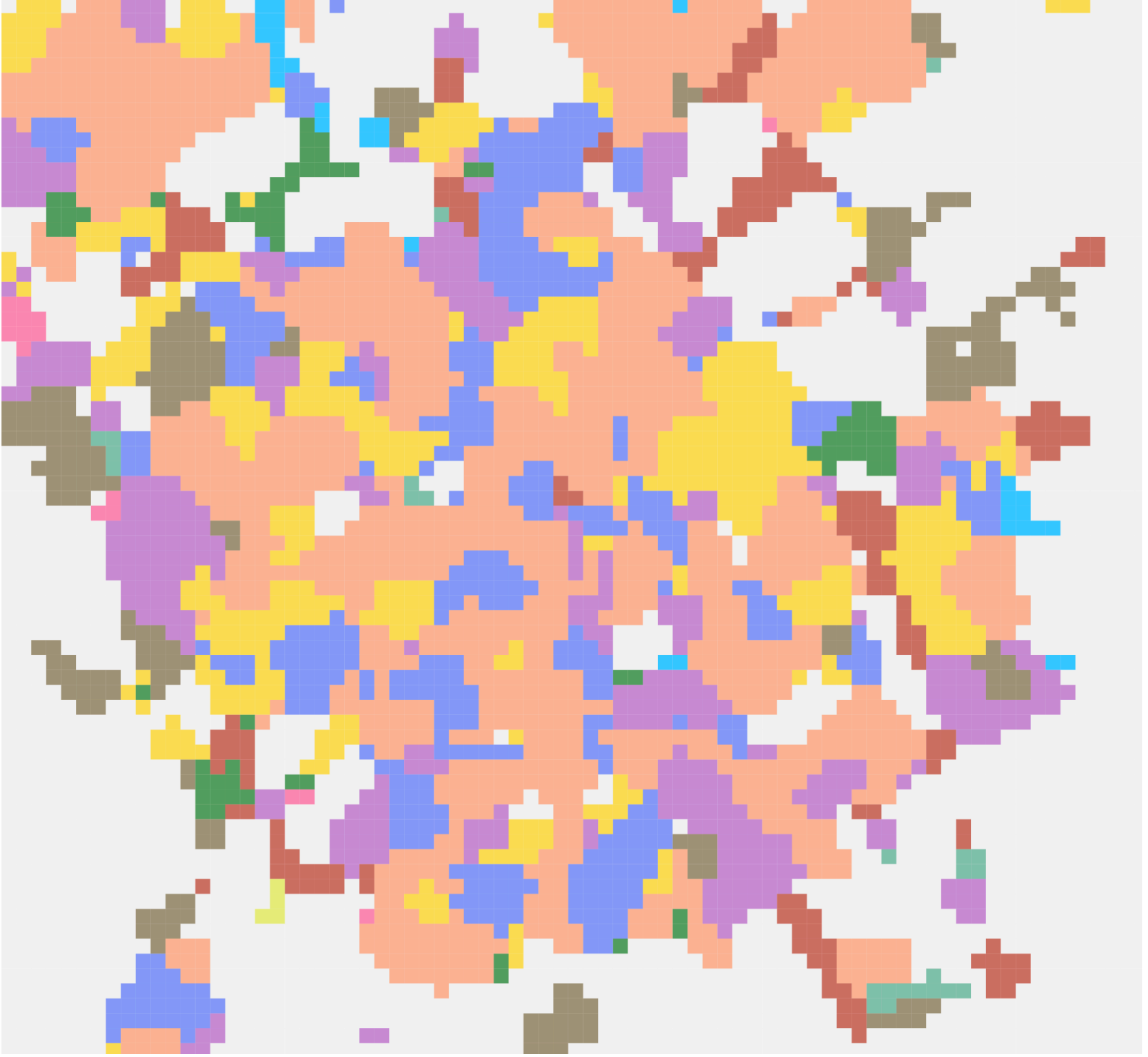


The City Stack

A Morphology-Based City Analysis and Generation Framework

TU Delft
MSc Geomatics
Thesis - P5

Oliver J. Post

Delft University of Technology

Supervisors

Hugo Ledoux
Delft University of Technology

Akshay Patil
Delft University of Technology

Co-Reader

Claudiu Forgaci
Delft University of Technology

Abstract

This thesis introduces the "city stack" framework and the concept of the "typology grid" to analyze and procedurally generate cities using publicly available geospatial data. It builds upon existing urban morphology research and combines existing metrics with new metrics to examine road networks and building footprints across 43 cities worldwide. The resulting unsupervised clustering of road patterns produced mixed outcomes: some global road typologies were consistently identified, while others lacked clarity or validity. Supervised classification of building typologies showed potential for city generation applications but faced challenges related to data quality and validation methods.

To encode the captured urban form, the typology grid was introduced, which allows for straightforward comparison between cities and links the analysis phase with the generation phase. While this approach simplifies complex urban patterns for better understanding, it may oversimplify details needed for effective city regeneration, needing future research into parameterizing the grid.

The simulated annealing optimization technique was applied to generate new city models to produce typology grids resembling those of actual cities. It proved a promising method, with relatively plausible results from just a few shape-based rules in the objective function. Computation time and grid size were a limiting factor, ruling out real-time use. The road and building generation methods based on the typology grid and city stack framework demonstrated the approach's feasibility but indicated that further refinement is necessary.

Further contributions by this thesis are an open source command line tool for analyzing the urban form real-life cities based on publicly available geospatial data, a proof of concept tool for generating cities, and a publicly available dataset of the analyzed cities.

In conclusion, the city stack framework and typology grids offer a viable method for capturing and generating urban form and can be used as a starting point for future research.

Acknowledgements

I want to express my sincere gratitude to the people that supported me throughout this journey.

Firstly, my supervisors, Hugo Ledoux and Akshay Patil. Thank you for all the invaluable feedback, support, and ideas you have given, for the interesting brainstorming, and the fun meetings we've had in the past year!

Secondly, thank you to the co-reader, Claudiu Forgaci. You've provided very important feedback and insight from the perspective of urbanism that helped the thesis forward tremendously.

A very big thank you to my girlfriend, who supported and encouraged me so much throughout this whole project!

I want to thank my mom, my dad, my brother, and oma Jose, who always supported me and were always keen to stay up to date and have interesting conversations.

Finally, I want to send my love to my grandpa, who sadly passed away in the last year. I miss you, opa en oma Post!

Contents

1 Introduction

This is a thesis about cities that don't exist. This doesn't mean fantasy cities from an alternate universe or sci-fi cities from the future. Cities that don't exist, but look like they could exist. This topic has fascinated me personally for a long time. Starting from age 12, I was building cities in the game *Cities XL*, which I later replaced with the newer *Cities: Skylines*. I always found it very difficult to make a city look realistic, especially the road network. Even the apparent simplicity of the North American grid seemed impossible to recreate without something feeling "off". With this thesis, I want to give this complex but fun challenge another try from an academic angle.

1.1 Scientific Relevance & Motivation

Generating cities is not just constrained to city-building games. In the field of Geomatics, three-dimensional city models are an important form of urban data that offer a way to integrate the many domains of a real-life city, including terrain, buildings, vegetation, and road networks. These models combine both geometry and semantics in a single framework. Increasingly, 3D city models are used in more and more research domains, among which estimation of solar irradiation, energy demand estimation, estimation of the propagation of noise, and computational fluid dynamics (CFD) for wind simulation (Biljecki et al., 2015).

Despite their growing application, the creation and availability of accurate 3D city models present significant challenges. These include the limited availability of existing models (Girindran et al., 2020), frequent geometric validity errors (Ledoux, 2018), and the intricate, labor-intensive process of creating detailed models (Girindran et al., 2020). While fully automatic generation using LIDAR data is feasible, it relies heavily on data availability and can still result in invalid geometries, which can be problematic for subsequent analysis or processing (Peters et al., 2022).

Parametrically generated city models propose a promising solution for numerous use cases. While many research topics need data on cities that exist in real life, a wide range of domains can utilize or even benefit from models of non-existing cities that offer parametric control over their creation. Implementing a parametric generator allows for the rapid production of new city models (Groenewegen & Smelik, 2009) and the easy application of city-wide changes, such as adjustments to building setback rules, which would be cumbersome manually.

Parametrically generated cities have diverse research and practical applications including urban and social simulations (Aschwanden et al., 2011; Badwi et al., 2022), serving as test files for algorithms, file formats and software, as input for parameter dimensionality reduction for Computational Fluid Dynamics (CFD) studies, and as training data for machine learning models. Other applied use

cases for these models include virtual reality, video games, animation, rendering, and as tools in urban and architectural design prototyping. It is important to clarify that these procedural cities are not intended as designs for new cities, which would encroach on the roles of architects and urban planners. Instead, they are realistic variations of existing cities to serve as tools in research, development, and visualization.

Besides these use cases for parametric cities, Kim et al. (2018) suggest that many research domains traditionally reliant on actual urban data could benefit from procedurally generated models. One advantage is eliminating quality issues associated with volunteered geographic information (VGI). Furthermore, population data in these models can be detailed yet fully anonymous, avoiding privacy issues linked with real-world data by synthesizing cities that resemble, but are not directly connected to, real locations, buildings, or people.

Another potential application is the testing and public demonstration of controversial urban plans, such as the placement of wind turbines. Using an existing city model for such tests could provoke local opposition if residents identify their surroundings in the test scenarios. A procedurally generated city, similar but not identical to the real one, allows for comprehensive planning and testing without the risk of controversy.

1.2 Problem Statement

There are numerous existing methods for parametrically generating cities or their components, see Section 3.1 for a review. However, there are problems, which have to be addressed for the models to serve their purpose. These methods often lack plausibility (Groenewegen & Smelik, 2009; Kim et al., 2018) as they are often based on generalized rule-based approaches instead of real city data. Real cities are complex and consist of many different types of urban tissue that are distributed within the city domain (Araújo De Oliveira, 2022). Existing methods that incorporate city layout as part of the methodology, require input maps created by designers or experts (Parish & Müller, 2001; Vanegas et al., 2012a). The only existing city generation methodologies that focus on producing realistic layouts do so based only on land use (Groenewegen & Smelik, 2009; Lechner et al., 2006). There exists an opportunity to generate city layouts based on urban morphology. Research exists on the study of urban tissue distribution within a city (Fleischmann et al., 2022), but this has not been connected with city generation. Using Inverse Procedural Modelling, the outcome of morphological analysis of real cities can become the input of existing city generation methods.

1.3 Research Question

To investigate solutions to this problem, I aim to develop a morphology-based city generation framework to connect urban tissue distribution research with existing and new methods for city generation. As a background for this investigation, the proposed case study aims to generate new cities with urban tissue characteristics similar to real-life cities.

The main research question of this thesis is:

How can a digital city model be procedurally generated to resemble the character of a real-life city?

The "character" of a city is defined as the composite of various urban tissue areas comprising the city and their spatial interrelationships.

This question is broken down into the following sub-questions:

- *How can the urban form of real-world cities be captured using publicly available geospatial data?*
- *How can the captured urban form be encoded in a way that allows for the comparison of different cities and the generation of new cities with similar character?*
- *How can this encoded data be utilized to procedurally generate a digital city model that resembles the form of the encoded real-life city?*

1.4 Proposed Framework

The hypothesis of this thesis is that the "city stack" framework offers a way to link urban morphology analysis with city generation. The city stack describes a city in terms of individual layers that are connected by constraints, see Figure 1. The stack allows a city to be analyzed layer by layer, where the layout of each layer is described using distinct locations (PoI, city center, etc.) or distinct areas (i.e., road systems and buildings). All layers combined form the *character*. For example, a location's character could be described as a "Hilly location near the city center with grid-like major roads, a big hospital, and an organic street pattern."

This is a flexible approach that allows for the insertion and deletion of layers anywhere in the stack. I focus specifically on layers that that are ambiguous to divide into distinct categories or areas, specifically the (minor) roads layer and the buildings layer. The approach used for this is based on research into programmatically classifying areas of similar urban tissue (Fleischmann et al., 2022). The

methods in this research are adapted. Notably, that classification happens separately per layer in the city stack instead of one combined classification based on buildings, plots, and roads.

Even though both analysis and generation often happen separately per layer, the layout of these different layers is connected. For example, industrial buildings might usually be close to major roads and/or waterways, the streets around industrial buildings are more likely to have a grid or orthogonal pattern than an organic pattern, and the city center is more likely to be next to the river than high in the mountains. The connections can be described in multiple directions; for example, a stadium might be placed near existing major roads, or a new major road might be built to improve access to the stadium. These multi-directional connections are both challenging to analyze and to take into account when generating new cities.

Therefore this thesis proposes simplifying these constraints in the form of the city stack. It proposes that there is a way to organize the different layers of the city in a stack where layers higher in the stack form constraints for layers lower in the stack, and only in that direction. Because of this single direction, city generation can generate layers one by one by going down the stack. The constraints are visualized by the magenta lines in Figure 1.

The generation pipeline can then be as follows:

- Analyse the city stack of real cities, layer by layer
- Encode the city stack of real cities in a standardized template
- Generate the city stack layer by layer for a new city with this template as input

User input is important in city generation (Bidarra et al., 2010). Therefore, the layout analysis and generation must be intuitive, comprehensible, and adjustable. Some layers are very unambiguous. For example, the location(s) of points of interest, like airports or the planar partition of land use areas, which is clearly divided into distinct categories. These are easy to understand. This thesis explores how more ambiguous layers can be analyzed. These also happen to be the layers that contribute a lot to the visual form of the city (mainly street patterns and buildings). Research exists on how these layers can be divided into distinct areas of similar urban tissue (Fleischmann et al., 2022). In the city stack, each of these areas is assigned a distinct typology, which is consistent for any city across the world and easy to understand. Examples of building typologies could be industrial buildings and apartments, and street patterns could be grid patterns, mountain roads, and dense cores. I test out methods using both supervised (classification) and unsupervised (clustering) machine learning techniques for the determination of these typology areas, using research by Fleischmann et

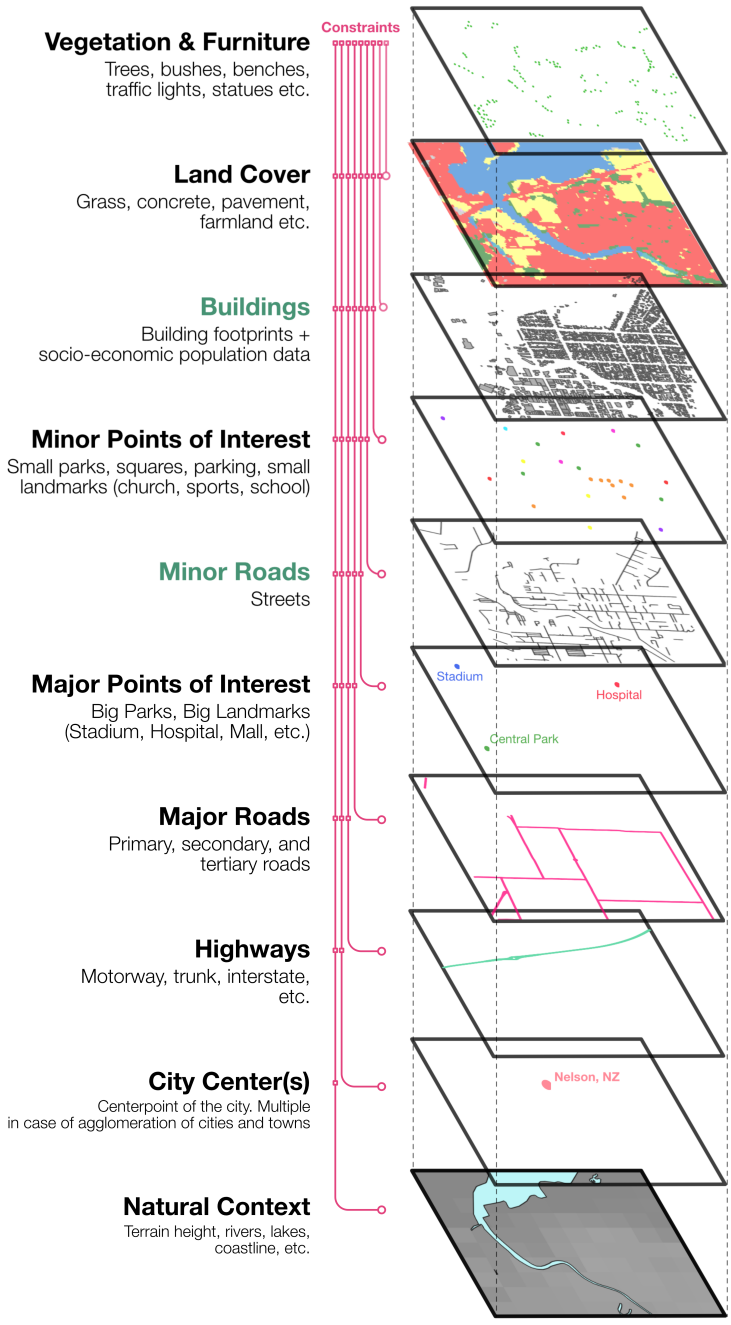


Figure 1 - Schematic representation of the city stack. Highlighted in green are the layers that are investigated in this thesis. The magenta lines show that layers lower in the stack form the constraints for layers higher in the stack. Figure contains data from OpenStreetMap contributors (2017), Pesaresi (2023) and Zanaga et al. (2021).

al. (2022) as a starting point. Finally, the "typology grid" is introduced to encode these typology areas in a format suitable for generating similar cities in a new input, adjusting their context and allowing user interaction.

The typology grid forms the base for the generation phase. For each ambiguous layer, a new typology grid is created that is adapted to a new context and new user parameters, while remaining as similar in character as possible as the original typology grid. I investigate the use of the simulated annealing optimization technique for this step. Once the typology grid of a layer is finished, existing methods for city generation are employed to generate the individual elements (building footprints and road segments).

1.5 Scope

Cities are endlessly complex structures, from their general structure to the individual bricks of a building. This thesis deliberately narrows its scope to manage both the analytical and generative processes effectively.

The main aim is to test the feasibility of the city stack framework and the typology grid concepts both in the analysis and generation phases. Because the framework heavily relies on the ability to distill any city into a set of distinct typologies, one of the main focuses is creating a pipeline for analyzing any city across the world and encoding it into typology grids. A big part of this is testing clustering and classification approaches to determine these typologies.

The generation step is limited to testing a proof of concept for generating new typology grids and conceptually testing ways of adapting element generation techniques based on existing methodologies.

The aim of this thesis is not to create a fully functional workflow from start to finish but to lay the groundwork for a pipeline and test its feasibility.

Scope

This thesis:

- does not aim to implement the complete city stack framework. Instead, it aims to test its feasibility through proof of concepts for the minor roads and buildings layer.
- is restricted to 2.5D analysis, simplifying the complexity and enhancing data availability.
- avoids methods that produce results through "black box" processes, specifically excluding the use of deep learning techniques. This does not exclude the use of all machine learning techniques, as both clustering and classification methods exist that offer statistical insight into why features are classified as a certain class/cluster.

2 Theoretical Background

2.1 Morphology & Urban form

To generate plausible cities, an understanding of urban structure is crucial. Urban morphology, the study of the physical form of cities, provides a robust theoretical foundation for this. Araújo De Oliveira (2022) explains that urban morphology has diverse definitions. Key descriptions include:

> [Urban morphology is] an approach to conceptualizing the complexity of physical form. Understanding the physical complexities of various scales, from individual buildings, plots, street blocks, and the street patterns that make up the structure of towns helps us to understand how towns have grown and developed.
>
> — Larkham (2005)

> A method of analysis which is basic to find(ing) out principles or rules of urban design.
>
> — Gebauer & Samuels (1981)

Araújo De Oliveira (2022) states that cities are, morphologically, extremely complex objects. To deal with these complexities, the city is divided into distinct elements of different scales that can be isolated from their context. He mentions from small to large scale: the natural context, the streets system, the plots system, and the buildings system.

Kropf (1996) defines Urban Tissue as the character of an urban environment, which emerges from the interplay of the scale layers of the city. The nature of urban tissue is dependent on the "resolution", the scale we are looking at the city. At a high resolution, the materials of individual buildings contribute, among other factors, towards the character of the city, while at a low resolution, the urban tissue only includes street patterns and block shapes (Araújo De Oliveira, 2022). Urban tissue not only varies from city to city but also within different zones of the same city.

This means the different scale layers can be analyzed separately and then combined to define the city's character in the form of urban tissue. Since the urban tissue can differ within the city, this analysis is not city-wide but in terms of zones within the city.

2.2 Clustering & Classification

Both clustering and classification are machine-learning techniques for grouping data into distinct clusters/classes.

Clustering does this by organizing data into groups based on similarity. For this process, it is not necessary to decide in advance what these groups are, and it's also not necessary to create a training dataset where the data is already marked to which group it belongs. This makes it an "unsupervised" technique. In clustering, data points within the same group, or cluster, are more similar to each other than to points in other groups. This unsupervised learning approach is commonly applied in fields like pattern recognition, image analysis, bioinformatics, and data mining (Madhulatha, 2012).

Clustering methods are categorized into several types, including hierarchical, density-based, grid-based, and model-based approaches. Each method has unique strengths, depending on data characteristics and the application context (Saxena et al., 2017).

Classification on the other hand is aimed at assigning predefined labels to data points based on learned patterns. Unlike clustering, classification is a supervised learning approach requiring labeled training data to guide the learning process. Common classification techniques include decision trees, support vector machines (SVM), k-nearest neighbors, and neural networks, each offering distinct advantages depending on data characteristics and application needs (Aized Amin Soofi & Arshad Awan, 2017).

2.3 Simulated annealing

Simulated annealing is an optimization technique inspired by the process of annealing in metallurgy, where a heated material slowly cools to create a highly organized structure. This process was translated by Kirkpatrick et al. (1983) into a method for finding a global minimum in a solution space by iteratively exploring potential solutions and occasionally accepting suboptimal ones to escape local minima, mimicking the random energy changes in physical annealing.

This method works as an iterative process that gradually reduces a "temperature" parameter. At higher temperatures, the algorithm has a high probability of accepting worse states, helping it explore the problem space. As the temperature lowers, the probability of accepting worse solutions declines, leading toward convergence on an optimal or near-optimal solution. This dynamic helps the algorithm avoid getting trapped in local minima, a common limitation in deterministic approaches like gradient descent (Trosset, 2001).

The effectiveness of simulated annealing depends on the specific cooling schedule and acceptance criteria, which are adjusted based on the problem's complexity and solution space characteristics (Aarts & Laarhoven, 1989).

3 Related Work

3.1 City Generation

Many methods for generating (parts of) cities exist, and these can be categorized. Kim et al. (2018) suggest categorizing procedural city generation techniques into generative grammar, simulation-based, tensor field, stochastic, data-driven, and inverse procedural generation. Conversely, Smelik et al. (2014) propose a slightly different classification that includes artificial intelligence and computational geometry but excludes inverse procedural generation and tensor fields. This thesis utilizes a hybrid of these classifications to explore the range of available methods, but excludes stochastic methods, as they are primarily used for terrain generation (Kim et al., 2018; Smelik et al., 2014), and artificial intelligence methods, as both are outside the scope of this thesis.

The following is a summary of notable techniques from each category.

