from the geometry by proprietary analysis methods, or manually labeled.

[1] https://github.com/ondrej-vesely/massingGAN

Urban planning limitations The model does not take into concern many administrative, technical or legal limitations that would otherwise restrict the possible massing design in the area (e.g. land easements, infrastructural corridors, or building massing limitations put in place in the masterplan). Although for some of them it can probably infer these limitations from the surrounding site context - e.g. the maximum building heights, without their inclusion, the model can be prone to generating infeasible solutions.

Such data could be added as an additional input layer. The model seems to follow the restrictive layers already included in input images quite well (e.g. road extents, parcel extents, block extents). Alternatively, subtracting the parts of massing conflicting with areas restricted to the building design could be done as a post-processing step.

Clarity of the features The models trained during the course of the research, at least to some degree, lack the capability to generate straight regular features. The U-Net based Generator seems to struggle to generate repetitive and orthogonal patterns and define straight continuous edges. At least in some cases, the ResNet based Generator tested during the development has shown improved results in this regard compared to the primarily used U-Net Generator. However, some of these difficulties with generating clearly defined shapes are inherent to the nature of Pix2Pix GANs, which *"... have difficulty in handling discrete output as Generator always predicts continuous output"* [Isola, 2017].

Discriminator receptive fields The CNN architecture for the Discriminator used in the pix2pixGAN model is PatchGAN. PatchGAN architecture divides the image generated by the Generator CNN into multiple 70x70 pixel kernels (called receptive fields) and then classifies contents of each receptive field as real or fake. Finally it averages the classification probability for all patches to calculate the probability of the full image being real.

This means that the Discriminator can rate the images as fake or real only based on local details contained within these 70x70 pixel large fields. Relationships between elements located further across from each other within the image frame than the 70 pixels cannot be fully evaluated and do not influence the classification score output by the Discriminator.

The Generator still consumes the full image representation and generates the results based on all information present within the image, but is not able to evaluate some of the larger scale relations between the elements depicted. Since the training of the Generator is ultimately steered by the GAN loss dependent on the Discriminator, this myopic trait of the Discriminator network can be a limitation to the quality of the results.

Spacematrix method Using *Spacematrix* as a framework for evaluating the typological traits of the generated massings has an advantage of mapping the results to metrics established for quantitative analysis of building forms in the field of urban design. However, it has some limitations;

Firstly, the framework was originally intended for uses on scales larger than the ones found in this research. On the scale of a single street block or especially a single parcel, the definition of what constitutes the measured 'plot' influences the resulting metrics quite drastically (e.g. do measured extents of a single block end at street center line or at street setback limits)?

Further, for a given plot, the three metrics (FSI, GSI, and OSR) are essentially all codependent on only two variables — the total footprint *and* the total volume of the building massings. They do not take further details such as the exact footprint shape or the roof geometry into consideration.

6.3 Discussion

Are the massing designs generated by the model the best possible design solutions?

The goal of this work is not for the generated massings to 'optimal' or 'ideal' designs, since defining what constitutes an optimal or ideal design lies well beyond the scope of this work. Instead the goal is to generate massing designs which are feasible and plausible. The difficulty still lies in defining what it means for generated design to be plausible. In the context of this work, I measure the similarity of the design to existing design solutions, assuming that designs that are proven to be informed by previous solutions are a valid configuration.

While it can certainly be argued that informing designs only by previous solutions could lead to repetitiveness and lack of innovation, it is also true that making a new building form fit into an existing context is one of the important objectives of architectural design, so a certain degree of familiarity and similarity to existing building forms is certainly desirable. Ultimately, all designs are in some form a synthesis of concepts present in the precedent designs. It is still up to building industry professionals to curate which ones are the optimal synthesis of elements for a specific site and context.

Why is the pix2pix GAN in the given context a good choice? How could we overcome the limitations of the fixed grid (border size and resolution) that's necessary for the algorithm?

The necessity to convert the geometrical building features into discrete data in the form of voxelized geometry (respectively voxelized geometry encoded into a form of heightmap raster) is indeed a limitation of this method. Ideally the conversion from vector representation to voxel and back would be completely omitted. However, the necessity to convert the training data into discrete arrays of fixed size is inherent to the nature of CNNs, which always operate with the data in the form of tensors (multidimensional arrays).

While some early efforts of converting simple vector geometries into tensors using graph representations exist, the standard way for dealing with three dimensional geometries when training GANs is voxelization, which ensures uniform representation of the captured shapes regardless of the geometrical complexity.