3.1.1 Generative Grammar

Shape grammars consist of rules that define the structure of a language, which can be used to generate geometry. For instance, Lindenmayer systems (L-Systems) were initially developed to model organic structures (Prusinkiewicz & Lindenmayer, 1996). Parish & Müller (2001) adapted this concept to generate road networks from input image maps such as terrain and population density. The process involves constructing the road network segment by segment, where the initial angle of each new segment is influenced by global goals. Adjustments like pruning, rotation, and snapping are applied to ensure each segment conforms to local constraints. Originally this L-system uses string reformatting, but Barett (2009) describes a way to efficiently implement this as an algorithm based on a priority queue. This system produces plausible results, at first glance looking like a road pattern that could exist in a real city, and offers the possibility of extending it by modifying the global goals and local constraints. Downsides include that it depends on input maps for the generation, and the limited typologies of street patterns it can generate (grid, organic, or radial). See Figure 2 for an example of a street pattern generated with this method.

In addition to street generation, generative grammars are widely used in procedural building generation systems (Smelik et al., 2014). One basic approach involves combining extruded primitives of varying heights (Greuter et al., 2003), see Figure 3. Another technique employs L-systems on a rectangular floorplan, facilitating automatic Level of Detail (LOD) generation (Parish & Müller, 2001). Müller et al. (2006) developed the Computer Generated Architecture (CGA) shape grammar to produce more intricate geometries.

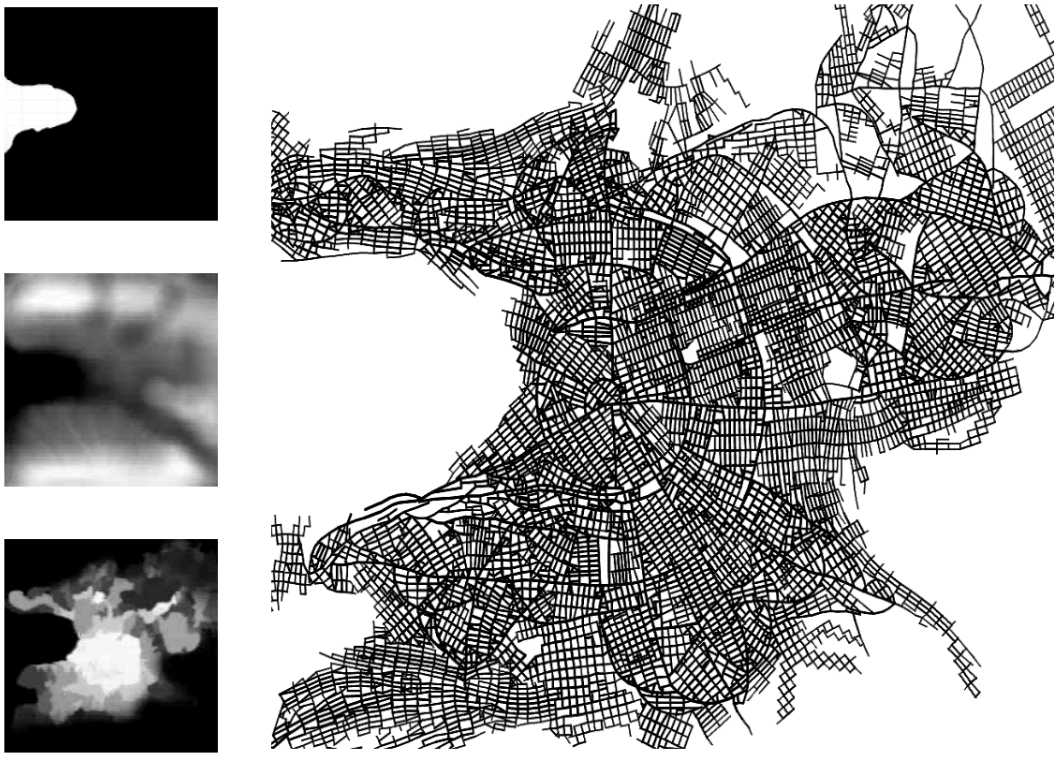


Figure 2 - Example result from the L-System approach from Parish & Müller (2001), with the input water, terrain, and population map on the left (Figure from Parish & Müller (2001))

The commercial software CityEngine integrates L-systems for street layout and CGA for building generation (Esri R&D Center Zurich, n.d.). The major downside of shape grammars are their complexity and the required expertise to use them

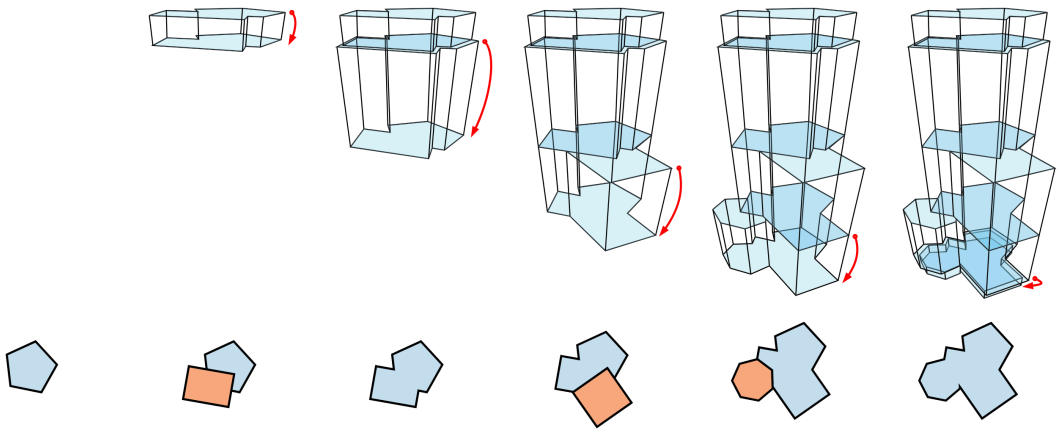


Figure 3 - Example of extrusion method by Greuter et al. (2003) (Figure from Greuter et al. (2003))

(Smelik, 2011). This makes these methods less accessible for non-experts to use, and therefore decreases their applicability.

3.1.2 Simulation-Based

Weber et al. (2009) combines the L-System from Parish & Müller (2001) with traffic simulation to model city growth over time. This integration yields plausible outcomes, though scaling to larger cities remains a challenge. Lechner et al. (2006) employs agent-based simulation to determine land use within a raster representation of a city domain, see Figure 4. The land use patterns generated appear realistic, with industrial and commercial areas grouped around major roads; however, drawbacks include significant computation time and reliance on arbitrary rule sets based on assumptions rather than empirical data.

3.1.3 Tensor Field

Chen et al. (2008) introduce tensor fields as an intuitive and flexible framework for generating the streets system; see Figure 5. They argue that street patterns mostly consist of two dominant directions and they utilize the properties of tensor

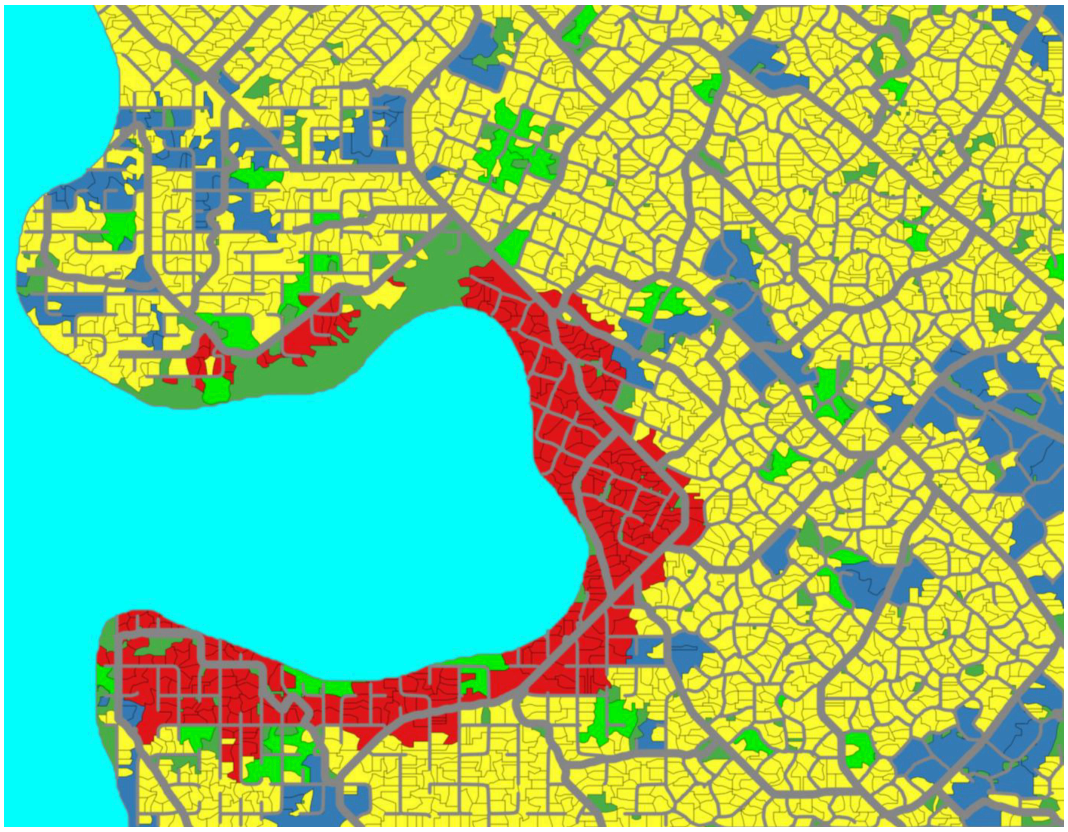


Figure 4 - Example of land-use simulation by Lechner et al. (2006). (Figure from Lechner et al. (2006))

Figure 5 - Example of tensor field road generation (Figure from Chen et al. (2008))

fields to generate major and minor roads from these mathematically derived directions. A big advantage of this method is the ease of use, as the patterns of the tensor field are reasonably intuitive and easy to edit manually. The results look plausible for North American gridiron city centers but are limited in plausibility for other urban fabric types.

3.1.4 Data-Driven

Aliaga & Vanegas (2008) describe a technique that uses an existing road network as a template to generate extended road patterns while maintaining a similar structure; see Figure 6. Nishida et al. (2016) further this approach with their *grow*, *warp*, and *blend* operations. These techniques generate plausible results for extending an existing example but currently do not seem suitable for producing plausible city-wide street patterns.

3.1.5 Inverse Procedural Generation

Inverse procedural generation addresses the challenge of complex input parameters in traditional procedural generation, which often requires specialized knowledge and experimentation. This method determines optimal procedural inputs based on predefined goals or examples.

It does this by deriving procedural rules from an existing model to generate similar structures or patterns. Unlike forward procedural modeling, where predefined rules create models, IPM works backward, discovering rule sets that can reproduce or extend the structure of the input model (Aliaga & Vanegas, 2008).

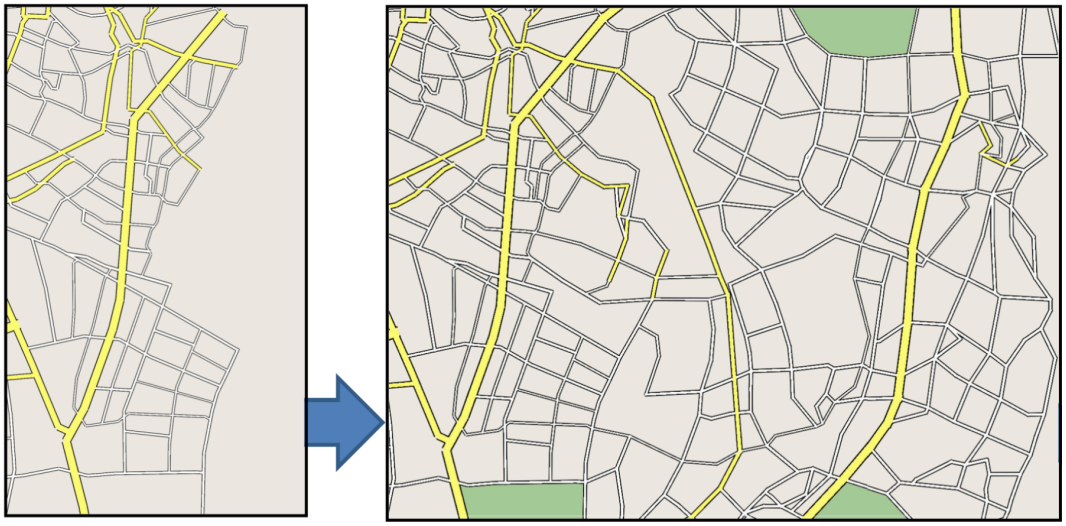


Figure 6 - Example of data-driven road network generation (Figure from Aliaga & Vanegas (2008)))

Vanegas et al. (2012a) introduce such a method for city generation. One example within this work is the user-defined target of optimized sun exposure, where the model figures out the optimal procedural parameters to achieve this. The main advantage of this method is that a procedural model can be reused to optimize for new target indicators without having to reprogram the original model.

3.1.6 Computational Geometry

A variety of geometric methods are employed for parcel generation. Generating a Voronoi diagram based on points that represent building locations typically yields unconvincing results. Kelly (2014) introduces a straight-skeleton method for creating *modernist* parcel divisions typical of planned suburbs, which produces highly plausible outcomes but is specific to this parcel style. For other parcel types, Kelly (2014) suggests using Object Oriented Bounding Box (OOBB) methods. Vanegas et al. (2012b) adapt the parcel generation technique from Parish & Müller (2001) by recursively splitting the OOBB until a user-specified criterion is met, see Figure 7. An alternative method is proposed by Emilien et al. (2012) and uses a multistep process of anisotropic land conquest. This method was developed for generating rural landscapes but seems promising for urban landscapes. It has the ability to follow predetermined rules and costs.

3.2 City Analysis

Previous work on programmatically classifying typologies and urban tissue in cities includes analyses separately for street patterns (Section 3.2.1), buildings

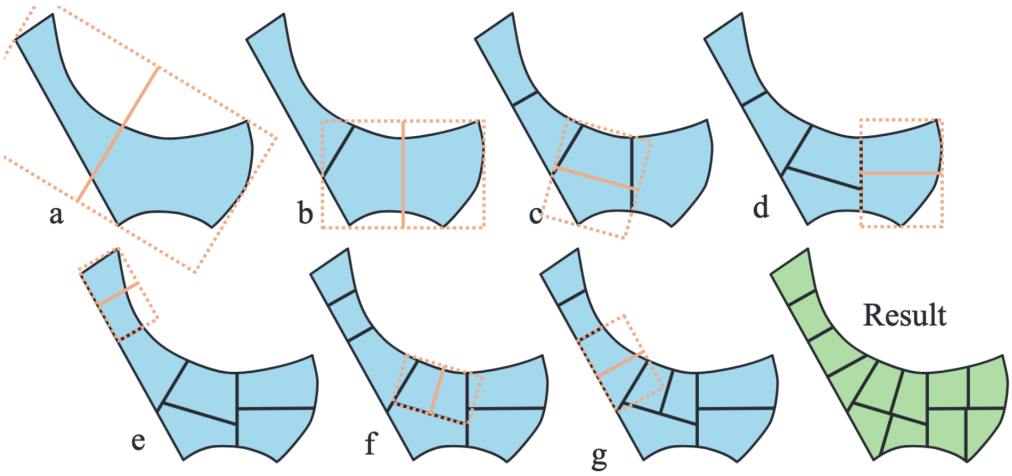


Figure 7 - Methodology of Object Oriented Bounding Box guided parcel generation by Vanegas et al. (2012b) (Figure from Vanegas et al. (2012b))

(Section 3.2.2), and analyses that combine multiple layers for one classification (Section 3.2.3).

3.2.1 Street Pattern Classification

Street pattern typologies exist that are associated with specific locations and time periods. For example, Figure 8 shows street pattern typologies specific to specific periods of urban development in the United States (Southworth & Ben-Joseph, 2003). For city generation, however, a set of typologies is needed that can be applied globally.

Several studies have applied clustering to identify urban form typologies based on morphological metrics. Badhrudeen et al. (2022) and Louf & Barthelemy (2014) have used clustering on a global scale, aggregating metrics from individual roads within a city to articulate street pattern characteristics. Both methods prove the applicability of clustering for identifying street patterns

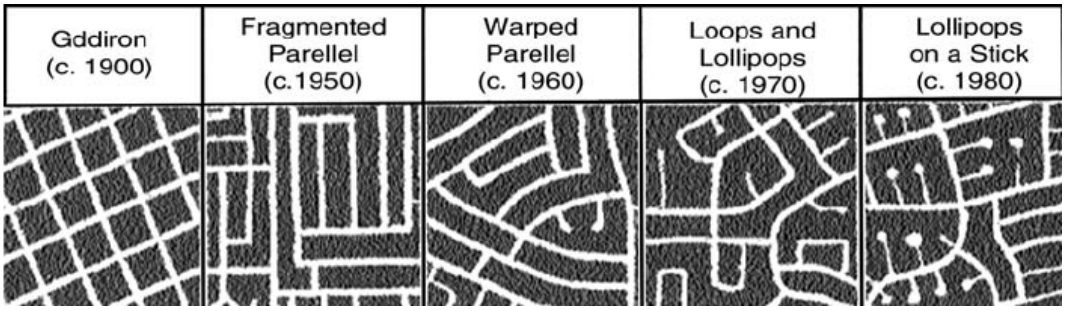


Figure 8 - Example of a location and time-period based street pattern typology. (Figure from Southworth & Ben-Joseph (2003))

on a global scale. However, they do not differentiate between areas within the same city, which is a necessity for street generation. Also, the granularity of both approaches is low, with Badhrudeen et al. (2022) identifying the classes Gridiron, Long Link, Mixed, Organic, and Hybrid and Louf & Barthelemy (2014) identifying four unnamed classes.

Marshall (2005) describes the ambiguousness of the street pattern and how many ways it can be defined. City-wide descriptions can be used like from Lynch (1981), including types like radial patterns, rectangular grid cities, and baroque axial networks. These are the sort of typologies derived by Badhrudeen et al. (2022) and Louf & Barthelemy (2014). A city consists of multiple patterns, however, that can be described individually. Marshall (2005) proposes a taxonomy of patterns, based on both composition and configuration. Composition is associated with geometry, including aspects like length and area ratios, and is typically expressed in real numbers. Configuration is associated with topology, including aspects like connectivity and adjacency, and is typically expressed in integer numbers. The typology aims to distinguish types both in configuration and composition. The taxonomy separates patterns with and without cells or circuits. Since most of a city's street pattern consists of blocks, I only consider the cellular patterns in the taxonomy, see Figure 9.

This type of taxonomy forms a solid theoretical foundation but can be difficult to apply in practice. A big advantage is that each type comes with its own set of rules, which could correspond to different generation algorithms. However, to analyze real cities by programmatically classifying streets according to this typology, a ground truth dataset would be needed. Due to the complex nature of street patterns, the decision as to what type a certain segment belongs to would, in many situations, be difficult and arbitrary. Besides this point, it also offers a very high amount of granularity in rectilinear patterns while offering little granularity in the common "organic" pattern.

An opportunity exists to create a global, neighbourhood-scale, set of street pattern typologies. Since the distribution of street patterns within a city is vital for generation cities with similar character,

3.2.2 Building Classification

Previous research has taken two approaches to classifying buildings. One focuses on deriving a new classification from data (Schirmer & Axhausen, 2015a); the other starts from a predefined classification and applies it to a case study (Hartmann et al., 2016; Steadman et al., 2000; Steiniger et al., 2008; Wurm et al., 2016).

Starting with the first approach, (Schirmer & Axhausen, 2015a) uses clustering to programmatically derive building typologies without specifying a predefined

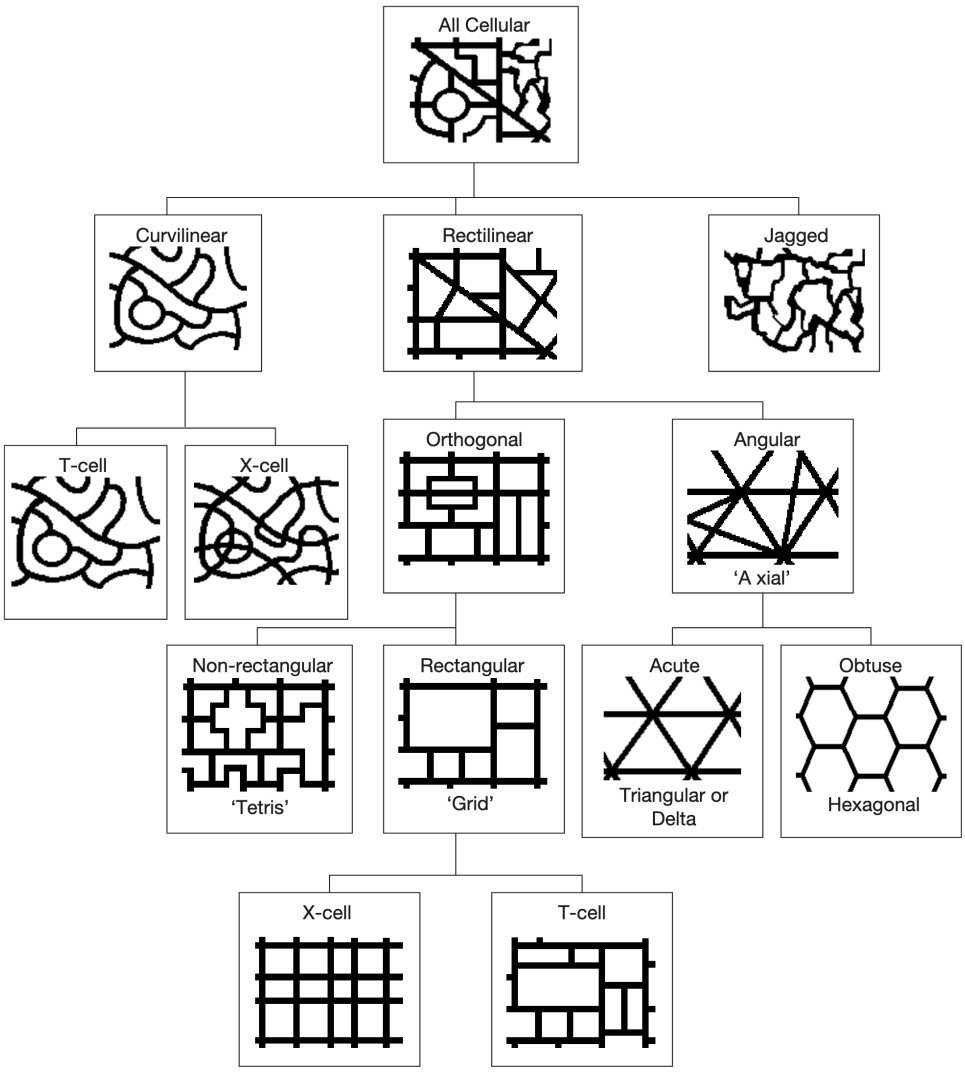


Figure 9 - Cellular typologies of the street pattern taxonomy by Marshall (2005) (Figure from Marshall (2005))

set of typologies. The final derived typologies are labeled as linear housing, large-scale buildings, punctual development, and buildings with multiple wings when clustering on buildings individually and street-aligning buildings (Figure 10.1), block-defining residential (Figure 10.4), block-filling solitaries of large scale (Figure 10.3), and detached punctual forms that form a block (Figure 10.2) when clustering when taking the neighborhood of each building into account.

The downside of this approach is the interpretability of the results and the sensitivity to the input data and outliers.

The other methods use a predefined set of typologies.

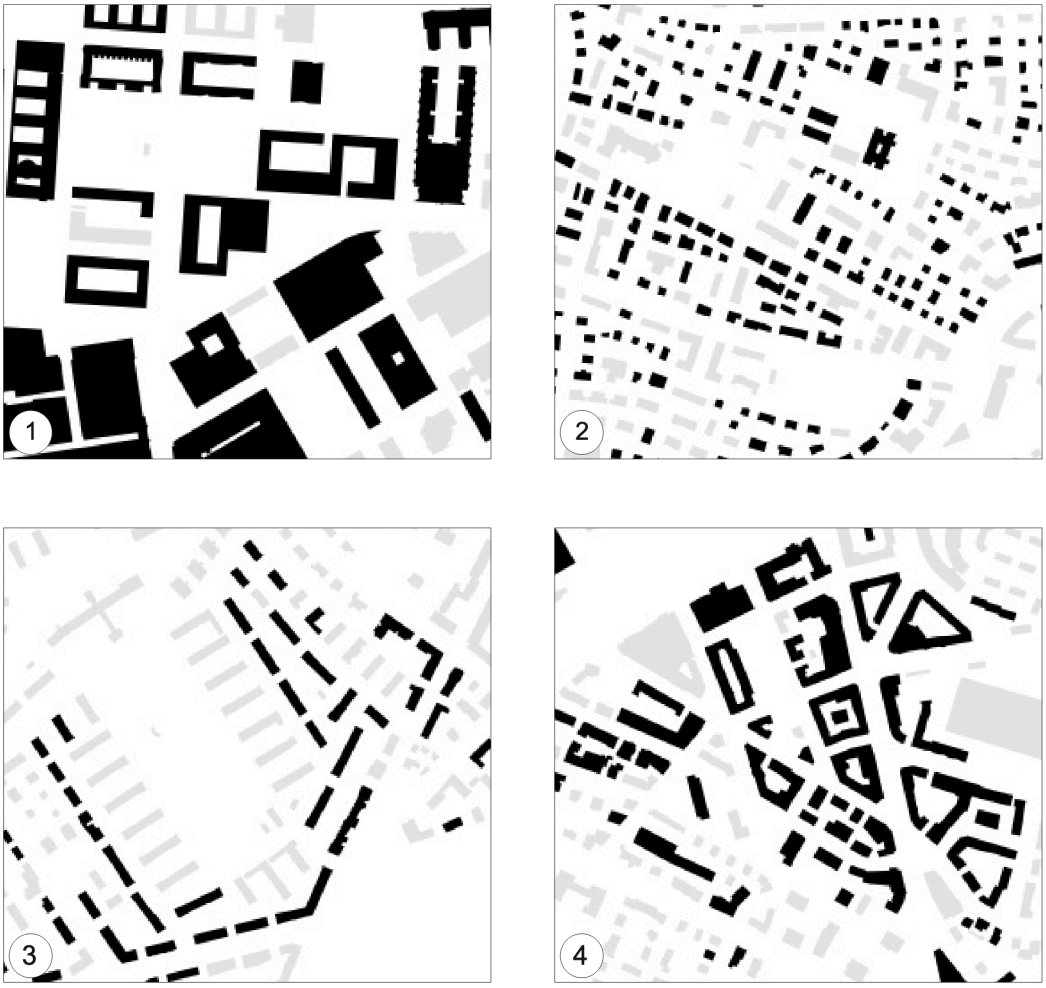


Figure 10 - Example of clustering result by Schirmer & Axhausen (2015a) taking neighborhood into account, containing street-aligning buildings ([1]), block-defining residential (4), block-filling solitaries of large scale (3), and detached punctual forms that form a block (2). (Figure from Schirmer & Axhausen (2015a))

Episcope (2014) sets up a set of building typologies for Europe with more granular country-specific typologies and four overarching typologies: "single-family houses, terraced houses, multi-family houses, and apartment blocks". This classification is not applicable to this thesis due to it only including residential building typologies and the granularity being country specific. Steadman et al. (2000) proposes a classification that encompasses all building types, but it is focused on energy analysis and separates buildings based on materials, daylight, activities, and services besides shape and use characteristics. This results in a typology that is too granular in aspects that are not relevant for city generation. (Poon, 2024) uses the granular country-specific (Netherlands) set of typologies for programmatical classification and shows it's possible to achieve accuracies

between 60 and 98 percent. This highlights the importance of the morphological distinction between types on classification performance, as lower accuracies were observed in morphologically similar building typologies like variations of different apartment buildings.

Hartmann et al. (2016) utilizes a set of typologies based on land use, the distinction between main and outbuilding, and the morphologic type. This results in 6 residential building types (detached, semi-detached, and terraced with both purely residential and mixed-use variants) and 20 non-residential types (i.e., detached transport outbuilding, undetached industrial & commerce main building). This typology offers a lot of granularity but is more focussed on land use than on morphological character. For example, it doesn't distinguish between a big residential apartment block and a single detached house, or between a terraced house and a house in an inner city block.

Steiniger et al. (2008) uses a classification that is more focused on morphology and neighborhood characteristics. In Figure 11 you can see the 5 types, with a distinction between industry and all other types, where the other types correspond to density and urban tissue. This approach is more suitable for generation and can be used as a starting point. These typologies are, however, more focused on entire areas, whereas the finest level of detail in city generation would have granularity at the building level (i.e., apartment buildings mixed with terraced housing).

The typologies used in Wurm et al. (2016) are highly based on morphology, separating buildings into perimeter block development, block development, terraced/row houses, detached/semidetached, and halls (i.e. industrial). These typologies correspond to the different building generation methods that would be used to generate the different typologies. Still, the typology is based on German cities and, therefore, highly specific to that location. Also, common typologies like (slab) apartment buildings are missing, and the granularity of the "halls" typology is low as both industrial, commercial, service, agricultural, and more could fall within this category.

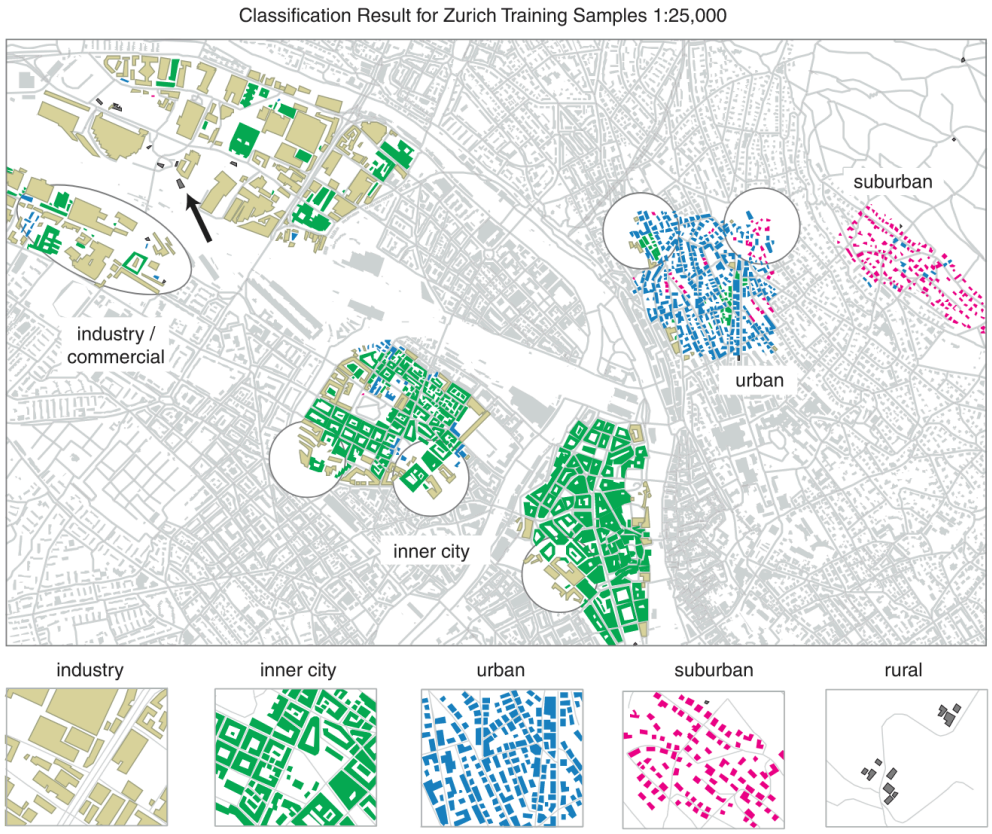


Figure 11 - Example of the building typologies from Steiniger et al. (2008) (Figure from Steiniger et al. (2008))

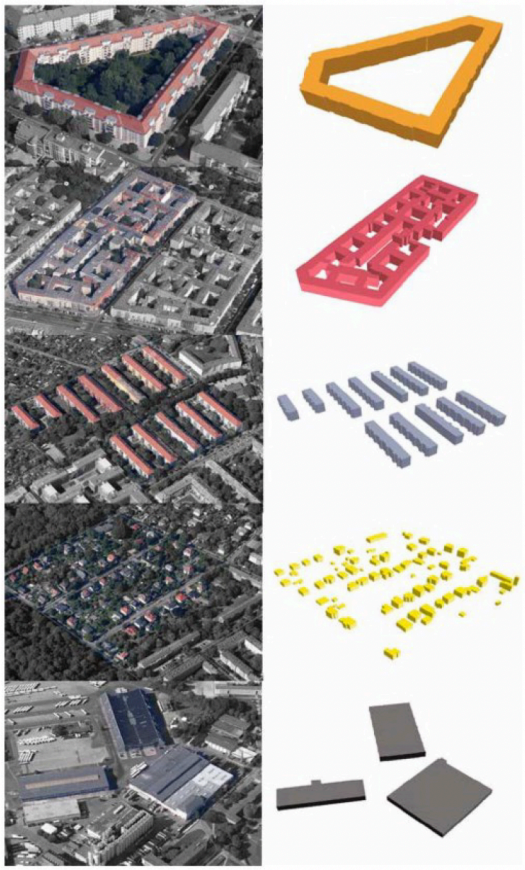


Figure 12 - Example of the building typology by Wurm et al. (2016), from top to bottom perimeter block development*,* block development*,* terraced/row houses*,* detached/semidetached*, and* halls*. (Figure from Wurm et al. (2016) with background map from Google Earth (2024))*

3.2.3 Combined Classification

Other research on classification methods does not distinguish between different types of morphological elements (e.g., street patterns and buildings). Instead, it aims to classify urban tissue as a whole. Examples of these are a clustering approach based on morphological metrics (Fleischmann et al., 2022) and a machine learning approach based on computer vision (Wang et al., 2024).

Wang et al. (2024) shows that the computer vision approach is a viable alternative to morphological analysis, effectively identifying different urban tissues with multiple city case studies. However, this approach remains

Fleischmann et al. (2022) show the viability of the clustering approach, using unsupervised learning on morphological metrics of both elements themselves and their direct neighborhood to detect contiguous areas of the urban tissue.

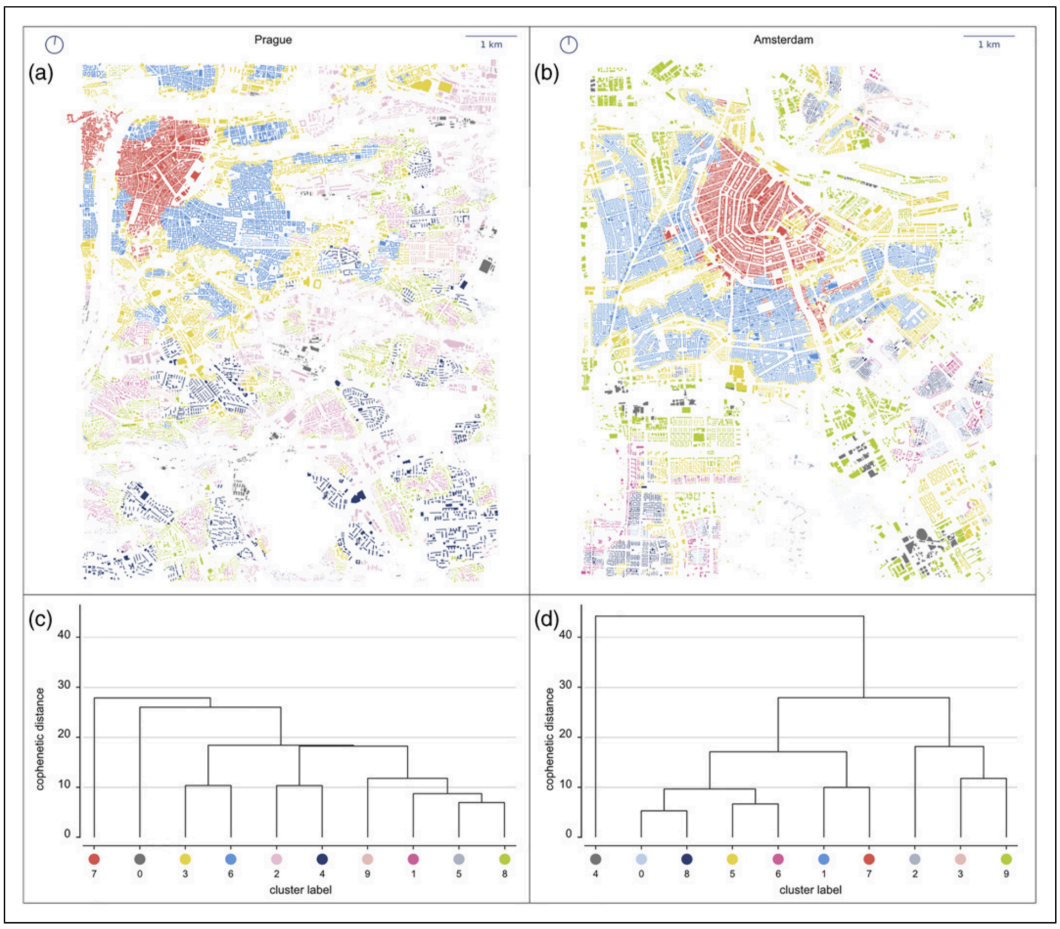


Figure 13 - Results from clustering approach of Fleischmann et al. (2022) applied to central Prague (a) and central Amsterdam (b) accompanied by dendrograms representing the results of Ward's hierarchical clustering (Figure from Fleischmann et al. (2022))

Figure 13 shows how this approach can be applied to individual cities to derive a clustering valid for that specific city.

Fleischmann et al. also introduce a software library Momepy (Fleischmann & PySAL Developers, 2018) to reproduce the metrics used. This paper forms an invaluable source for the applicability of deriving urban tissue areas programmatically using open geospatial data and forms the starting point for the city analysis in this thesis. There are, however, some downsides to this approach, some specifically for the aim of city generation. Firstly, the fact that clusters are not labeled with a descriptive name, and are the result of clustering on more than 200 metrics, the comprehensibility could be improved. Areas are marked as belonging to a different type of urban tissue, but finding out how the urban tissue differs between the two areas would involve interpreting the differences

between hundreds of metrics describing buildings, streets, and plots. Secondly, even though Fleischmann et al. show that the method can be applied to derive universal clusters across multiple cities, it has challenges for applying it on a global scale. Since multiple morphological layers (buildings, streets, and plots) are clustered together, there is no separation between street pattern typology and building typology. Analysis-wise, this will likely result in a much larger amount of needed classes to achieve the same granularity as could be achieved with separate classes for building typology and street pattern typology. Every possible combination of building typology and street pattern typology would need a new class in a combined typology, but with separate typologies per layer it would be possible to make all possible combinations with a limited set of classes per layer. Classifying layers separately has the added benefits of choosing different classification methods for each layer based on suitability and effectiveness and the possibility of adding more layers (i.e. vegetation, land cover, etc.) without having to recompute all combined classes and without increasing the number of classes, even more, to keep the same granularity (i.e needing two different classes to separate terraced housing in a gridded street pattern on predominantly concrete and terraced housing in a gridded street pattern on predominantly grass). Finally, specifically for city generation, it is likely that different classes within a layer correspond to different generation algorithms (i.e., detached housing is generated differently than city blocks). Separate classes per layer, therefore, also follow the generation process more closely.

All in all, the existing work on different methods of classifying morphological elements in cities forms a solid foundation to develop a framework for city generation based on programmatical analysis of real-life cities.

4 Methodology

4.1 Overview

The three sub-questions of the thesis can be translated into three methodology steps

Question 1: *How can the urban form of real-world cities be captured using publicly available geospatial data?*
Step 1: **Analyse** the character of real-world cities by capturing the urban form separately for each layer in the city stack.

Question 2: *How can the captured urban form be encoded in a way that allows for the comparison of different cities and the generation of new cities with a similar character?*
Step 2: Encode the analysis results for every individual city and derived typology in template files.

How can this encoded data be utilized to procedurally generate a digital city model that resembles the form of the encoded real-life city?
Step 3: **Generate** a new city, layer by layer, based on these templates

4.1.1 Analyse

How to capture the urban form for a layer in the city stack? This depends on what layer we are talking about. The point of interest layers define specific locations of distinct types of points of interest (i.e. a stadium or a mall), which calls for a different analysis method than layers that are better described as areas of similar typology. I aim to investigate the analysis of these more ambiguous typology areas, to test if this analysis is a feasible starting point for city generation. The layers I analyze are the *buildings* layer and the *minor roads* layer, but the same approach can be extended for layers like *major roads*, *highways*, and possibly *vegetation & furniture* as all these layers describe areas of similar character that do not fall within a distinct unambiguous set of typologies (like land cover or land use would).

Fleischmann et al. (2022) laid the groundwork for a programmatical approach to determining these areas, but their approach determines these areas for all layers combined instead of separate areas for each layer. For comparability and generation reasons, see Section 3.2.3, I propose a method that determines areas separately for each layer:

The hypothesis is that the urban form of each separate ambiguous city layer can

be described by partitioning the layer domain into distinct areas of similar urban tissue, see Figure 14. Each of these areas belongs to a distinct set of typologies that is valid for every city, so cities can be compared globally. These typologies are given a descriptive name to make them less ambiguous, resulting in a set of distinct typologies describing the ambiguous layer. For example in the case of the *minor roads* layer, a city in the USA might consist of areas belonging to typologies called *gridded streets*, *curvy suburban streets*, *winding mountain roads*, and so on. A city in Europe might also have areas of these same typologies, and also would include typologies that might not exist in the American city, like *dense organic streets*. In the case of the buildings layer, a European city might consist of areas of typologies called *city block*, *terraced housing*, *apartment*, etcetera. The American city might predominantly consist of the *detached housing* typology.

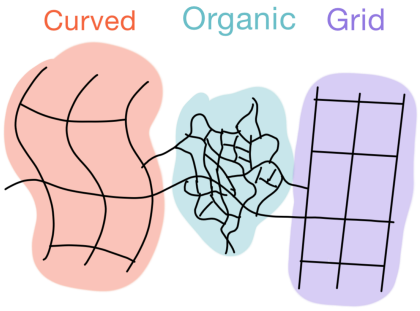


Figure 14 - Illustration of how an ambiguous city layer can be described by partitioning the layer domain into distinct areas of similar urban tissue.

These areas are determined by analyzing the individual elements of the city layer, following the approach followed by Fleischmann et al. (2022), but separate for each layer. These elements are the street segments for the streets system layer and individual buildings for the building layer. Each individual element is assigned to belong to a distinct typology based on the characteristics of the element itself and of the neighboring elements around it. For example, a specific street segment is determined to be part of a *gridded* street pattern, or a specific building is determined to be part of the *perimeter city block* pattern.

Different techniques are used for the streets layer and for the buildings layer. In the case of the buildings layer, it is possible to predefine a list of distinct topologies. The architectural and functional characteristics of buildings make it possible to make clearly distinguishable typologies. For example, a detached residential house is a clear typology and is not easily confused with a terraced house.

This thesis uses the following building typologies:
- Detached housing
- Terraced housing
- Filled city block
- Perimeter city block
- Irregular blocks
- Apartment buildings
- Industrial buildings
- Complex buildings
- Big commercial buildings

See Figure 47 for examples of each typology.

The set of typologies was derived from using the classifications from Steiniger et al. (2008) and Wurm et al. (2016) as a basis and adding granularity based on visual investigation of maps of cities across the world, using OpenStreetMap. For example, the *irregular blocks* class was added to represent dense city blocks of mostly detached buildings in a highly irregular pattern, as was observed in multiple cities on the Asian, African, and South American continents.

Each building in the buildings layer is assigned to one of these typologies using supervised machine learning (classification) based on a range of numerical and categorical attributes of the building and its neighborhood. Some simple examples of these attributes are footprint area, shared wall ratio, land use category, and neighboring building distance. This is a distinct advantage of analyzing layers separately, as opposed to the combined classification of Fleischmann et al. (2022). Layers can be analyzed with different methods that are most appropriate to that specific layer.

Street patterns are less easily distinguished, and boundaries are less clear. Section 3.2.1 describes existing methods, like classifications for specific countries Figure 8 or entire cities (Badhrudeen et al., 2022; Louf & Barthelemy, 2014), but the only classification that could be applied globally at the street segment or neighborhood level (Marshall, 2005) is unbalanced granularity wise and difficult to apply to real data. I propose to determine a new set of street pattern typologies instead of starting from a predefined list. To achieve this, unsupervised machine learning (clustering) is used in order to group street patterns together that are statistically similar. This is similar to the city-wide analyses of Badhrudeen et al. (2022) and Louf & Barthelemy (2014), but more granular and at element scale instead of city scale. It uses an adapter approach from Fleischmann et al. (2022), but only for the *minor roads* layer. Some examples of the attributes used in clustering are the curvilinearity of stretches of road, the distribution of street segment orientations, and the distance to neighboring roads. The result of this

clustering analysis is a set of unnamed clusters. The final step is to give each cluster a descriptive name, like a *gridded pattern* or *irregular curvy pattern*. This naming is done based on the statistical character of the attributes of each cluster.

The analysis phase results in the characterization of individual elements for every layer. From this result, areas of similar urban form start to become visible, see Figure 15, because the characterization of each element takes the neighborhood into account, following the approach from Fleischmann et al. (2022).

Figure 15 - Example result of building classification, showing the emergence of areas of different typology. (Building footprint data from OpenStreetMap contributors (2017), Overture Maps (2023))

4.1.2 Encode

This interpretation is formalized in the form of the *typology grid*, one grid for every (ambiguous) layer in the city stack. This grid is a gridded planar partition of the city domain encoding a single typology for every 100x100 meter cell. The typology of a cell is determined by the most common topology within that cell, determined by the sum of the intersection area of the elements within that cell of each typology. This categorical grid approach is not only comprehensible and easy to apply in the analysis phase, but it is also adapted from landscape analysis statistics, offering a wide range of statistical metrics to describe the resulting patterns (McGarigal & Marks, 1995).

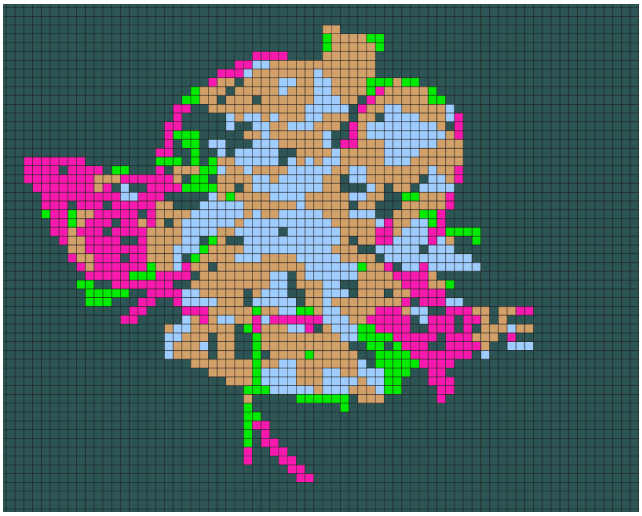


Figure 16 - Simple example of a typology grid, where each color signifies a different typology.