Why not use some even more detailed topographic datasets to inform the site context, e.g. the BGT (Large-scale Topographic Map of the Netherlands)?

While BGT indeed includes more data, such as detailed shapes of any sidewalks and paved areas, I believe the more simplified topographic map such as TOP10NL strikes the perfect balance between detail and ambiguity, when it comes to informing the site context of the model creating building massings.

This is my assumption, but more detailed topographic information, such as the one in BGT, is often too closely correlated to the existing building layouts and could 'give away' the

ground truth massing situation to the model, essentially teaching it to just follow the contours of e.g. the existing paved areas. However, addition of some site information from these datasets, such as the existing vegetation objects, could be an interesting experiment for future work.

> *Is the described method only applicable in data rich environments such as the Netherlands? Can one train a similar model with the LoD of the building models available in other countries? What global alternatives are there to 3D BAG?*

While the 3D BAG is certainly not the only openly available LoD 2+ building geometry dataset and many municipalities around the world offer LoD 2+ 3D models of their city extents, experiments conducted during this research show that the proposed model is capable of generating results of with similar, or even better accuracy when using only LoD 1.2 building geometry models. While I am not aware of any other dataset than 3D BAG with LoD 2+ accuracy and detail at the same scale of full national coverage or larger, it could be argued there are some global datasets with LoD 1.2 coverage, such as OSM. While the quality and reliability of the information included in these global datasets differs from city to city, since many of the experiments presented in this work use only circa 1,000 data samples, I deem it feasible to replicate the results with data sourced from only one (larger) municipality.

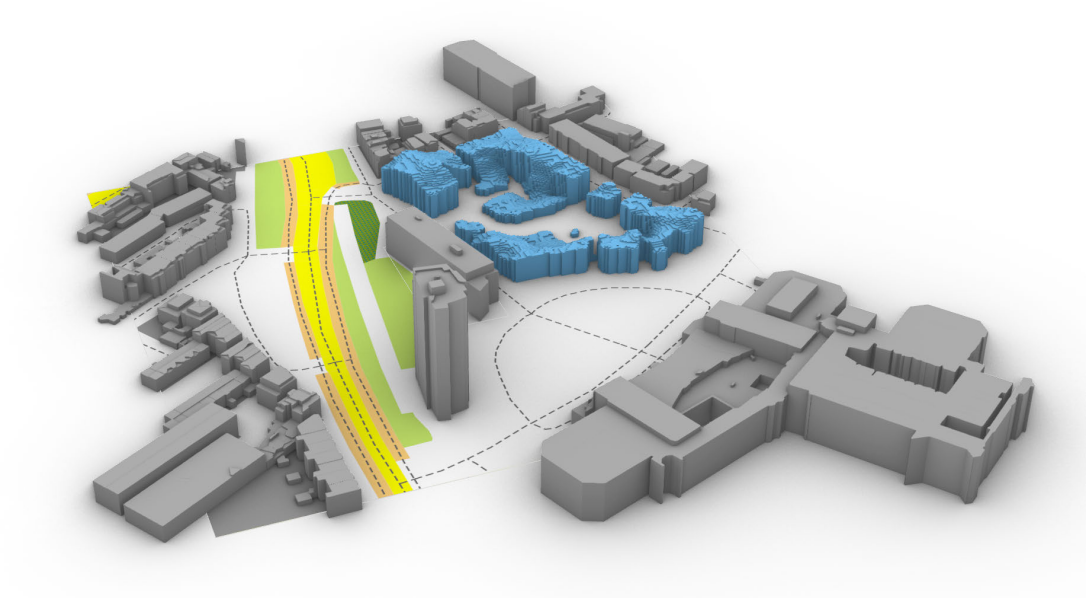


Figure 6.2: Rendering of another massing generated by the 256x256m model trained as part of Study on scale

6.4 Future work

Evaluation metrics It would be beneficial to the rigorousness of evaluation of the results to include additional metrics for assessment of similarity of generated massings to the training data. While quantifying visual similarity of outputs of GAN models to ground truth data is still an open problem and is best done in real user studies, some methods for measuring visual similarity between features exist.

In the field of GAN research, the currently most established method is *Fréchet Inception Distance* (*FID*), developed by Heusel et al. [2017]. *FID* could be used to better evaluate the similarity of the forms generated by the models — beyond the simplified analysisi with quantitative metrics of the *Spacematrix* framework.