The typology grid forms a way to encode the distribution of typologies within a city, thereby describing the city. Besides the city, the typologies themselves also have to be encoded.

The typologies are universal across all cities worldwide and are therefore encoded separately from the cities. Each typology is statistically described based on an aggregation of its numerical and categorical attributes. For example, the description of a gridded street pattern typology could include a typical range of intersection angles (supposedly centered around orthogonal angles) and a regular distribution of distances between intersections.

This interpretation of the city is an oversimplification of the real structure of urban tissue. Condensing the form of cities worldwide into a small set of worldwide typologies is helpful for understandibility but doesn't accurately capture the diversity and complexity of cities.

To address the variety of urban tissue within a single typology, a finer level of detail is added. The *parameters* of a typology, with a city-level, area-level, or cell-level granularity. A simple example is density; areas within a "detached housing" typology can range from being almost wall-to-wall to being figuratively miles apart. More complex examples could be the aspect ratio of blocks within a gridded street pattern, ranging from square blocks to extremely elongated blocks. The resulting encoding of a single cell in the typology grid could, for example, be.

High density **grid pattern** with consistently sized mostly square blocks and mostly cardinal direction.

Where the bold text is the typology, and each color represents a different parameter. In practice, the parameters are encoded numerically, for example, a *mean block squareness* attribute with a value of 0.82.

This approach remains a simplified representation but aims to strike a balance between encoding detailed information and making the methodology and results comprehensible.

4.1.3 Generate

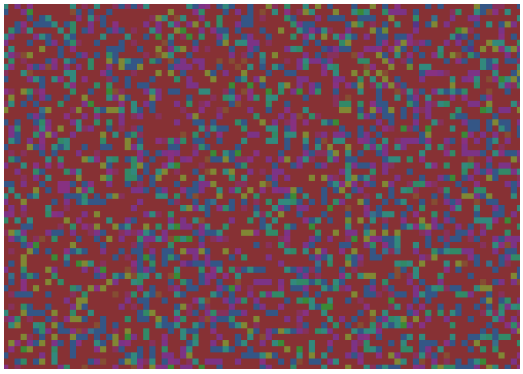
Generation of new cities uses the city stack as a framework. The heuristic is that layers lower in the city stack form constraints for layers higher in the stack (see Figure 1). Therefore, the city is generated layer by layer, from bottom to top. For each new layer, the layers below it serve as input constraints for the generation step.

In concept, different layers can use completely different generation methodologies. The layers implemented for this thesis follow a similar heuristic aimed specifically at generation ambiguous city stack layers.

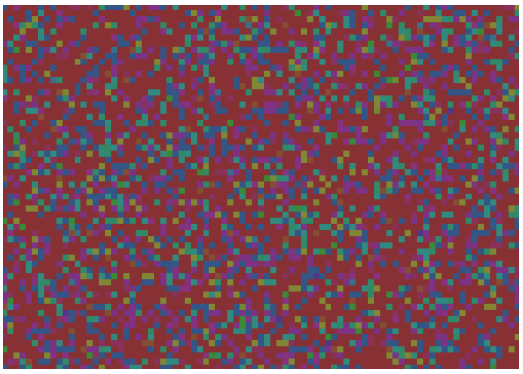
Step 1: Generate a new typology grid for this layer by adapting the grid from the template city to the constraints of the new city's lower-level layers while retaining the original grid's similar character.

Step 2: Generate individual layer elements following a predefined algorithm that takes the local value of the typology grid (including parameters) and surrounding elements of the lower-level layers as inputs.

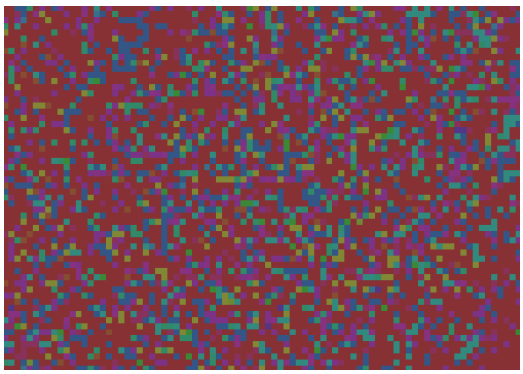
Multiple different methods were considered for generating the typology grid in a new context, including classification, optimization, and deep learning approaches. The method simulated annealing optimization technique Section 2.3 was chosen because of its stochastic nature while offering a lot of control. This approximates a globally optimal typology grid that is both similar to the typology grid from the template grid (by utilizing the aforementioned landscape statistic metrics) and conforms to the constraints of the lower-level layers of the new city. See Figure 17, for an example of the simulated annealing process of the typology grid.



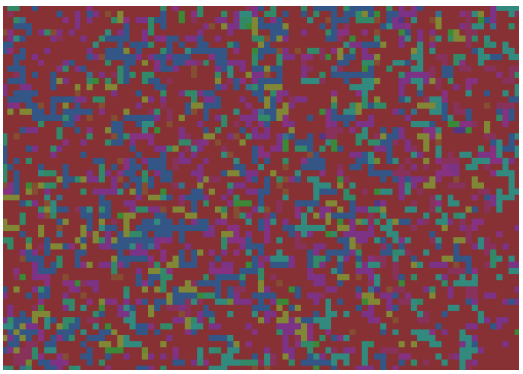
(a) Simulated annealing at 0%.



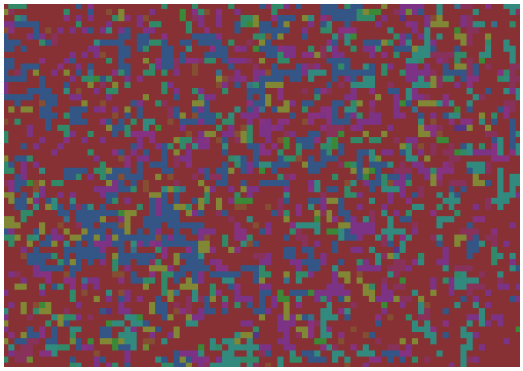
(b) Simulated annealing at 20%.



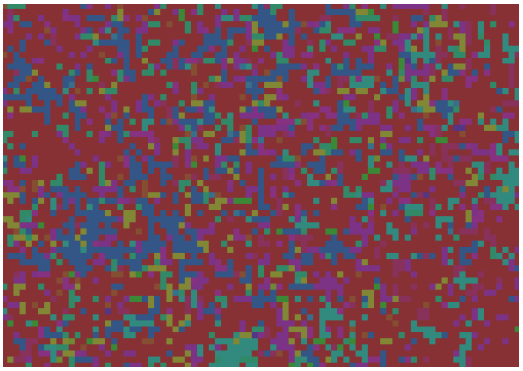
(c) Simulated annealing at 40%.



(d) Simulated annealing at 60%.



(e) Simulated annealing at 80%.



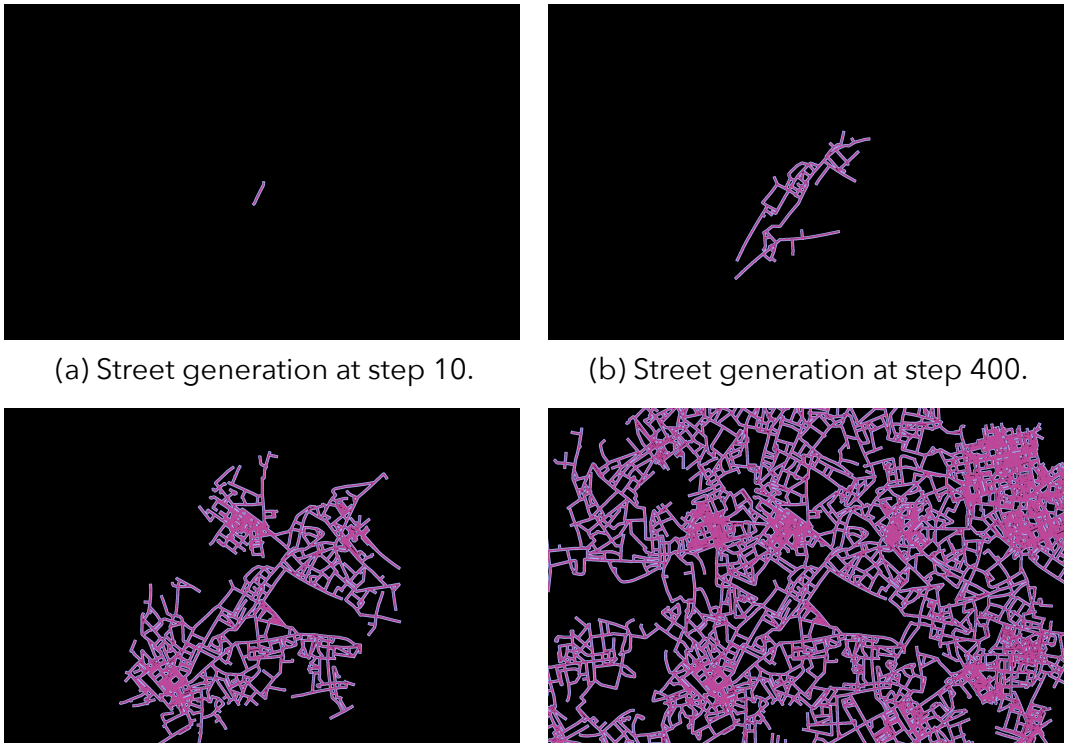
(f) Simulated annealing at 100%.

Figure 17 - Simulated annealing process.

Generating individual elements for the street network uses the same method for each of the three road system layers. The L-System approach from Parish & Müller (2001) is adapted to use the characteristics of the local typology on the typology grid as input parameters. The basic idea is:

1 **Start from a single segment query in a queue**
2 **While there are segments queries in the queue**

3 Adapt the segment query to local constraints like terrain, water, and existing segments
4 If the segment is not possible to be built, **continue**
5 Add the new segment to the finalized road network
6 Determine if and how many new segment queries to add at the end point of the new segment. The probability is based on the statistics of the local typology from the typology grid. 0 is a dead end, 1 is a continuing road, 3 is a T-intersection and 4 is an X-intersection.
7 Add the new segment queries to the queue. Set the length and angle based on the statistics of the local typology from the typology grid.
8 **End**



(a) Street generation at step 10.
(b) Street generation at step 400.
(c) Street generation at step 4000.
(d) Street generation at step 10000.

Figure 18 - Example of the street generation process.

Within the buildings layer, different methodologies are used for different typologies. Each method runs per individual block from the three combined street system layers (only minor roads are implemented in this thesis). The methods used are described in Section 5.5.

Finally, all layer elements are combined in a single 3D city model output file.

4.2 Analysis

This section provides a detailed explanation of the Analyze step, as outlined in Section 4.1.1.

4.2.1 Data Download

The first step of the pipeline involves acquiring geographic data for any city worldwide for every layer. A set of cities was selected to have a diverse global dataset. See Figure 19. Due to time constraints, not all selected cities were analyzed, resulting in a final set of 43 cities.

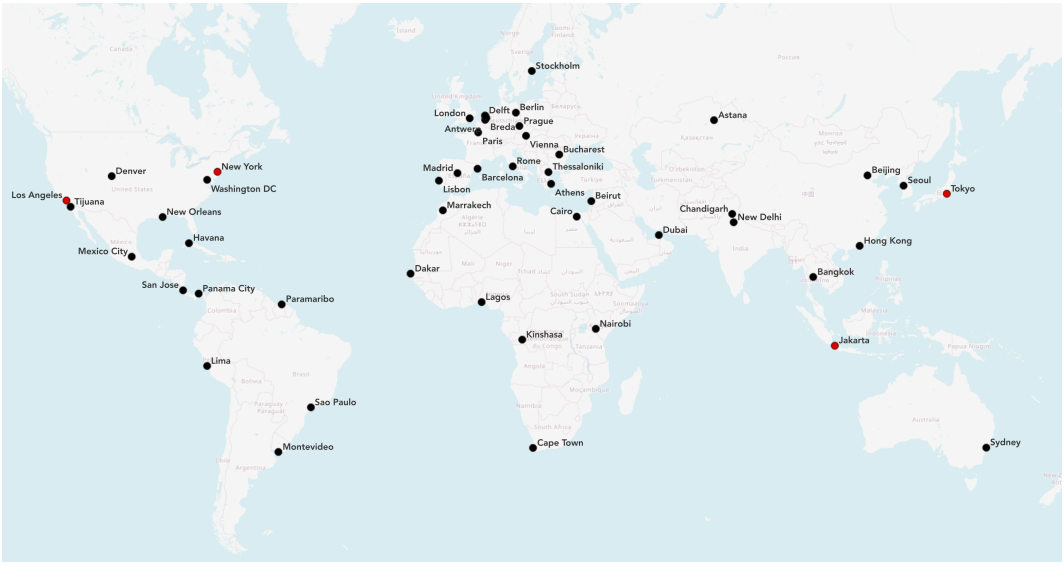


Figure 19 - Selected cities for the global dataset. Cities in red were canceled due to computation time. Rotterdam is also included, but obstructed by other labels. (Background map from OpenStreetMap contributors (2017))

The natural context layer involves terrain and water. For terrain height, the Copernicus GLO-90 Digital Elevation Model (European Space Agency & Airbus, 2022) is used, which gives global DSM raster coverage at a resolution of 90 meters. This data source was selected due to its free license, easy availability via code, and sufficient resolution. The data is stored as points - the centroid for every cell- and the only included attribute is height data.

For water, vector data is sourced from the volunteered geographic information source OpenStreetMap in the form of water areas and coastline features (OpenStreetMap contributors, 2017). This data source was similarly selected because of its permissible license, easy availability via code, global coverage and completeness, and sufficient data quality. The data is stored as polygons and line strings (to indicate coastlines), and it includes an attribute to distinguish between different types of water bodies, i.e., coastline, lake, river, and so on.

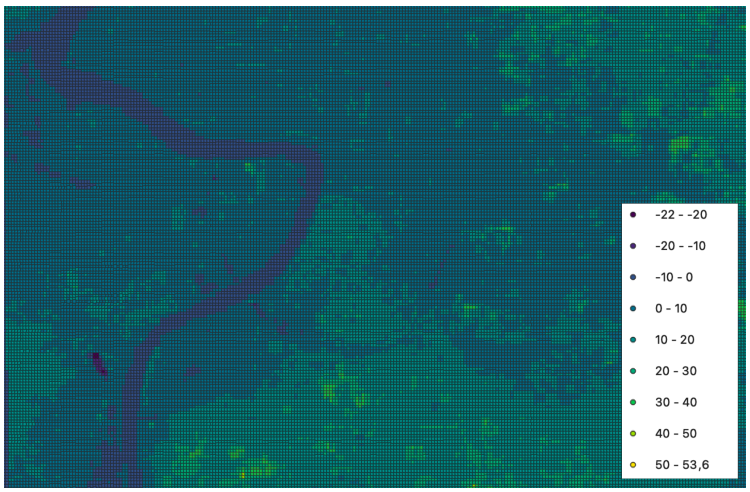


Figure 20 - Example of terrain height data for (a part of) the city of Antwerp, Belgium (Data from European Space Agency & Airbus (2022))

The produced dataset contains natural context data for future research, but I did not use this data for any analyses in this thesis.

The road system layer involves road segments and their associated metadata. All data for this layer was sourced from OpenStreetMap for the same reasons as indicated in the previous paragraph. The raw data includes a high level of granularity in the type of the road, for example distinguishing between 5 levels of

Figure 21 - Example of water data, in this case from the city of Antwerp, Belgium, with different types of water bodies narrowed down to a few categories. (Source data from OpenStreetMap contributors (2017))

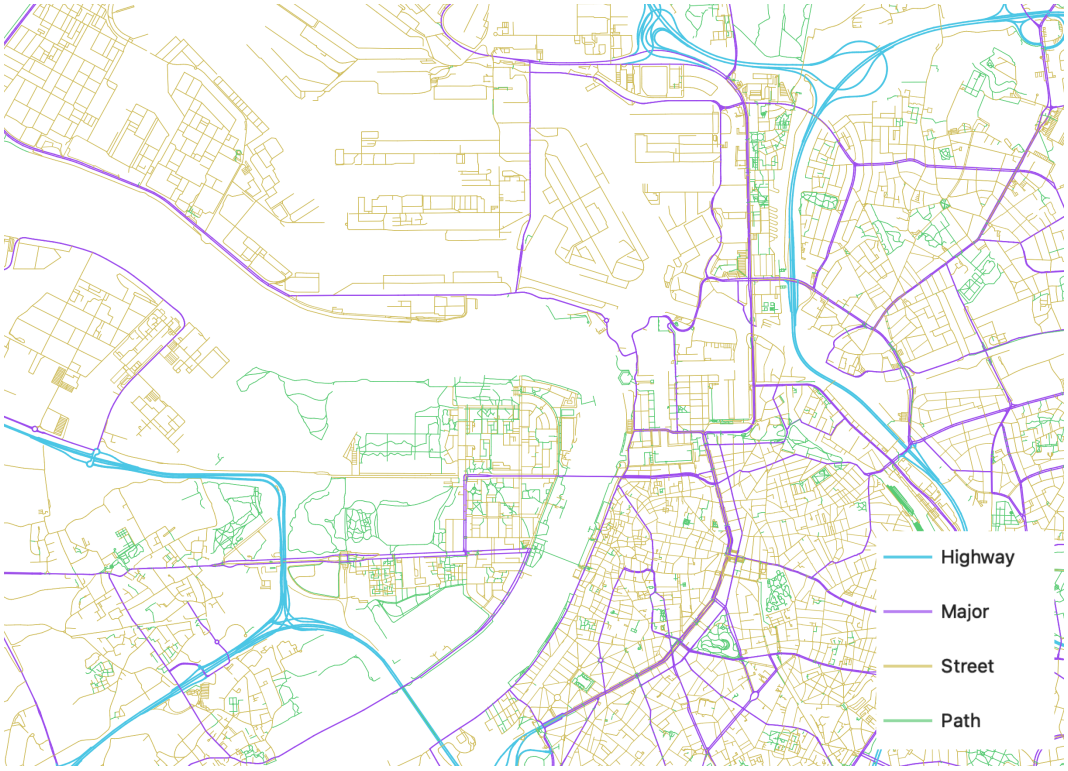


Figure 22 - The road system layer (Antwerp, Belgium) separated into 4 distinct levels (Source data from OpenStreetMap contributors (2017))

main road, several types of special roads, and many possibilities for small streets and paths. For this thesis, this typology has been simplified into 4 levels: highway, major road, street, and path. Besides the street typology, the attributes include segment length, width, tunnel true/false, bridge true/false, and intersection type (i.e. roundabout).

The building layer involves building footprints and their associated metadata. OpenStreetMap building footprints spatial coverage is uneven (Zhou et al., 2022), with over 75% of cities having data completeness below 20% (Herfort et al., 2023). Therefore, the data completeness is not sufficient for proper urban tissue analysis. The Overture Maps Foundation offers a dataset that combines building footprints from OpenStreetMap (OpenStreetMap contributors, 2017), Google Open Buildings (Sirko et al., 2021), Esri Community Maps (Esri, n.d.), and Microsoft Open Building Footprints (Microsoft, 2024). Together, the dataset includes 2.4 billion footprints (Maps, 2023), up from the 600 million in OpenStreetMap (OpenStreetMap Wiki, n.d.). The dataset especially offers much higher coverage in the Global South. The main downside is lower data quality due to many of the footprints being generated through artificial intelligence. This data source was chosen due to good coverage, free license, availability through code, and okay data quality.

Figure 23 - Example of building footprint data, with different colors showing the highest quality type of height data that is available for that footprint. (Source data from OpenStreetMap contributors (2017), Overture Maps (2023))

However, both OpenStreetMaps and Overture are missing building height data and level data for most buildings. To complement the available data, approximate height values from the GHSL Built-H dataset (Pesaresi, 2023) are also added as an attribute. This dataset offers a global approximation of building heights at a resolution of 100 meters. The value of the raster cell containing this building is added as an attribute to the building.

The final attributes for each building footprint include real height, real levels count, GHSL height, building roof shape, building name, building class, building subtype, original data source, and machine learning confidence score.

Additional supporting layers are also downloaded, namely land use data (OpenStreetMap contributors, 2017), city extents (OpenStreetMap contributors, 2017), and approximate city center location (OSM Contributors, n.d.). Land-use has been simplfied into 6 categories (airport, commercial, industrial, residential, retail, and unkown). This categorisation combines both the airport layer and land-use layer from OpenStreetMap contributors (2017).

All layers are bundled together in a single GeoPackage data file.

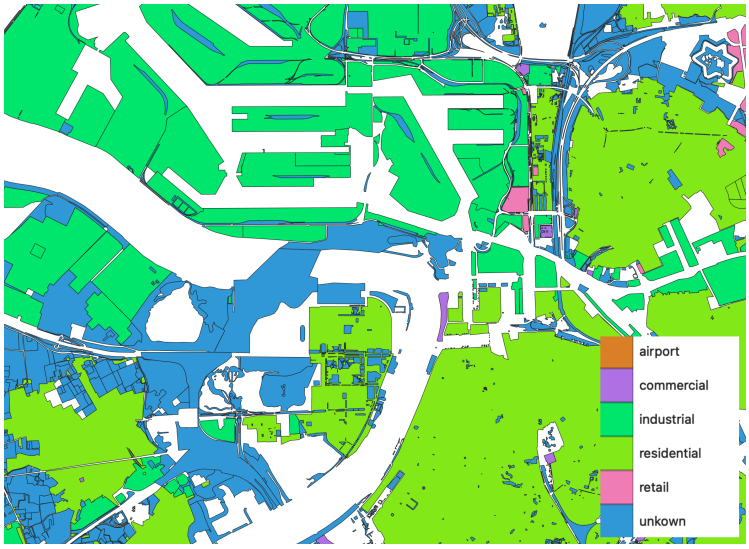


Figure 24 - Example of simplified land use data for a part of the city of Antwerp, Belgium. (Source data from OpenStreetMap contributors (2017))

4.2.2 Metric Computation

The second step of the pipeline involves computing a set of metrics for the aforementioned road system and buildings layer. This section gives an overview of the computed metrics for both of these layers. Additionally, new supporting layers are generated that are necessary either for the computation of the metrics or for the neighborhood aggregation step (Section 4.2.3).

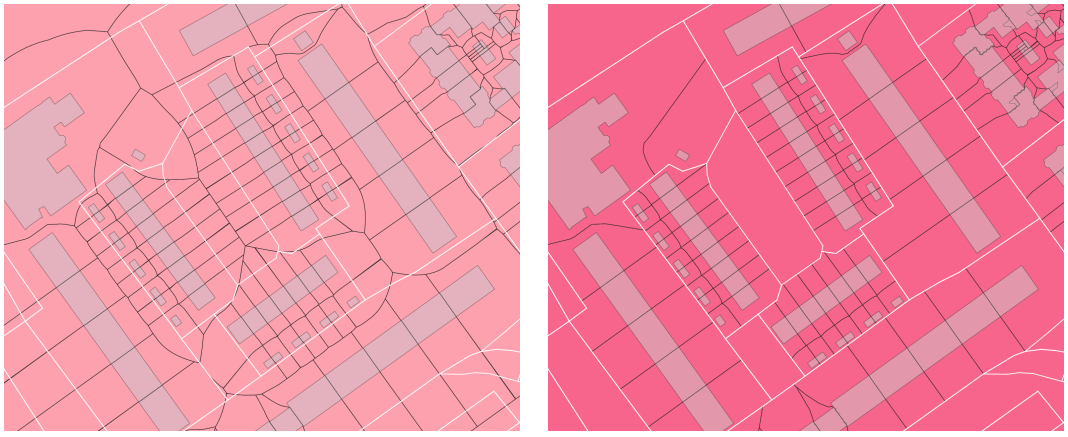
SUPPORTING LAYERS GENERATION

Two additional layers are generated, the enclosures layer and the tesselation layer.

Enclosures are defined as areas enclosed on all sides by the road system. This includes both big areas like city blocks and small areas like the middle of a roundabout. The enclosures layer has a separate polygon for every enclosure and has no additional attributes besides a unique ID.

Figure 25 - Example of generated enclosure polygons.

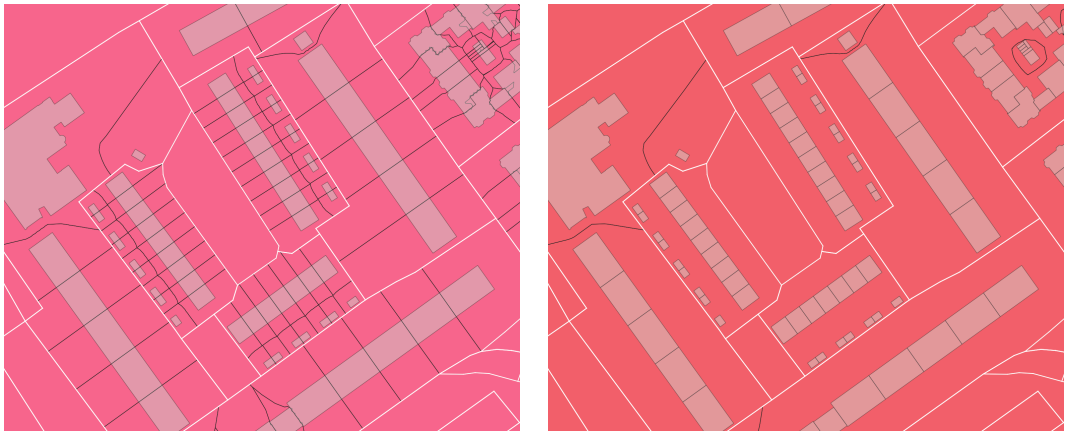
The morphological tesselation is an approximation of building plots. It partitions the space by creating a Voronoi tesselation of the building footprints (Fleischmann et al., 2020). An enclosed morphological tesselation builds on this by constraining the tesselation to the aforementioned enclosures. This also significantly improves performance and reduces memory usage.



(a) Morphological tesselation. (b) Enclosed tesselation.

Figure 26 - Examples of different types of tesselation.

In this thesis, the enclosed morphological tesselation is generated based on building groups, not on individual buildings. Essentially, a building group is a collection of buildings that are connected; see Section A. This simplifies the processing of the enclosures, and also groups together buildings that naturally belong together so they can be used for building generation method 1; see Section 5.5.



(a) Based on individual buildings. (b) Based on building groups.

Figure 27 - Examples of different ways of generating the enclosed tesselation.

ROAD NODE METRICS

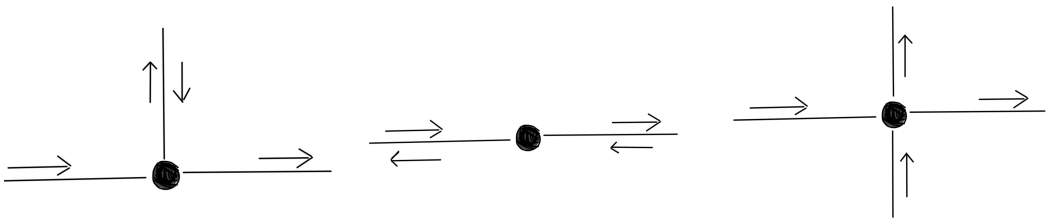
Metrics are calculated separately for road nodes and for road segments. Appendix A contains a table of all the computed metrics for road nodes, including references to the sources they were based on.

A noteworthy change I made to an existing node metric is described in Section A.

A NODE DEGREE & INTERSECTIONS

The node degree in a graph is defined as the number of edges connected to a node. More specific versions include the in-degree as the number of in-edges and the out-degree as the number of out-edges in a directed graph.

This poses a challenge for characterizing street networks since the urban form of the three intersections in Figure 28 is different, but the degree, in degree, and out-degree, are the same.



(a)	(b)	(c)
Degree: 4	Degree: 4	Degree: 4
In degree: 2	In degree: 2	In degree: 2
Out degree: 2	Out degree: 2	Out degree: 2
$\text{degree}_{\text{street}}$: 3	$\text{degree}_{\text{street}}$: 2	$\text{degree}_{\text{street}}$: 4
intersection: true	intersection: false	intersection: true

Figure 28 - Three different intersections with the same node degree.

Therefore, a redefined version of node degree was developed as the count of unique nodes v that have an edge to u and/or an edge from u. This functions the same as the node degree on an undirected graph, but then on a directed graph. This can be mathematically defined as:

The street network is seen as a directed graph, where a one-way street is an edge in one direction and a two-way street is an edge in both directions.

Let $G = (V, E)$ be a directed graph where V is the set of nodes (vertices) and E is the set of edges. For a node $u \in V$, define the connected set C as:

$$C(u) = \{v \in V \mid \text{there exists a directed edge from } u \text{ to } v \text{ or a directed edge from } v \text{ to } u\} \quad 1.$$

The degree of streets can then be defined as the cardinality of the connected set C.

$$\text{degree}_{\text{streets}}(u) = |C(u)| \quad 2.$$

Subsequently, any node with a street degree higher than 2 can be considered an intersection, and any node with a degree of 1 as a dead end.

$$\text{intersection} = \begin{cases} \text{True} & \text{if degree}_{\text{streets}} > 2 \\ \text{False} & \text{otherwise} \end{cases} \quad 3.$$

$$\text{dead end} = \begin{cases} \text{True} & \text{if degree}_{\text{streets}} = 1 \\ \text{False} & \text{otherwise} \end{cases} \quad 4.$$

Road Segment Metrics

Table Appendix B contains an overview of all metrics for road segments, including whether they are new or based on existing sources. Noteworthy newly introduced metrics are described in Section A through Section F.

A Next Segment & Forward Angle

This metric aims to select the next segment in the forward direction for each road segment in the road system. The next segment is chosen as follows.

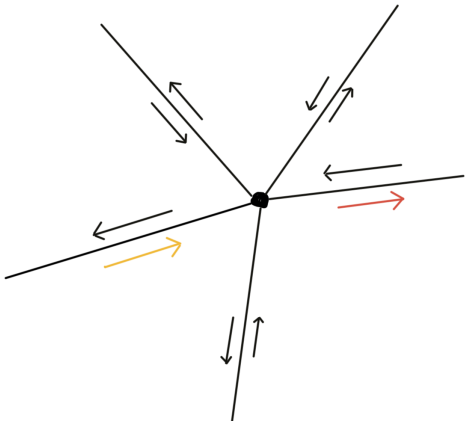
Start from current segment (u, v)
Calculate the bearing of segment (u, v)
Create a list of all outgoing segments from node v, excluding (v, u)
For every outgoing segment (v, w) do
    Calculate the bearing of segment (v, w)
    Determine the difference between the bearing of (u, v) and (v, w)
    Normalise the difference between −180 and 180 degrees
End
If there are no outgoing segments
    Next segment is None
Else if there is one outgoing segment
    Next segment is the single outgoing segment
Else if there are multiple outgoing segments
    Next segment is the outgoing segment with the smallest angle difference, **if** the smallest angle difference is greater than 45°, otherwise None
End

This means that in road sections without intersections, the next segment does not have an angle limit (since it belongs to the same road). After an intersection, there is a limit of 45°, since that would most likely constitute a T-intersection instead of a continuing road. See Figure 29 for examples.

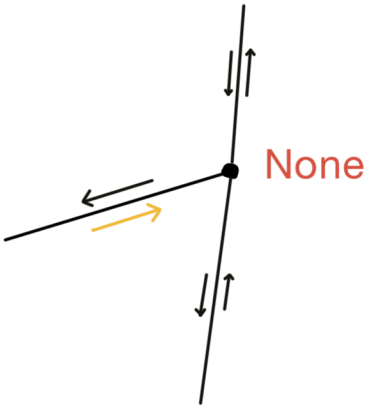
The (u, v, key) identifier of the next segment is added as the `next_segment` attribute to the current segment, and the identifier of the current segment is added as the `previous_segment` attribute to the next segment. This allows for traversal in both directions and is a prerequisite for the computation of other metrics.

The angle difference between the current segment and the next segment is saved in degrees as the `forward_angle` attribute.

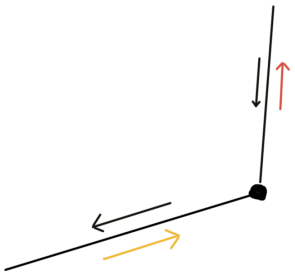
(a) Next segment is only available forward segment.



(b) Next segment is available forward segment with smallest angle difference.



(c) No next segment, as there are multiple available forward segments, but none below $45°$.



(d) Next segment is only available forward segment, even though the angle is above $45°$.

Figure 29 - Identified next segment in different situations with current segment in yellow and next segment in red.

This can be formally defined as follows.

Given two consecutive road segments:

1. (u, v) – the current segment.
2. (v, w) – a potential next segment.

Let the coordinates of u, v, and w be $(x[u], y[u]), (x[v], y[v]), (x[w], y[w])$, and the angle of a segment (a, b) be defined with Equation 5.

$$\theta(a, b) = \text{atan2}(y[b] - y[a], x[b] - x[a]) \quad 5.$$

Then the angle difference between (u, v) and (v, w) can be defined with Equation 6

$$\Delta\theta(u, v, w) = ((\theta(v, w) - \theta(u, v)) + \pi)\%(2 * \pi) - \pi \quad 6.$$

Finally, the next segment can determined using Equation 7

$$\text{Next segment} = \begin{cases} (v, w) & \text{if } \text{len}((v, w)\text{'s}) = 1 \\ (v, w) \text{ where } \min(|\Delta\theta(u, v, w)|) & \text{if } |\Delta\theta(u, v, w)| \leq \frac{\pi}{4} \\ \text{None} & \text{otherwise} \end{cases} \quad 7.$$

B ROAD SECTION

A road section is defined as a set of consecutive road segments between two intersections. What constitutes an intersection is defined in Section A. For two-way roads, the road section includes the segments in only one direction.

Each section is assigned a unique ID, which is saved in the `section id` attribute of every segment belonging to the section.

The section ID is used as a prerequisite for metrics that measure something for the whole section.

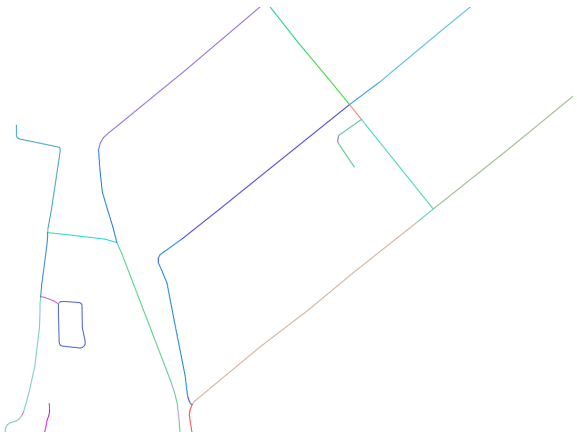


Figure 30 - Example of road section ids (each color signifies a different id). Note: Sometimes, the color of the underlying road in a different direction clips through. (Source data from OpenStreetMap contributors (2017))

C Road Stretches & Continuity

Continuity aims to capture how much you can keep driving straight on the same road. To calculate this, we first must define the *road stretch*. A road stretch aims to capture a stretch of continuous road segments up to a certain length limit. This limit is imposed for comparability reasons since the length of a road stretch is potentially infinitely big.

The stretch is computed individually for each segment. It starts from the current segment and recursively grows in forward and backward directions. If the angle difference between the segments is above a certain threshold, the stretch does not grow further in that direction. If the maximum length is reached, the stretch also stops growing. See Figure 31 for examples.

Figure 31 - Examples of road stretches. A terminates on the right side because there is no forward segment and on the left side because the forward angle is higher than the limit. B and C terminate on both sides because there is no forward segment. D terminates because the maximum length of 500 meters is reached. (Source data from OpenStreetMap contributors (2017))

The stretch can formally be defined as follows.

Let the initial segment be S_0. The road stretch R is defined as the largest set of contiguous segments $\{S_0, S_1, ..., S_n\}$ such that:

1. For each pair of consecutive segments $S_i, S_{i+1} \in R$, the angle difference satisfies:

$$|\theta_{i+1} - \theta_i| \leq 25° \tag{8.}$$

2. The total length of the stretch does not exceed 500 meters:

$$\sum_{i=0}^{n} L_i \leq 500, m \tag{9.}$$

Continuity is defined as the ratio between the length of the road stretch and the maximum possible length of a road stretch (the threshold distance). Values approaching 0 indicate very low continuity, where this road segment does not belong to a longer stretch of segments without having to take a significant turn. Values near 1 indicate high continuity, where this segment belongs to a stretch of road that is not interrupted by turns or T-intersections.

Formally, continuity can be defined as.

$$\text{continuity} = \frac{L}{L_{\max}} \tag{10.}$$

where L denotes the length of the stretch to which the current segment belongs, and $L_{\max}$ is the maximum length of a stretch, in this case 500 m.

D STRETCH LINEARITY & CURVILINEARITY

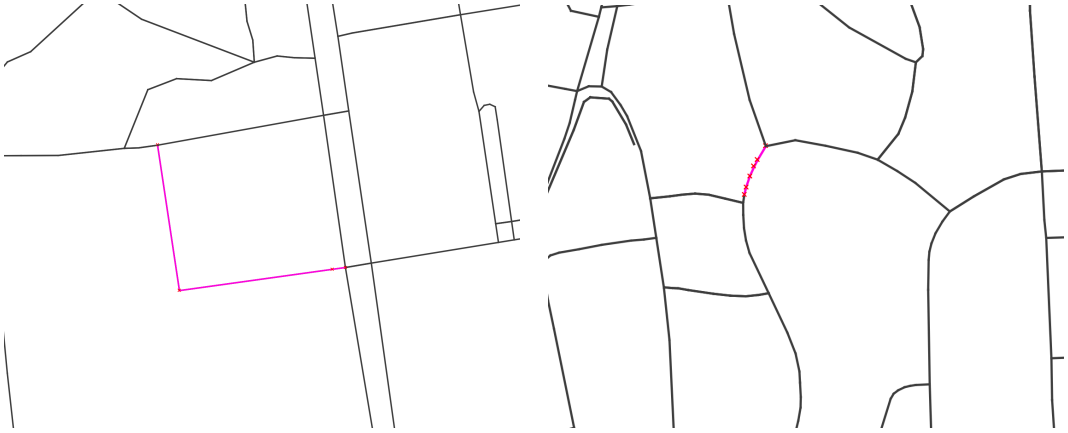
Stretch Linearity and Stretch Curvilinearity aim to capture the straightness and curviness of the roads of the road system.

Momepy (Fleischmann & PySAL Developers, 2018) includes a definition of linearity, adapted from Araldi & Fusco (2019). Here, linearity is defined as the ratio between the shortest distance between the start and end of a line string, and the total length of the line string.

$$\text{linearity} = \frac{l_{\text{euclidian}}}{l_{\text{segment}}} \tag{11.}$$

Note that what Momepy refers to as a segment is equivalent to what this thesis refers to as a road section (since segments would always have a linearity of 1.0).

Linearity can be used to determine the straightness of a segment. The curvilinearity could then be seen to have an inverse relationship with the linearity. Low linearity would then signify high curvilinearity. Using this approach, however, with the linearity of road sections, has some problems. Example Figure 32a will have low linearity. One would, however, not classify this as a "curvy" street. Example Figure 32b will have high linearity since the section is almost straight, but one would instinctively see these segments as part of a "curvy" road.



(a) Example of "linear" streets with low linearity (and therefore high curvilinearity).

(b) Example of "curvy" streets with high linearity (and therefore low curvilinearity).

Figure 32 - Examples showing the problem of using section linearity as a measure for curviness (Source data from OpenStreetMap contributors (2017))

To combat both of these issues, this thesis proposes an alternative to road section linearity, namely road stretch linearity. Instead of calculating the linearity of the road section bounded by intersections, the linearity is calculated on the road stretch as defined in Section C. The angle threshold for road stretches is set to a higher value of 40° to account for the possibility of sharp turns in curvy roads. Because road stretches don't stop at intersections, but do stop at sharp turns, both issues from image … are addressed.

The final definition would then be an adaptation of Equation 11.

$$\text{linearity}_{\text{stretch}} = \frac{l_{\text{euclidian}}}{l_{\text{stretch}}} \tag{12.}$$

Stretch curvilinearity can then be seen as an inversion of stretch linearity, see Equation 13.

$$\text{curvilinearity}_{\text{stretch}} = 1 - \text{linearity}_{\text{stretch}} \tag{13.}$$

One issue remains with this approach. Curvilinear stretches with low curvature have linearity values approaching 1.0. Therefore, they would be assigned a curvilinearity value approaching 0. For proper clustering performance, more separation is needed. This is addressed by using a logarithmic approach for the linearity equation (see Equation 14). The linearity has to be capped at $0,9999$ to prevent infinite values or NaN values because floating point imprecision can lead to values above 1.0.

$$\begin{aligned} \text{linearity}_{\text{stretch:simple}} &= \frac{l_{\text{euclidean}}}{l_{\text{stretch}}} \\ \text{linearity}_{\text{stretch:capped}} &= \min\big(\text{linearity}_{\text{stretch:simple}}, 0.9999\big) \\ \text{linearity}_{\text{stretch:log}} &= \frac{\log\big(1 - \text{linearity}_{\text{stretch:capped}}\big)}{\log(1 - 0,9999)} \end{aligned} \tag{14.}$$

The final stretch curvilinearity can be seen in Figure 33.

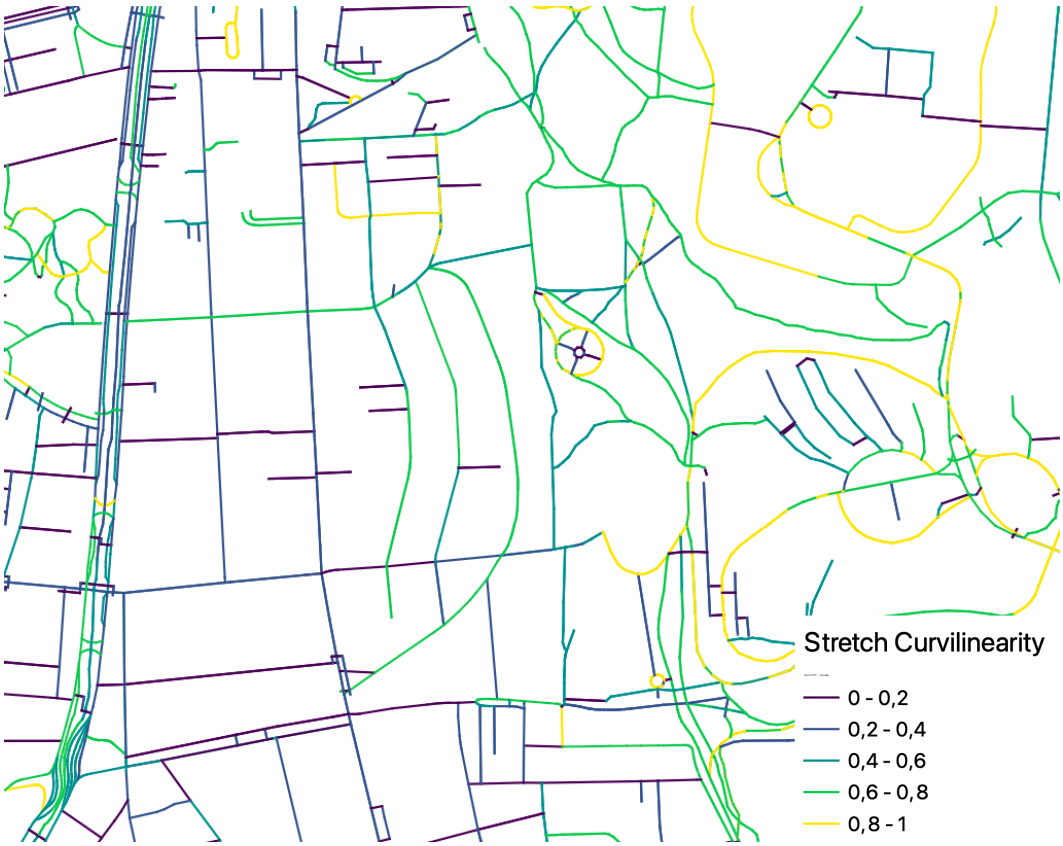


Figure 33 - Example of stretch curvilinearity. (Source data from OpenStreetMap contributors (2017))

E Intersection Left Angle

The intersection left segment is the first occured segment when rotating clockwise from the current segment, except if this segment is the next segment, see Section A.

The angle difference is defined using the aforementioned Equation 6 from Section A.

The algorithm works as follows.

1 Start from segment (u, v)
2 Calculate bearing of segment (u, v)
3 **For every adjacent segment (v,w) do**
4 | Calculate bearing of segment (v, w)
5 | Calculate angle difference between (u, v) and (v, w)
6 | Normalise angle difference between −180° and +180°
7 **End**

If v is not an intersection
| *Intersection left segment* is None
Else
| *Intersection left segment* is the segment with the lowest angle difference.
| **If *Intersection left segmemt* is the same as the forward segment**
| | *Intersection left segment* is None
| **End**
End

See Figure 34 for an example of the final values.