Additionally, some secondary metrics evaluating extrinsic behavior of the generated massings, such as their influence on urban microclimate (e.g. solar radiation, wind speeds), could be used to evaluate the similarity of the generated forms in the context of their influence on the built environment.

Software integration Since one of the goals of this thesis is to assess how feasible it is to use similar GAN based models in architectural and urban design practice, it would be interesting to demonstrate how such tools can be technically integrated into existing software commonly used by design professionals.

Since large parts of the methodology have already been implemented in *Rhinoceros 3D* / *Grasshopper 3D* environment, a software package popular by built environment design professionals, it would be sensible to explore how the rest of the products of this research could be integrated into this software environment — in particular, investigate how to containerize and deploy the pre-trained models on a target machine and create an API capable of requesting and retrieving inferred design solutions from the model.

Further experiments The experiments conducted during the course of the research still left some questions unanswered. Mainly, the question of how the lower LoD of training data with only simplified geometry can lead to models with higher similarity of generated geometries to ground truth, compared to models trained on higher LoD data. The next step necessary to investigate this phenomena would be replicating the experiment with larger and more diverse training datasets and testing whether this behavior persists.

Additionally, the influence of dataset size could be further explored, since most of the experiments were conducted with around 1,000 training data samples, which is considered a modest dataset size in the field of GAN research. Finally, it would be interesting to define more of targeted typological studies, such as the one presented in Section 4.4.3, to see what other typological features of building configurations can the model capture and replicate.

Bibliography

3D Geoinformation research group (2021). 3D BAG. https://docs.3Dbag.nl/en/. [Online] Accessed 3 May 2022.

Alonso, N. (2017). Suggestive drawing along human and artificial intelligences. https://www.gsd.harvard.edu/project/suggestive-drawing-along-human-and-artificial-intelligences/. [Online] Accessed 3 May 2022.

Berghauser Pont, M. Y. and Haupt, P. A. (2009). *Space, density and urban form*. TU Delft.

Biljecki, F., Ledoux, H., and Stoter, J. (2014). Height references of CityGML LOD1 buildings and their influence on applications. In *Proceedings. 9th ISPRS 3DGeoInfo Conference 2014, 11-13 November 2014, Dubai, UAE,(authors version)*. Citeseer.

Biljecki, F., Ledoux, H., Stoter, J., and Vosselman, G. (2016). The variants of an LOD of a 3D building model and their influence on spatial analyses. *ISPRS Journal of Photogrammetry and Remote Sensing*, 116:42–54.

Caetano, I., Santos, L., and Leitão, A. (2020). Computational design in architecture: Defining parametric, generative, and algorithmic design. *Frontiers of Architectural Research*, 9(2):287–300.

Carta, S. (2021). Self-Organizing Floor Plans. *Harvard Data Science Review HDSR*.

Chaillou, S. (2020). Archigan: Artificial intelligence x architecture. In *Architectural intelligence*, pages 117–127. Springer.

Chang, K.-H., Cheng, C.-Y., Luo, J., Murata, S., Nourbakhsh, M., and Tsuji, Y. (2021). Building-GAN: Graph-conditioned architectural volumetric design generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11956–11965.

Chen, M., Yu, R., Xu, S., Luo, Y., and Yu, Z. (2019). An improved algorithm for solving scheduling problems by combining generative adversarial network with evolutionary algorithms. In *Proceedings of the 3rd International Conference on Computer Science and Application Engineering*, pages 1–7.

Chronis, A., Aichinger, A., Duering, S., Galanos, T., Fink, T., Vesely, O., and Koenig, R. (2020). INFRARED: An intelligent framework for resilient design. In *25th International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA 2020)*.

Dong, S., Wang, W., Li, W., and Zou, K. (2021). Vectorization of Floor Plans Based on EdgeGAN. *Information*, 12(5):206.

Fedorova, S. (2021). GANs for Urban Design. *arXiv preprint arXiv:2105.01727*.

García, R., de Castro, J. P., Verdú, E., Verdú, M. J., and Regueras, L. M. (2012). Web map tile services for spatial data infrastructures: Management and optimization. *Cartography-A Tool for Spatial Analysis*, pages 26–48.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hesse, C. (2017). Image-to-image translation in tensorflow. *Online: https://affinelayer.com/pix2pix/, 08/14*.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30.