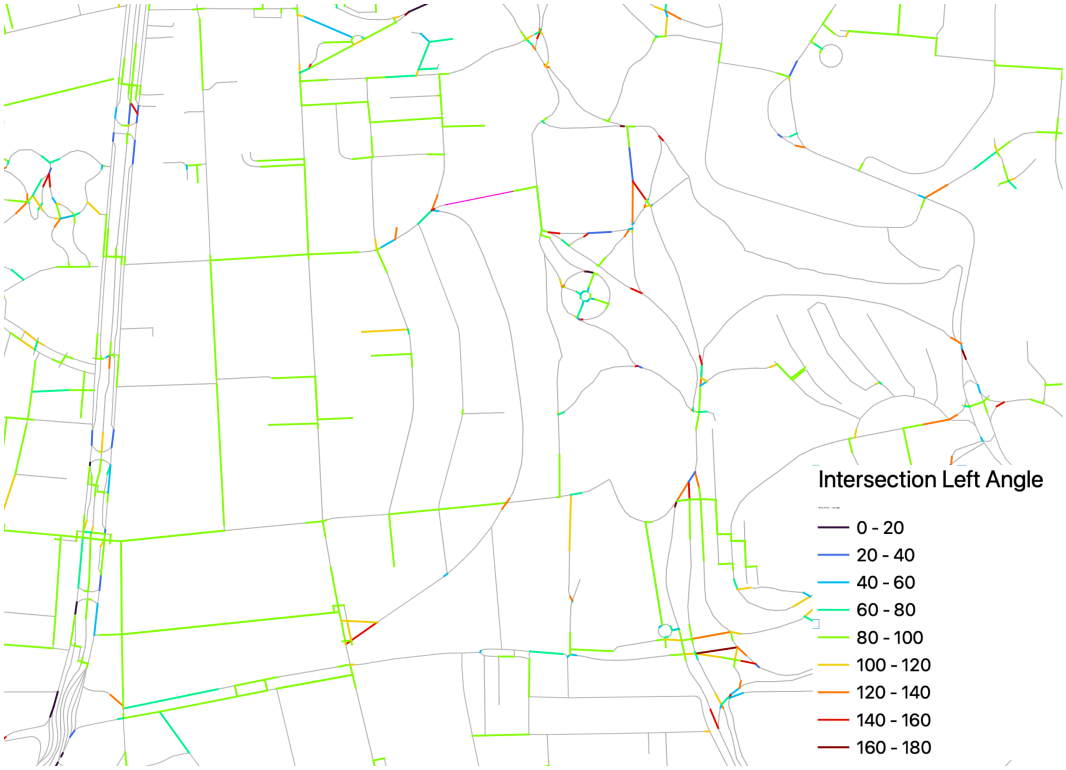


Figure 34 - Example of intersection left angle values. (Source data from OpenStreetMap contributors (2017))

F RIGHT NEIGHBOUR METRICS

To capture relationships between adjacent street segments, a method had to be developed to determine the neighbouring street segment. In concept, the neighboring segment is the first segment encountered when looking to the right of the current segment. Similarly for major roads, it is the first major (or highway) segment encountered to the right.

The algorithm works as follows.

1 Start from current segment (u, v)
2 Get centroid c of segment as the point exactly in the middle of u and v
3 Create a unit vector perpendicular to segment (u, v) starting from centroid c
4 Scale the unit vector to a predefined threshold distance to get line l. 300 m is used for `right_neighbour` and 1000 m is used for `major_right_neighbour`.
5 Find all road segments intersecting with l, only consider segments with type category *major* or *highway* for `major_right_neighbour`.
6 The `right neighbour` is the intersecting segment with the smallest distance from c, except distances below a threshold of 5 m for `right_neighbour` and 20 m for `major right neighbour` to disregard parallel segments of the same road in the other direction.

In addition to a reference to the right neighbor, the distance to the segment and the angle deviation calculated with Equation 6 are saved as attributes.

Building Metrics

Appendix C shows an overview of all computed building metrics, including the sources they originate from.

Section A through Section C highlights noteworthy changes or new metrics.

A Building Group ID

Both enclosures and certain metrics operate on collections of buildings that are connected. The `building_group_id` attribute encodes these groups as an attribute. The algorithm works as follows.

1 Perform a "touches" (Egenhofer 9-IM predicate) inner spatial join on all buildings with itself
2 Remove rows between a building and itself
3 Create a graph where every row is an edge, linking one building to another
4 Identify the connected components
5 Give every component a unique ID, assign this ID to all objects within that component

B Building Group Enclosure Perimeter Coverage

This metric aims to capture what percentage of the enclosure's perimeter (the surrounding streets) are directly bordering this building group. This is defined as follows:

Let E be the enveloping enclosure of building group B. P denotes the perimeter linestring of E_{10}, inset by a constant value of 10 m.

$$\text{building group enclosure perimeter coverage} = \frac{\text{Length}(E_{10} \cap B)}{\text{Length}(E_{10})} \tag{15.}$$

C APPROXIMATE HEIGHT

As indicated in Section 4.2.1, height and level data are often missing. GHSL Built-H data (Pesaresi, 2023) is available, but since it's aggregated, it should be used as a last resort. The `approximate height` metric aims to assign the highest-quality height value available to every building. In case the real height is known from the source data, that height is assigned. In case the amount of levels is known, height is estimated by multiplying this count with a level height constant. If both height and levels are unknown, the GHSL height is used.

$$\text{height}_{\text{approximate}} = \begin{cases} \text{height}_{\text{real}} & \text{if height}_{\text{real}} \text{ is not None} \\ \text{levels}_{\text{real}} \times 3.0m & \text{if levels}_{\text{real}} \text{ is not None} \\ \text{height}_{\text{ghsl}} & \text{otherwise} \end{cases} \tag{16.}$$

4.2.3 Neighbourhood Aggregation

Categorizing features is very difficult without knowing their context. You can't know if a straight road next to an orthogonal intersection is part of a strict grid without zooming out and looking at its neighborhood. This is where the contextualizing step comes in. It gathers the neighborhood for every element and saves aggregated metrics of that neighborhood as attributes on the element. This is done separately for the road system layer and the buildings layer.

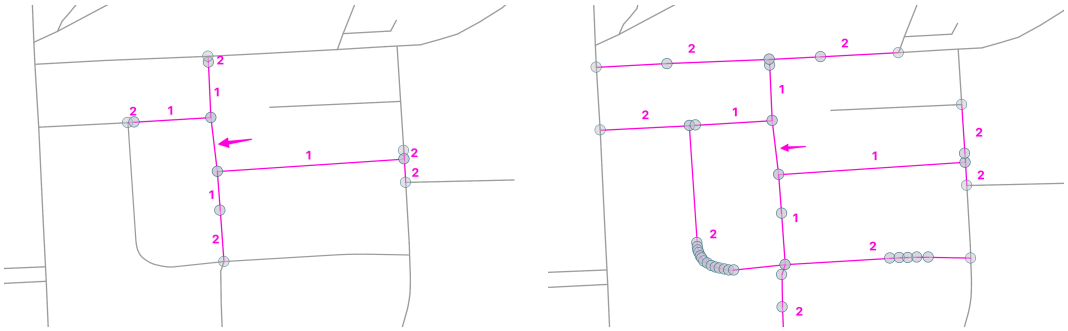
ROAD SYSTEM CONTEXTUALIZING

Several implementations were considered, their description and pros and cons are listed in Appendix D.

All methods have pros and cons, but *Recursive network growing* (Starting from the current segment, recursively grow n levels deep in all possible directions in the road graph. It is constrained to not grow past major road intersections.) was chosen due to its not being scale sensitive and its high level of control while capturing a logical neighborhood.

For performance reasons, to get more consistently sized neighborhoods, and to counteract the negative effect of very short segments part of a longer (curved)

section, the recursive network growing is done on a graph of all road sections and not just road segments. See Figure 35 for a comparison.



(a) Based on road segments. (b) Based on road sections.

Figure 35 - Example showing the difference between segment and section-based network growing (both with $n = 2$). The arrow shows the starting segment, the numbers indicate the recursive growing step. (Source data from OpenStreetMap contributors (2017))

For road graph $G(N, S)$ with nodes N and sections s, the road section neighbourhood set S_n of all reachable sections from s_0 within n steps can be defined as follows:

$$S_n(s_0) = \{s \in S \mid s \text{ can be reached in } n \text{ steps from } s_0 \text{ and is of the same class or higher as } s_0 \text{ and does not require to cross a street of a higher class}\} \quad 17.$$

The simple aggregations performed for every road segment on its neighborhood are summarized in Table 1, and the custom aggregations in Table 2

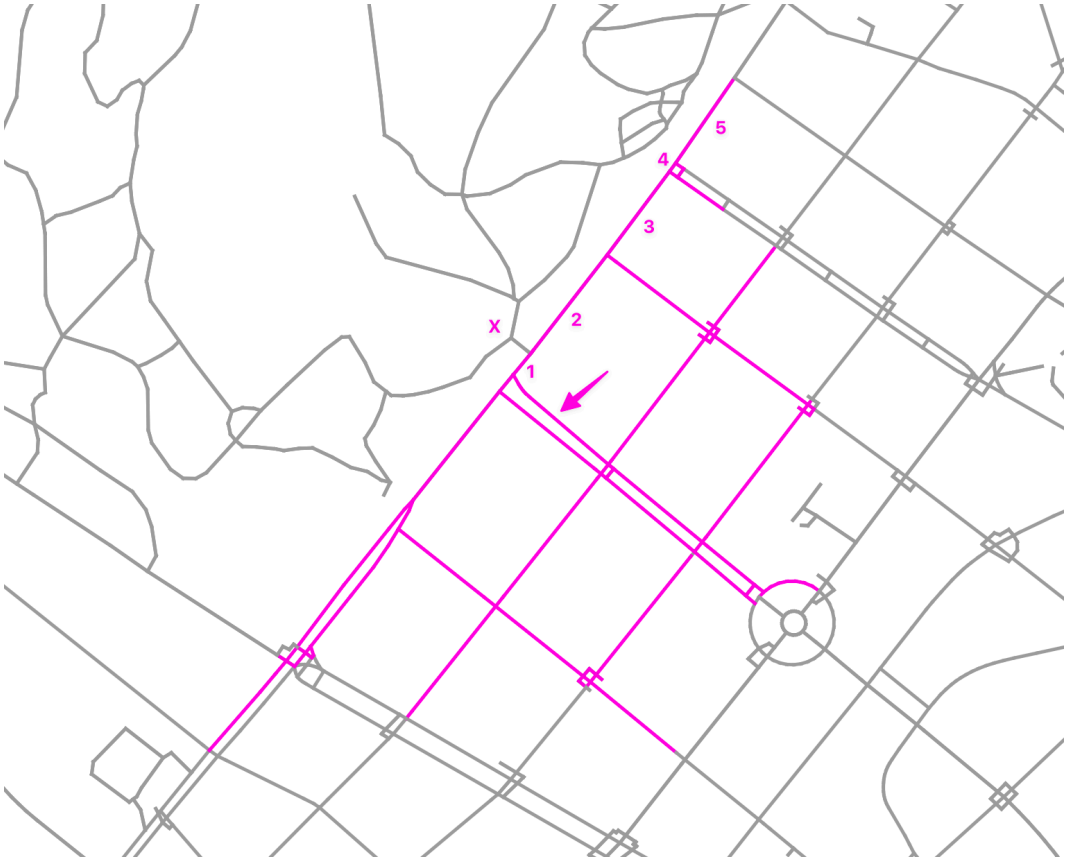
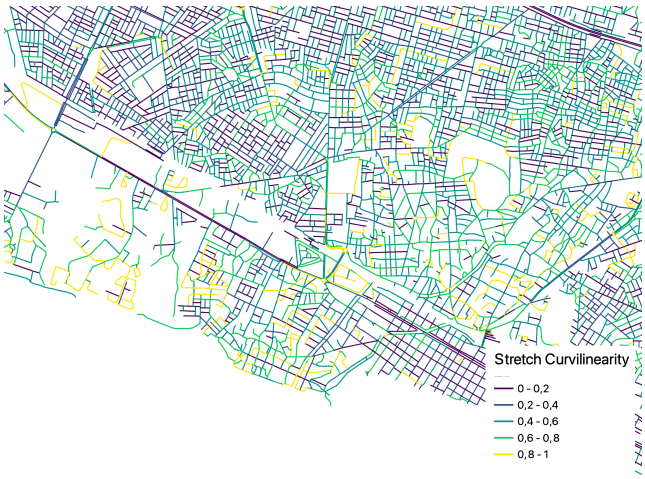


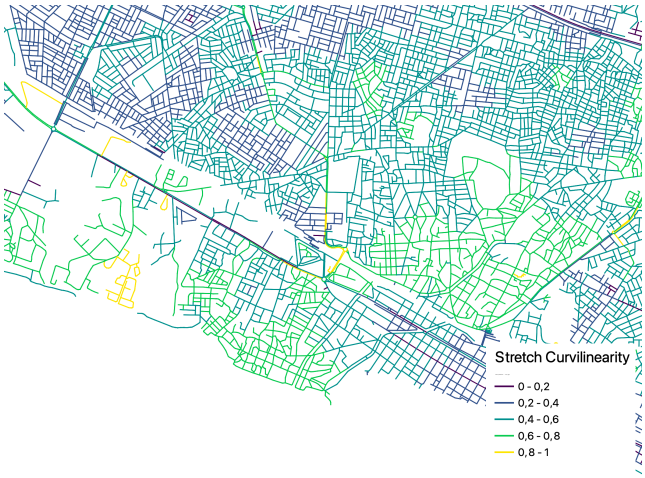
Figure 36 - Example of the neighborhood of a road section (marked with an arrow) based on network growing with $n = 5$. The contextualization does not grow to section X because they lie on the other side of a major road. (Source data from OpenStreetMap contributors (2017))

Metric name	Mean	Std
`stretch linearity`	X	X
`forward angle`	X	X
`continuity`	X	X
`stretch curvilinearity`	X	X
`intersection left angle`	X	X
`right neighbour angle deviation`	X	X
`right neighbour distance`	X	X
`forward angle` (absolute)		X
`section_length`	X	X

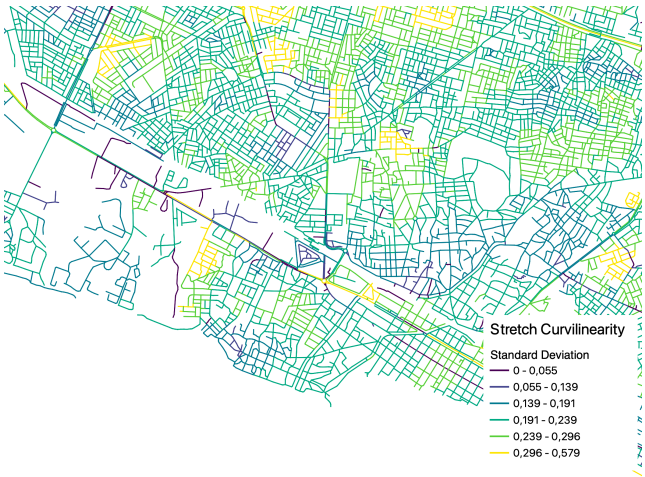
Table 1 - Road neighborhood simple aggregations.



(a) Without neighborhood aggregation.



(b) With 5 step neighborhood aggregation (mean).



(c) With 5 step neighborhood aggregation (standard deviation).

Figure 37 - Examples of the effects of neighbourhood aggregation. (Source data from OpenStreetMap contributors (2017))

Metric name	Description
`angle entropy`	See Section A
`bearing_cardinality`	See Section B
`dead_end_sections`	Percentage of dead-end road segments, weighted by segment length
`continuity`	Weighted median by segment length, to remove the difference between grid street pattern with and without small service streets within the grid
`stretch_curvilinearity`	Weighted mean by segment length
`right neighbour angle deviation`	Weighted mean by segment length
`right_neighbour_distance`	Weighted mean by segment length
`forward_angle_abs`	Weighted mean by segment length

Table 2 - Road neighborhood custom aggregations.

A ANGLE ENTROPY

This metric aims to capture the uniformity (or the lack thereof) of the angles within a street pattern. Grids would be examples of having high uniformity, and organic patterns would have low uniformity.

To capture the lack of uniformity in the street-bearing distribution, the method proposed by Boeing (2019) is used. In this method, the bearings of all segments within a city are divided into 36 bins (each 10°). The bins are shifted 5° so common bearings like 0°, 45°, 90°, etc. fall within the middle of a bin and slight offsets like 89.9 ° and 90.1° fall in the same bin. Finally, the Shannon entropy (Shannon, 1948) of the bearing distribution is computed.

B BEARING CARDINALITY

This metric aims to capture in how far the street orientations follow the cardinal directions (North, South, West, East). This is computed as the percentage of road segments within the neighborhood that fall within 5° of any of the cardinal directions 0°, 90°, 270°, and 360°.

BUILDINGS CONTEXTUALIZING

Building contextualizing uses the *Radius* method from Table 9. A building's neighbourhood is defined as all other buildings with their centroid within 300 meters of this building. A second contextualizing method selecting all buildings in the same enclosure was rejected due to problems handling arbitrarily big enclosures at the edges of the city.

Figure 38 - Example of angle entropy values (Source data from OpenStreetMap contributors (2017))

The following aggregations are performed on the neighborhood of every building, see Table 3.

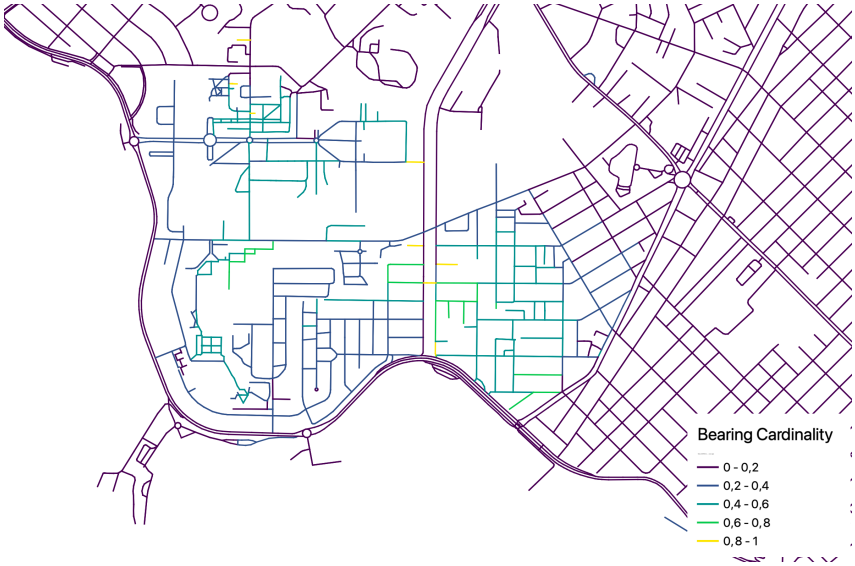


Figure 39 - Example of bearing cardinality values (Source data from OpenStreetMap contributors (2017))

Metric name	Mean	Std	Max	Mode
area	X	X		
building group enclosure perimeter coverage	X	X		
shared walls ratio	X	X		
shape index	X	X		
building group elongation std	X			
building group area std	X			
equivalent_rcetangular_index	X	X		
elongation	X	X		
covered_area	X	X		
alignment	X	X		
squareness	X	X		
approximate_height	X	X	X	
land_use_category				X

Table 3 - Building neighborhood aggregations.

4.2.4 Clustering & Classification

STREETS CLUSTERING

I used a Gaussian Mixture Model (GMM) as clustering model, which is a probabilistic derivative of the k-means clustering method (Reynolds, 2009). Fleischmann et al. (2022) identifies this model as the most suitable for this type of clustering methodology. Besides the better sensitivity to clusters of varying sizes and the method not only relying on squared Euclidian distances mentioned by Fleischmann et al. (2022), this method has other advantages for my specific research:

- It allows the application of clusters to unseen cities without needing the data originally used to create the clusters. The parameters of the model can be saved, and at a later moment, the model can be reconstructed from these parameters.
- Relatively low time and space complexity. Fitting the model to more than 100 million rows took just a few minutes, whereas initial tests with HDBSCAN (McInnes et al., 2017) and OPTICS (Ankerst et al., 1999) already took significant time on less than 100 thousand rows.
- The road pattern typology clusters are not well defined, so density-based methods are not well suited. GMM can create clusters in data without clear cluster edges.

Both the GMM method and clustering in general did however pose challenges in seeing the impact of each of the selected metrics on the final clustering

Feature selection was based on observing street patterns in cities worldwide. Different metrics were developed to capture the observed differences. Section A through Section F highlight the set of metrics selected to establish a minimal set to distinguish the vast variety of street patterns worldwide.

A NEIGHBOURHOOD CURVILINEARITY

The metric called `stretch_curvilinearity::nb::weighted_mean` (in the format `<metric>::neighborhood::<aggregation method>`) defines the average curviness of the roads in the neighborhood of this segment. This metric was selected as the curvilinearity of the roads within a street pattern was observed as one of the primary characterizing attributes.

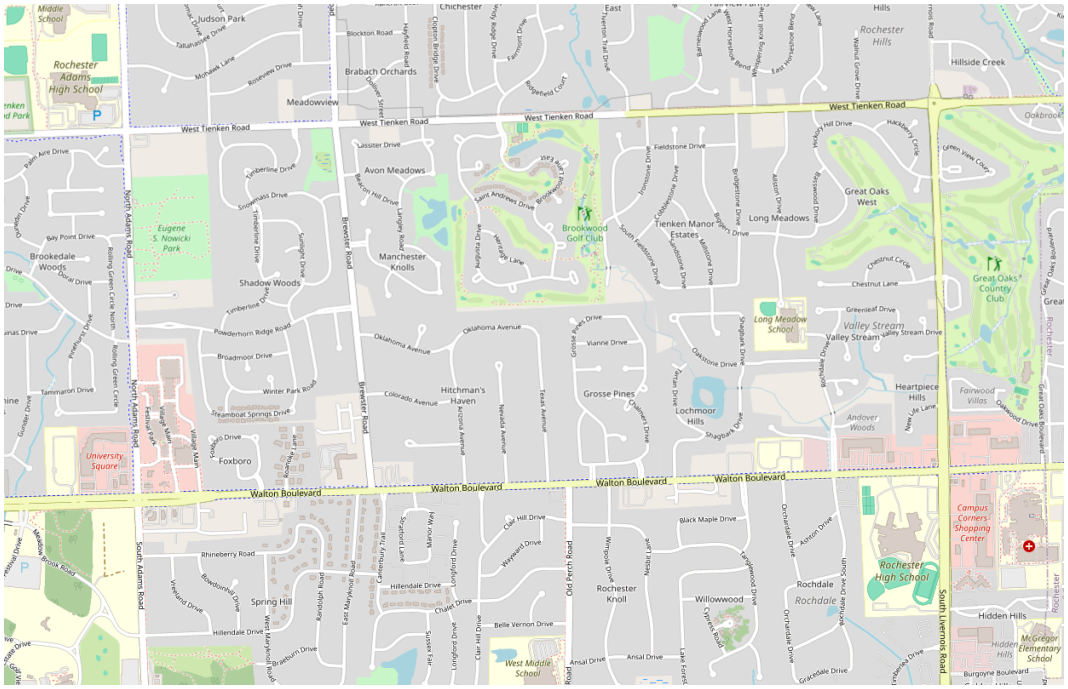
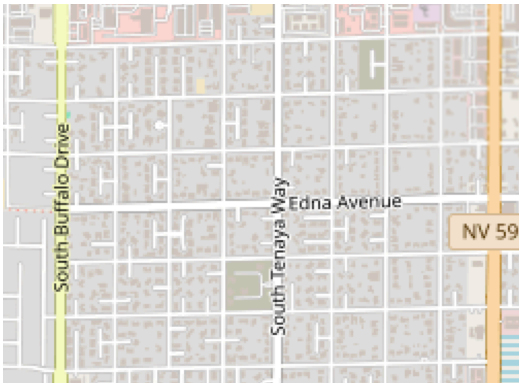


Figure 40 - Streets with high curvilinearity within major streets of low curvilinearity

B NEIGHBOURHOOD CONTINUITY

`continuity::nb::weighted_median` defines how uninterrupted the median street within this neighborhood is, weighted by segment length. The median was selected instead of the mean to discard any influence from small service streets within the overarching street pattern that does not significantly impact the urban tissue. This metric was selected to distinguish between areas that are similar in all other metrics, but still visually distinct. See Figure 41.