Isola, P. (2017). Phillipi/pix2pix: Image-to-image translation with conditional adversarial nets. https://github.com/phillipi/pix2pix. [Online] Accessed 3 May 2022.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134.

Khean, N., Kim, L., Martinez, J., Doherty, B., Fabbri, A., Gardner, N., and Haeusler, M. H. (2018). The introspection of deep neural networks-towards illuminating the black box-training architects machine learning via Grasshopper definitions. *CUMINCAD*.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

Ledoux, H., Arroyo Ohori, K., Kumar, K., Dukai, B., Labetski, A., and Vitalis, S. (2019). CityJSON: A compact and easy-to-use encoding of the CityGML data model. *Open Geospatial Data, Software and Standards*, 4(1):1–12.

Ledoux, H., Biljecki, F., Dukai, B., Kumar, K., Peters, R., Stoter, J., and Commandeur, T. (2021). 3dfier: automatic reconstruction of 3D city models. *Journal of Open Source Software*, 6(57):2866.

Mans, D. (2021a). interopxyz/bitmapplus: A bitmap manipulation plugin for Grasshopper 3D. https://github.com/interopxyz/BitmapPlus/. [Online] Accessed 3 May 2022.

Mans, D. (2021b). interopxyz/GraphicPlus: A scalable vector graphics plugin for Grasshopper 3d. https://github.com/interopxyz/GraphicPlus/. [Online] Accessed 3 May 2022.

Matalas, L., Benjamin, R., and Kitney, R. (1997). An edge detection technique using the facet model and parameterized relaxation labeling. *IEEE transactions on pattern analysis and machine intelligence*, 19(4):328–341.

Mokhtar, S., Sojka, A., and Davila, C. C. (2020). Conditional generative adversarial networks for pedestrian wind flow approximation. In *Proceedings of the 11th Annual Symposium on Simulation for Architecture and Urban Design*, pages 1–8.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

PDOK (2022). PDOK - Datasets. https://www.pdok.nl/datasets. [Online] Accessed 3 May 2022.

Peters, R. (2021). What is 3D BAG? https://3d.bk.tudelft.nl/courses/backup/geo1004/2021/data/slides2.2_3DBAG.pdf. [Online] Accessed 3 May 2022.

Pont, M. B. and Haupt, P. (2007). The relation between urban form and density. *Urban Morphology*, 11(1):62.

Ramer, U. (1972). An iterative procedure for the polygonal approximation of plane curves. *Computer graphics and image processing*, 1(3):244–256.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

Rutten, D. (2013). Galapagos: On the logic and limitations of generic solvers. *Architectural Design*, 83(2):132–135.

Selinger, P. (2003). Potrace: a polygon-based tracing algorithm. http://potrace.sourceforge.net/potrace.pdf. [Online] Accessed 3 May 2022.

Swahn, E. (2019). Latent spaces. https://www.studio9.arch.kth.se/author/erikswahn/. [Online] Accessed 3 May 2022.

Tian, R. (2020). Suggestive site planning with conditional GAN and urban GIS data. In *The International Conference on Computational Design and Robotic Fabrication*, pages 103–113. Springer.

Wu, A. N. and Biljecki, F. (2022). GANmapper: geographical data translation. *International Journal of Geographical Information Science*, pages 1–29.

Yao, J., Huang, C., Peng, X., and Yuan, P. F. (2021). Generative design method of building group-Based on generative adversarial network and genetic algorithm. *CUMINCAD*.

Ye, Y. and Van Nes, A. (2014). Quantitative tools in urban morphology: Combining space syntax, spacematrix and mixed-use index in a GIS framework. *Urban morphology*, 18(2):97–118.

Zhu, J. (2017a). junyanz/BicycleGAN: Toward multimodal image-to-image translation. https://github.com/junyanz/BicycleGAN. [Online] Accessed 3 May 2022.

Zhu, J. (2017b). junyanz/pytorch-CycleGAN-and-pix2pix: Image-to-Image Translation in PyTorch. https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix. [Online] Accessed 3 May 2022.

Zhu, J.-Y., Zhang, R., Pathak, D., Darrell, T., Efros, A. A., Wang, O., and Shechtman, E. (2017). Toward multimodal image-to-image translation. *Advances in neural information processing systems*, 30.

Colophon

This document was typeset using LaTeX, using the KOMA-Script class `scrbook`. The main font is Palatino.