(a) Street pattern with low median continuity. (OpenStreetMap contributors, 2017)

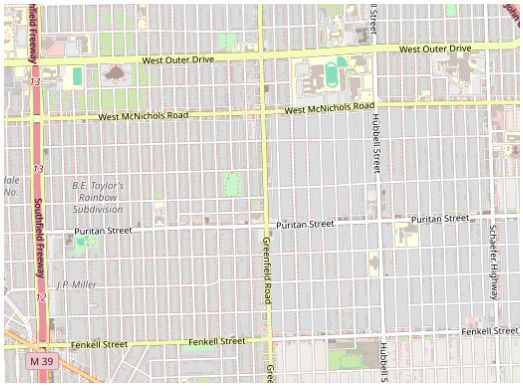


(b) Street pattern with high median continuity. (OpenStreetMap contributors, 2017)

Figure 41 - Two street patterns distinguished by the continuity that are visually distinct.

C NEIGHBOURHOOD ANGLE ENTROPY

`angle_entropy::nb` defines how regular the orientations of the streets in this neighborhood are. This metric was chosen to specifically distinguish out grid patterns, in combination with the continuity metric.



(a) Street pattern with low angle entropy. (OpenStreetMap contributors, 2017)

(b) Street pattern with high angle entropy. (OpenStreetMap contributors, 2017)

Figure 42 - Two distinct street patterns distinguished by angle entropy.

D NEIGHBOURHOOD RIGHT NEIGHBOUR DISTANCE

`right_neighbour_distance::nb::weighted_mean` defines the average distance to the closest street segment located to the right of the segments in the neighborhood, with a limit of 300 meters per segment. This metric is scale-sensitive, and the main distinguishing feature for density.

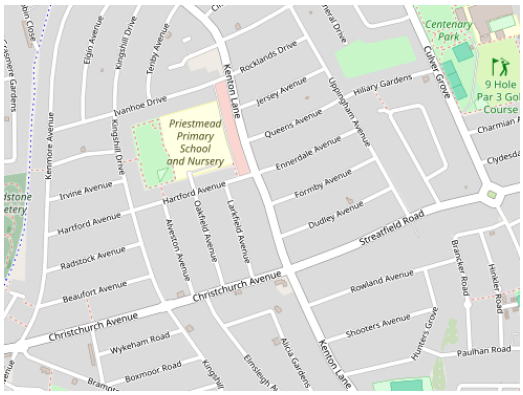
(a) Street pattern with low mean right neighbor distance. (OpenStreetMap contributors, 2017)

(b) Street pattern with high mean right neighbor distance. (OpenStreetMap contributors, 2017)

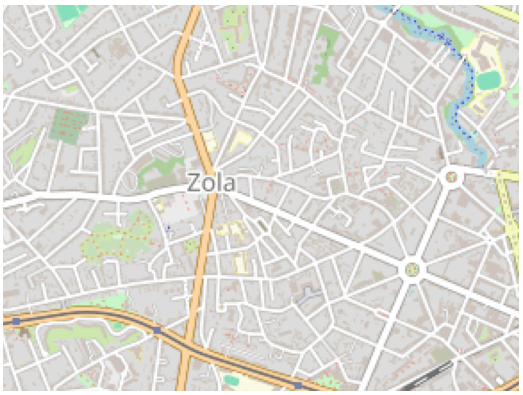
Figure 43 - Two distinct street patterns with vastly different right neighbor distance.

E NEIGHBOURHOOD RIGHT NEIGHBOUR ANGLE DEVIATION

`right_neighbour_angle_deviation::nb::weighted_mean` defines the average difference between the angle of the current segment and that of its neighbor on the right. This metric aims to capture to what extent streets are parallel to the one next to it. Angle entropy already captures this for grid-like street patterns, but parallel streets also occur in patterns that do not have regular angles across the whole neighborhood, as can be seen in Figure 44.



(a) Street pattern with low mean right neighbor angle deviation. (OpenStreetMap contributors, 2017)



(b) Street pattern with high mean right neighbor angle deviation. (OpenStreetMap contributors, 2017)

Figure 44 - Two distinct street patterns with different right neighbor angle deviation.

F NEIGBOURHOOD SECTION LENGTH STANDARD DEVIATION.

`section_length::nb::std` defines to what extent the section lengths differ within a segment neighborhood. This metric aims to capture a level of "regularness" in a pattern that is not captured by any of the other metrics.

(a) Street pattern with high regularness of section lengths. (OpenStreetMap contributors, 2017)

(b) Street pattern with low regularness of section lengths. (OpenStreetMap contributors, 2017)

Figure 45 - Two distinct street patterns distinguished by regularness of section lengths.

Computing the classes using GMM requires the passing of a value for the amount of classes k. For this thesis, clusters were computed for values of k ranging from 6 to 18. The elbow method is a popular method for determining the optimal amount of clusters by identifying the point with the biggest change in angle in a curve with a number of clusters on the x-axis and a scoring metric on the y-axis. This method was also used by Fleischmann et al. (2022) in combination with using the Bayesian information criterion (BIC) for estimating the goodness of fit of the model (Schwarz, 1978) to determine the optimal amount of clusters. I, therefore, use this same methodology for estimating the optimal amount of clusters.

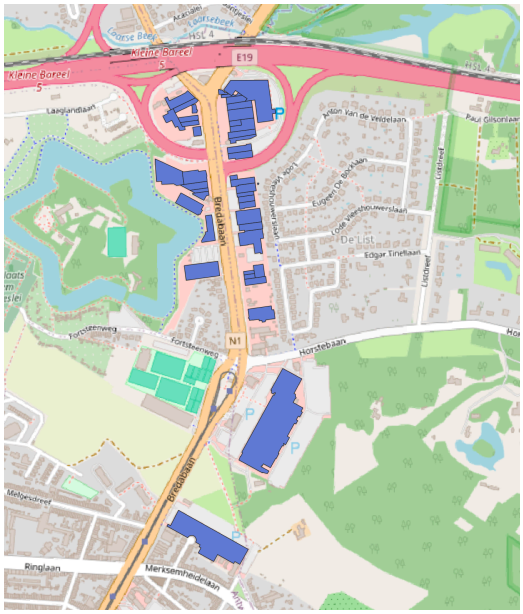
BUILDINGS CLASSIFICATION

The classification model chosen is gradient boosting, more specifically the CatBoost model (Dorogush et al., 2018). This model was chosen for its good handling of complex relationships, support for missing values and categorical data, and good performance.

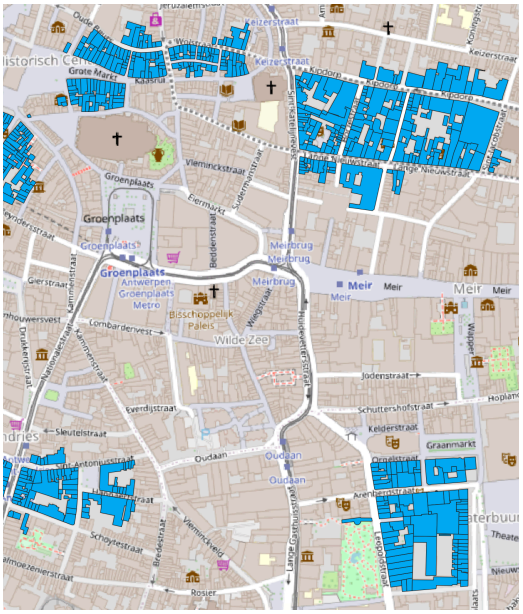
First, a ground-truth dataset had to be created by labeling footprints from real-world data. A subset of the city dataset was used, ensuring an even spread across continents. See Figure 46. Originally, Hong Kong was also part of the dataset, but it was discarded after the apartment typology from this data seemed to have a negative effect on the classification performance of all other cities.

Figure 47 shows examples of building footprints selected as ground truth for each of the building classes.

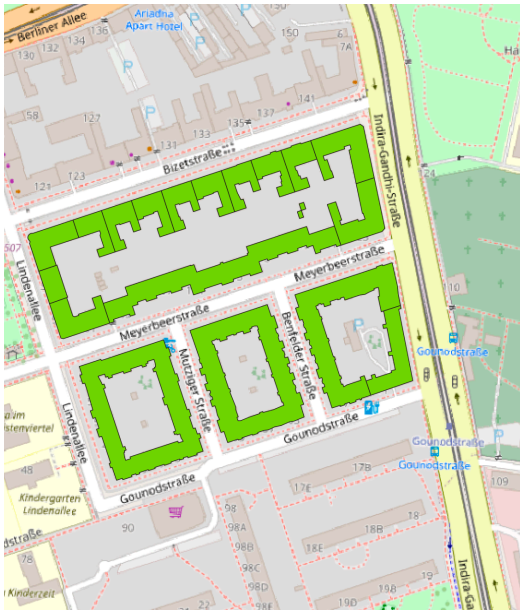
Figure 46 - Selected cities for ground truth dataset across the world. Background from OpenStreetMap contributors (2017)



(a) `big_commercial` (Antwerp, Belgium)

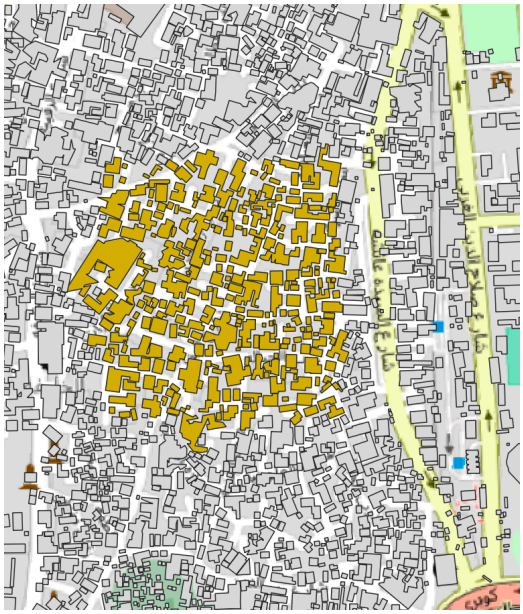


(b) `filled_block` (Antwerp, Belgium)

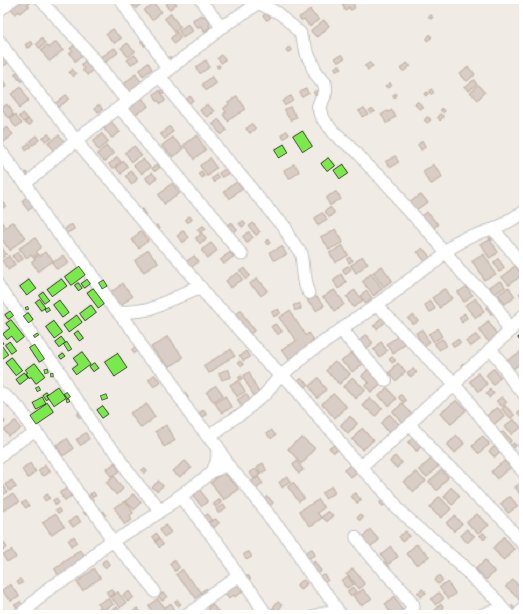


(c) `perimeter_block` (Berlin, Germany)

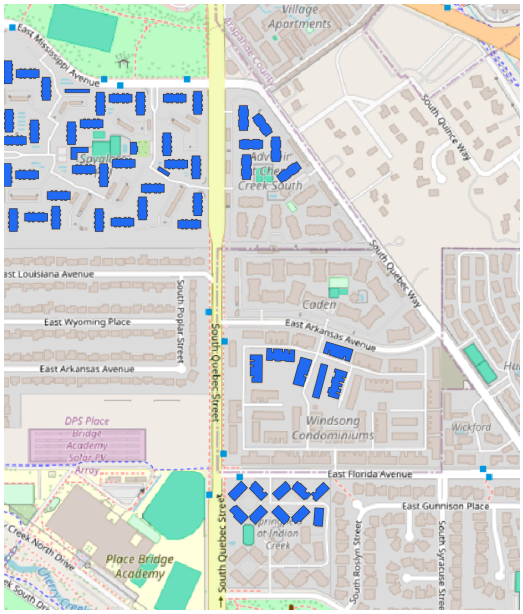
(d) `complex` (Berlin, Germany)



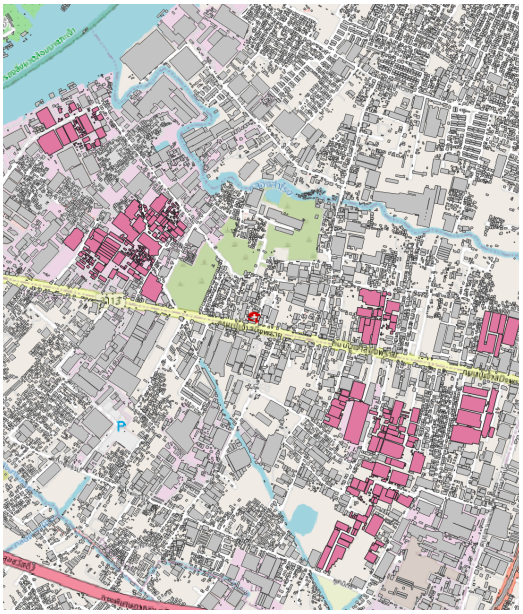
(e) `irregular_block` (Cairo, Egypt)
.



(f) `detached` in (Kinshasa, Democratic Republic of the Congo)



(g) `apartments` (Denver, USA)



(h) `industrial` (Bangkok, Thailand)

Figure 47 - Examples from ground truth dataset for building typologies (Backgrounds from OpenStreetMap contributors (2017))

Figure 48 shows a limitation of the ground truth dataset. As can be seen in Figure 48a, the real-world area consists of filled city blocks, but Figure 48b shows that the building footprints from sources that use AI generation do not, in fact, form filled blocks.

The full ground truth dataset, with typologies spread across the different cities, was used to train the CatBoost model.

4.3 Encoding

This section provides a detailed explanation of the Encode step, as outlined in Section 4.1.2.

4.3.1 The City Template

TYPOLOGY GRID ENCODING

The value of a cell in the typology grid is defined as the typology whose elements intersect the most with the area of that cell. This is more accurate than taking the mode, as the amount of elements of a certain typology does not automatically constitute the most dominant typology. Road segments can differ vastly in length, and buildings can differ vastly in area.

(a) Real world area in Montevideo, Uruguay (Airbus, 2024; Google, 2024; Google Earth, 2024; Maxar Technologies, 2024)

(b) Classified building footprints, with clearly visible effect of inaccurate footprints being assigned a different class. (Source data from OpenStreetMap contributors (2017))

Figure 48 - Influence of building footprint data quality on classification.

TYPOLOGY GRID CONVOLUTION

The resulting typology grid can have a noisy and mosaic-like appearance due to small areas of different classes within larger areas of more predominant classes. This can, for example, be caused by the dense road pattern in a parking lot. This absence of clear and contiguous areas harms the readability of the resulting map. It also degrades the generation performance with the simulated annealing technique, as the noisy pattern is more complex and more challenging to describe with landscape analysis metrics. Most patch-level shape metrics, see …, will give little to no distinguishable information for patches of just one or two cells.

A convolution process is used to counteract this noisy effect. In this process, a 3×3 kernel is applied to each grid cell using a mode filter. This filter is highly suitable for categorical data, as it can apply a smoothing effect on non-numerical data by choosing the predominant category in the neighborhood for every cell. This results in more clumped landscapes, making spatial patterns more obvious (He et al., 2002), see Figure 78. A downside is that dominant classes become more pronounced, while rare classes can become less prevalent (Coulston et al., 2014).

The result is a more consistent and easier-to-interpret typology grid, but this comes at the cost of possible overgeneralization. A high level of generalization can still be argued to be favorable since the aim is to detect contiguous areas, not tiny patches of suddenly different urban tissue.

See Figure 78a and Figure 78b for the before and after.

4.3.2 The Typology Template

The typology grid describes the spatial distribution of a layer's different typologies but does not describe the typology itself. That is where the typology template comes in. This template consists of statistical descriptions of the typology's defining characteristics. These descriptions are geared towards the specific values needed for the generation phase, where neighborhood data is (partly) known, and the parameters need to be determined for the next element to be constructed.

Typology templates can be defined both globally and locally. A global typology template is constructed from the data of that typology within all analyzed cities, while the local typology template only considers data from one specific city.

I developed a proof of concept for typology templates for the *minor roads* layer to be used in combination with the adapted Parish & Müller (2001) method for road generation. The input parameters for this generation method are *segment angle*, *segment length*, and a decision if the road should stop (node degree 1), continue straight (node degree 2), and it should split left and/or right (node degree 3 or 4). The typology template was designed to follow the inverse procedural modeling approach, determining the required input parameters for the generation algorithm from the input statistics.

Firstly, the input parameters of the generation algorithm are available in the data of the analyzed cities. *Segment angle* corresponds to the *forward_angle* attribute, *segment length* to the *length* attribute, and *node degree* to the *node degree* attribute. The distribution and probability of possible values for each of these parameters can be approximated by creating a probability density function or a probability histogram. The core of the typology template are the three probability distributions of these input parameters. Figure 49 shows an example distribution for the *forward angle* attribute of *cluster 3* for a local template of the city of Antwerp.

Visual investigation of road generation results from random sampling of these distributions showed that just the probability of the parameter is not sufficient for realistic results. Investigating the source data from OpenStreetMap showed apparent correlations, like multiple short segments following each other in curved segments or curved segments often following a consistent curvature across multiple segments. Therefore, new attributes were added to the analysis data to capture attributes of the "previous" segment, includ-

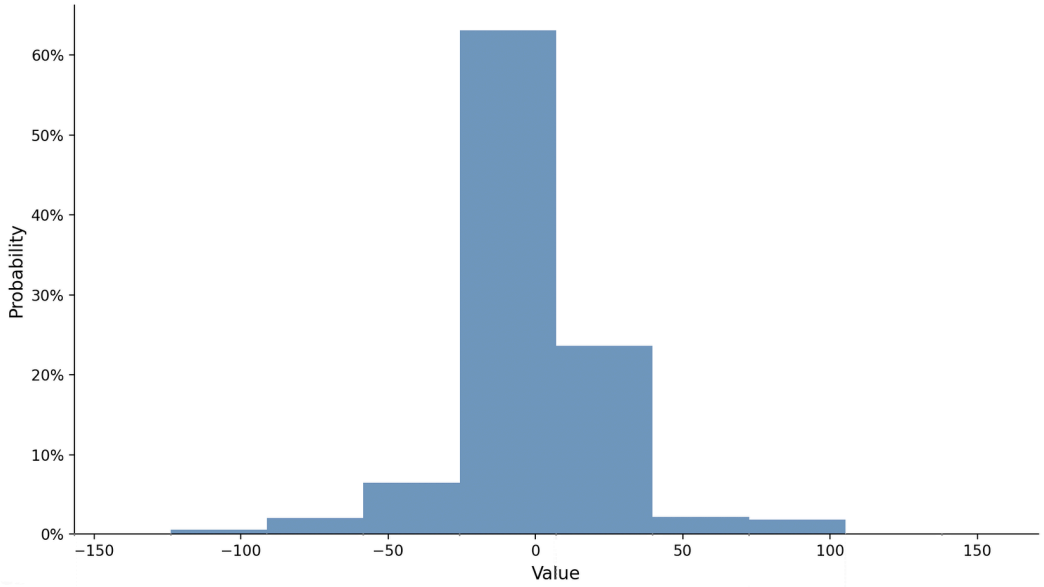


Figure 49 - Example probability histogram of forward angle *attribute for a local typology template.*

ing *previous_segment_forward_angle*, *previous_segment_node_degree*, and *previous_segment_length*.

One option for encoding the complex relationship between these "previous segment" variables and the target parameters is to use a deep learning approach. However, my aim in this thesis was to avoid opaque methods that obfuscate why a certain outcome was reached. I do, however, think that many opportunities lie in investigating deep learning approaches with possibly higher-quality outcomes, as also Breiman (2001) states that these models can represent reality more closely.

The approach I chose was to encode the influence of the "previous segment" variables statistically. The histogram is divided into evenly sized bins. Then, separately for each bin, the relationship between the value of each influencing variable and the probability of that specific bin is encoded using a probability density function for each influencing variable.

The probability for a bin can be defined as:

$$P(\text{bin}) = P_{\text{unweighted}}(\text{bin}) * \text{weight}_{\text{influence}}(V)$$

$$\text{weight}_{\text{influence}} = \frac{\sum_{i=1}^{n} P(D_i(V_i))}{n} \quad 18.$$

Where V is the set of influence values for a specific context, n is the total number of influence values, and $P\left(D_{i(V_i)}\right)$ represents the probability of the i-th influence value V_i under the corresponding distribution D_i.

Note that the final bin probabilities need to be normalized to add up to 1.0 across all bins, which is not incorporated in Equation 18.

For example, let's take the −70° to −30° bin. In the probability histogram, it might have a probability of 7%. The previous segment forward angle is −60°, the probability density function for this influence metric might give a much higher than average probability (reflecting the observation that segment angles are often in a consistent direction). Then, the previous segment length and distance to the last intersection might have less influence, both giving a roughly average probability. As a result, the final probability of the −70° to −30° bin might, for example, end up as a much higher 20%, while the probability of other bins was adjusted based on their influence probability distributions. I do not know if a similar approach has been introduced in earlier research or if it is specific to this thesis.

The Cauchy (red) and LogNormal (blue) distributions are currently implemented, where the most suitable distribution was chosen by minimizing the Error of Sum Squares (the default scoring metric in the used library). These distributions were chosen based on their low error with the data that was being experimented with, but more distributions should be added. Bins with little or inconsistent data cannot produce accurate distribution functions, these are discarded (indicated by either of the top two statistics in the cell being red). The top statistic is the error, and the bottom is the Kolmogorov–Smirnov statistic. The thresholds used were a Sum of Squares error above 1.0 or a Kolmogorov-Smirnov statistic above 0.4 and were arbitrarily chosen based on visual inspection of the quality of the resulting distributions.

Figure Figure 50 shows a summary overview of the *forward angle* distribution within the typology template.

Visual inspection of the generation results appeared to show a noticeable improvement in realism with the addition of the influence values (see Figure 51).

Typology templates for the buildings layer were not implemented, as the proof of concept implementation of the building generation methods does not take any input parameters. Different building generation methods would require different parameters, that could also be encoded in a similar inverse procedural modeling approach. The building generation methods that place sample buildings would also require a dataset of building footprints inside the typology template so that

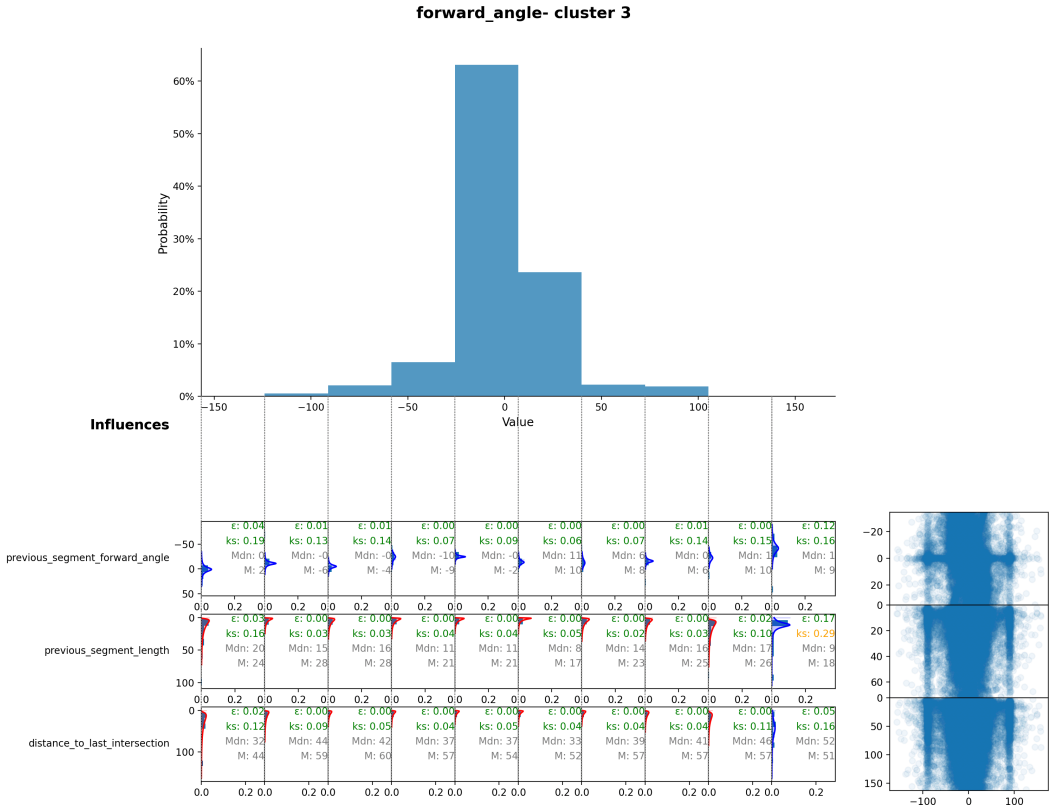


Figure 50 - Overview schematic of a typology template encoding of one of the three attributes (in this case forward angle). The top histogram shows the probability distribution of the different values of this attribute, divided into evenly sized bins. Below, each of the influencing attributes has one probability distribution per bin. Blue probability distributions are LogNormal distributions, while red signifies Cauchy distributions.

representative buildings for the typology or in the case of a local template of that specific city can be placed.

4.4 Generation

This section provides a detailed explanation of the Generate step, as outlined in Section 4.1.3.

4.4.1 Typology Grid Generation

A process of simulated annealing is used for the grid generation.

For this proof of concept, the typology grid from the template city is taken as a starting point for the simulation. For every simulation tick, the state of the grid is modified. Two methods were investigated: one where the values of two

randomly selected cells are swapped and another where one randomly selected cell is swapped with a random neighboring cell of a different typology. If it has no neighbors with a different typology, a new random cell is selected. Using the original grid as a starting point limits the flexibility, and a future implementation would benefit from being able to generate grids that are similar to the template without needing to be the exact same size.

The objective function compares the target grid of the original city in the template with the current grid on every tick. This is done separately for every typology class. The aim of the objective function is to score how similar the current grid is to the original city. For this, metrics from the field of landscape analysis are employed. The typology grid is analyzed by separating it into "patches" of connected areas of the same typology (rooks case).

A set of metrics is computed for each patch. These metrics were selected from the set of patch metrics from the FRAGSTATS software (McGarigal & Marks, 1995).

Patch Area is simply the area of the entire patch.

$$\mathrm{AREA}_{\mathrm{patch}} = N \times A_{\mathrm{cell}} \tag{19.}$$

with A_{cell} the area of a single cell of the typology grid and N the amount of cells this patch consists of.

Patch Core Area Index defines the ratio between the core area of a patch and the patch area as defined by Equation 19. Where the patch core is defined as the cells of the patch that do not lie on the perimeter.

$$\mathrm{CAI}_{\mathrm{patch}} = \frac{\mathrm{AREA}_{\mathrm{patch\_core}}}{\mathrm{AREA}_{\mathrm{patch}}} \tag{20.}$$

Patch Shape Index attempts to describe the irregularity of the patch shape in a scale-invariant way.

$$\mathrm{SHAPE}_{\mathrm{patch}} = \frac{0.25 \times \mathrm{PERIMETER}_{\mathrm{patch}}}{\sqrt{\mathrm{AREA}_{\mathrm{patch}}}} \tag{21.}$$

By analyzing the patches of the typology grid with these metrics, it can be compared to how similar the patches of the current state of the grid are to the original grid.

A problem with this approach is that cities do not have similar patterns across their domain. Patches of city blocks are more likely to be near the city center, while further away from the city center, they might only exist as small patches (for example, a small historic center of an agglomerated town). Many more complex relations can exist, like big commercial buildings being close to major roads, and industrial buildings more likely to exist in gridded than organic street patterns. Because the objective function of the simulated annealing method can be designed freely, it's possible to encode these relations in the comparison.

I investigate a statistical method to encode these relations by combining target metrics (i.e., patch area, core area index, shape index) with a "binning metric". The idea behind this is that the similarity of the target metric depends on the binning metric. For example, two typology grids might have the same statistical distribution of patch areas of the *perimeter block*, but if the patch areas are compared in bins according to distance to the city center, it might show that one grid has most of the *perimeter block* around the city center while the other has the patches scattered throughout the whole domain.

Binning is done according to an aggregation function, for example average, sum, or max. For each class, the patches are divided into bins based on the binning metric. All patch metric values are subsequently aggregated using the selected aggregation function. The final result is a list of histograms for each class, one for every metric-binning-aggregation combination.

Subsequently, each histogram is compared with the corresponding histogram of the starting typology grid to assign a value from 0-100% similarity. Initially, cosine similarity was considered for determining the similarity, but this was consistently assigned too high of a similarity score due to it not being scale-invariant. Instead, the intersection between the two histograms was computed as the similarity. This is computed as follows:

$$\text{similarity}(A, B) = \frac{\sum_{i=1}^{n} \min(a_i, b_i)}{\max\left(\sum_{i=1}^{n} a_i, \sum_{i=1}^{n} b_i\right)} \tag{22.}$$

where A is the current histogram as a vector of each value per bin, and B is the target histogram.

The metrics that are being compared are:

Compared metric	Binning metric	Average	Sum	Max
Patch Area	Patch distance to city center	X	X	X
Patch Core Area Index	Patch distance to city center	X		X

Patch Core Area Index	Patch Area	X
Patch Shape Index	Patch distance to city center	X

An opportunity exists to include metrics that incorporate the constraints between different layers in the city stack, such as the passability of the terrain or the distance to the closest river. Due to time constraints, these were not implemented for this thesis.

The final similarity of the grid is

$$S = \frac{\sum_{i=1}^{N} \sum_{j=1}^{n} \text{similarity}(A_{\text{ij}}, B_{\text{ij}})}{N * n} \quad 23.$$

with N as the number of classes and n the number of metric-binning-aggregation combinations.

The final score is the average of all similarities of all metrics across all classes.

To determine if the new state is accepted, the difference in score between the old state and the current state is fed to the following function, called the Metropolis Hastings Criterium:

$$P = e^{-\frac{\Delta}{T}} \quad 24.$$

With P being the probability of accepting the state, Δ being the signed difference between the old and new state, and T being the current annealing temperature. This ensures that at high temperatures, almost all states are accepted, and with decreasing temperatures, more and more "bad" states get rejected. This helps escape local minima.

4.4.2 Street Network Generation

The algorithm works as follows, with a segment query being a road segment that is being considered but not finalized and a road segment being a finalized segment in the road system. The algorithm is heavily based on the L-system approach from Parish & Müller (2001), more specifically, the adapted priority queue-based approach from Barett (2009).

The main change is replacing the global goals function and the way the node degree is decided. In the original implementation, the global goals function generated new segment queries based on a population map and a set of rules to create a grid, organic, or radial pattern. This has been replaced by generating new segment queries based on the local road typology from the typology grid.

More specifically, where Parish & Müller (2001) does not adapt to the local context in the global goals function (by design), the new approach adapts the generated query to metrics from the previously built segment (the so-called influence values). Section 4.3.2 describes how the *segment angle* and *segment length* parameters of Parish & Müller's algorithm can be derived from the typology statistics. The same can be done to determine if the road should continue and if a new segment should spawn to the left and or/right. Figure 51 shows the difference between deciding these three parameters based on a simple probability distribution ()

Start from a single segment query in priority queue Q, an empty road segment collection S, the typology grid G, and the typology template collection T

While Q is not empty do

  Get the next segment query q from Q

  If q intersects with any existing segment s in S

    Modify q so it's truncated to the intersection with s

    Split s in s_1 and s_2 and replace s with the new segments in S

  Else if the end point e_q of q is within threshold t of an existing end point e_s of a segment in S

    Modify q so that end point e_q is at the same location as e_s, essentially snapping to the existing intersection.

  End if

  Construct segment s from the (possibly) updated q and add it to S

  This marks the end of segment construction, now follows the creation of new queries.

  Get the influence values i_{previous} (encoded in the struct retrieved from Q), including previous segment length, previous segment forward angle, and distance to last proper intersection.

  Get the closest typology grid value g from G and the corresponding typology template t from T

  Sample next node degree d_{new} from t using i_{previous} as input for the encoded distributions.

  If d_{new} is 1

    This is a dead end, **Continue**

  End if

  If d_{new} is 2 or greater

    Sample forward angle α_{new} and segment length l_{new} from t using i_{previous} as input for the encoded distributions.

Add a new segment query q_{new} to Q with angle α_{new} and segment length l_{new}.

End if

If d_{new} is 3

Intersection is a T-split, randomly choose left or right. Sample segment length l_{new} from t. Currently, an angle of 90° is always used.

Add the left or right query to Q with a delayed importance value, so it will be constructed later (as suggested by (Parish & Müller, 2001))

Else if d_{new} is 4

Perform the same steps as for d_{new} is 3, but in both left and right direction.

End if

End while

(a) Without influence values.

(b) With influence values.

(c) Real life typology this local typology template is based on. (OpenStreetMap contributors, 2017)

Figure 51 - Examples showing the effect that influence values have on road generation.

4.4.3 Building Generation

This section outlines the 5 different methods used for building generation.

Method 1 starts from an inset polygon of the city block, and recursively splits it to derive individual building footprints until a certain area threshold is reached. The splitting method uses an object-aligned bounding box, split at a ratio chosen from a distribution. Both the split ratio distribution and area threshold distribution are determined from the typology statistics.

Method 2 uses real building footprints from the template city sampled from the original data. These building footprints are associated with a specific area in the typology grid of the original city. During building generation, the group of building samples is determined from the local typology grid value. Then, the samples are distributed along the roads of the block according to the typology statistics.

Method 3 is based on method 1 but randomly discards finished buildings and insets the kept buildings to produce an irregular pattern.

Method 4 also starts from an inset polygon and creates a courtyard within this polygon using another inset.

Method 5 selects specific street segments from the block as starting points and, at a set distance from these streets, sweeps a row of identical houses to create a continuous row.

	Method				
	1	**2**	**3**	**4**	**5**
Detached Housing		X			
Terraced Housing					X
Filled City Blocks	X				
Perimeter City Blocks				X	
Irregular City Blocks			X		
Appartments		X			
Industrial Buildings		X			
Complex Buildings		X			
Big commercial buildings		X			

Table 4 - Which generation method is used for which building typology.

For every resulting enclosure from the road generation phase, a single method is chosen to correspond to the value of the closest cell on the typology grid. The method chosen for each typology class is according to Table 4.

METHOD 1: OBJECT-ORIENTED BOUNDING BOX SPLITTING

This method is designed to replicate filled city blocks, consisting of often densely connected building footprints with different aspect ratios to fill the block as densely as possible.

This method is essentially the same as the algorithm introduced by Parish & Müller (2001) for generating parcels within a city block, but the algorithm is adapted for this specific use case with chosen parameters and heuristic for stopping the recursive splitting when no longer connected to the street.

The algorithm works as follows:

1 Start from the enclosure polygon e and an empty list of final buildings B
2 Buffer e with a negative distance to create a new polygon e_{inset} with a setback from the street
3 Add this polygon to the split_queue S
4 **While S is not empty do**
5 Remove a polygon p from the split queue
6 Calculate the object-oriented bounding box o of the polygon (the smallest rotated rectangle that contains the whole polygon)
7 Place a split line perpendicular l to a point on the longest axis of o at a randomly chosen fraction between 30% and 70%
8 Split p into two (or more) resulting polygons $p_1 ... p_n$ using l as split line.
9 **For every resulting polygon p_i in $p_1 ... p_n$ do**
10 **If the area of p_i is smaller than a preset threshold**
11 P_i is a resulting building footprint and is added to B
12 **Else if splitting further would result in new polygons that are no longer connected to the exterior of e_{inset}** (and therefore not connected to the street).
13 p_i is a resulting building footprint and is added to B
14 **Else**
15 Add p_i to S
16 **End if**
17 **End for**
18 **End while**

Downsides of this method include a lack of control, because final results are highly sensitive to input enclosure shape and a big contrast between small and big buildings due to the effect of the city block aspect ratio on the splitting procedure, see Figure 52.

Figure 52 - Example result from method 1: OOBB method

METHOD 2: PLACEMENT

This method was developed to produce realistic footprints in a similar character as the template city without needing a complex grammar system like CGA from Müller et al. (2006) that needs expert input. It works on the assumption that you can create a similar building character by simply sampling existing buildings from the template city and placing them in the new context.

For this proof of concept implementation, the sampling from real footprints was not implemented. Only differently scaled squares are used, see Figure 53.

The algorithm works as follows:

1 Start from the enclosure polygon e, an empty list of final buildings B, and a collection of sample building footprints F
2 Take the exterior line string l of e
3 **While the end of l has not been reached do**
4 Get point p at the current position on l
5 Determine the angle α of l at p
6 Draw a sample building b from F

Try to place building b at location p with rotation α (the building footprints have their street access anchor at $(0, 0)$)
If e does not fully contain b
    Discard b
    Move the current position 3 meters further along l
Else if b intersects with any finalized building in B
    Discard b
    Move the current position 1 meter further along l
Else
    Add b as a finalized building to B
    Determine building spacing s and building width w
    Move the current position $w + s$ meters further along l
End if
End while

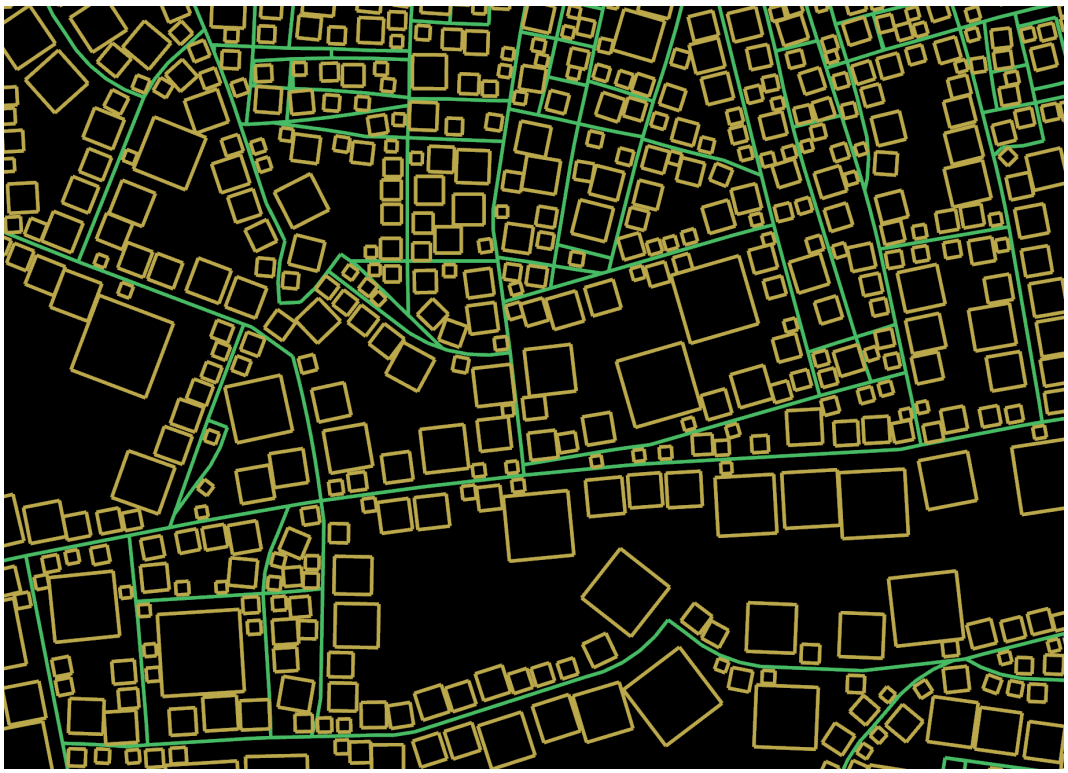


Figure 53 - Example result from method 2: Place method

METHOD 3: IRREGULAR BUILDINGS

The algorithm works the same as method 1, based on Parish & Müller (2001), but with the following changes:

1. The algorithm does not stop splitting when a resulting polygon p_i would no longer be contained in e_{inset}. This allows for buildings in the interior of the enclosure that are not connected to the street.
2. A finalized building has a 30% chance of being discarded to generate gaps in the resulting building pattern.

These changes were made to produce a similar pattern to the irregular city block typology. From observation, this pattern often consists of buildings with consistent orthogonal orientation but inconsistent aspect ratios, sizes, and spacing between the buildings. See Figure 54.

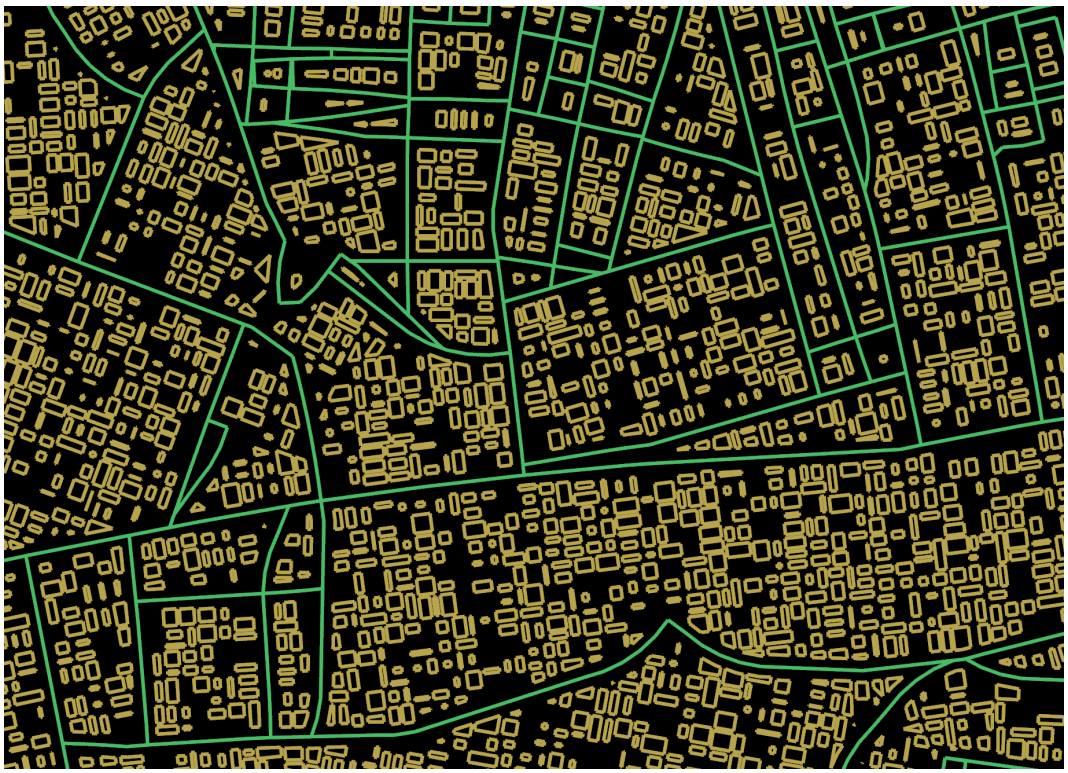


Figure 54 - Example result from method 3: Irregular method

METHOD 4: INSET

This very simple method was developed to create a similar urban tissue as the perimeter block building typology. The final result is quite oversimplified, see Figure 55, but adding further detail is non-trivial and was deemed outside the scope of the proof of concept nature of the building generation algorithms.

The algorithm works as follows:

1 Start from the enclosure polygon e, an empty list of final buildings B, and a collection of sample building footprints F

2 Buffer e with a negative distance to create a new polygon b_{exterior} as the building block exterior with a setback from the street

3 Buffer b_{exterior} with a negative distance to create a new polygon b_{interior} as the building block interior opening

4 The final building (block) b is defined as the exterior of b_{exterior} as outer ring and the exterior(s) of b_{interior} as inner ring(s)

Figure 55 - Example result from method 4: Inset method

METHOD 5: SWEEP

I developed this method to create urban tissue similar to the terraced housing building typology. The basic idea is to create rows of connected footprints, in this case, based on a simple rectangle shape. See Figure 56.

The algorithm works the same as Section , with the following changes:

1. The *overlap* predicate is used instead of the *intersects* predicate to determine if a new building can be placed. This allows for buildings that touch but do not overlap.
2. The exterior l of enclosure polygon e is simplified using the Ramer Douglas Peucker algorithm (Douglas & Peucker, 1973). This is to filter out small angle changes. Because every building is checked for overlap with all finalized

buildings, a small change in street angle could cause two buildings that were meant to only touch to overlap.

To improve realism, a heuristic could be developed to decide which streets should be used for terraced housing and which should be left empty, as real-life terraced housing areas often only have buildings on 2 out of 4 sides of a rectangular block.

Figure 56 - Example result from method 5: Sweep method

5 Implementation

5.1 Data and Code Availability

The GitHub repository for this project (https://github.com/OliverJPost/CityStack) contains all code written for this thesis. The README also contains information on how to access the produced (aggregated) data.

5.2 Overview

Part 1, "Analyze," and part 2, "Encode," are both executed using a command line tool called `citypy`. This tool is implemented in the Python programming language.

The Python language was chosen for these two steps for the following reasons.

- Availability of geospatial and analysis libraries, like `geopandas`, `osmnx`, and `momepy`.
- Previous experience with the language by the author
- Lower performance is less of a concern due to a city only needing to be analyzed once. The quicker development possibilities outweigh the performance concerns.

An example usage of `citypy` to analyze and encode a new city is as follows.

```
# Download all required data
citypy download "Washington DC" USA \
  --bbox="-77.505,38.697,-76.724,39.202,6.3"

# Add supplemental layers and compute all metrics
citypy process Washington_dc_USA.gpkg

# Aggregate building, road, and major road neighbourhoods
citypy contextualize Washington_dc_USA.gpkg --buildings --regular --major

# Apply road clusters
citypy clusters apply --file Washinton_dc.USA.gpkg -l road_edges --
col "street_cluster" --clusters citypy_gmm13_road_clusters.gmm --column-
filter="type_category=street"

# Apply building classes
citypy process Washinton_dc.USA.gpkg -s building_class

# Encode the result to a city character template and local typology templates
citypy encode Washington_dc_USA.gpkg --road-cluster-column "street_cluster"
--building-class-column building_class -o ./Washington_Template/
```

See Section 5.3 and Section 5.4 for implementation details.

Part 3 “Generate” is implemented in a separate tool, called `citystackgen`, which is implemented in the Rust programming language. This language was chosen for the generation step for the following reasons:

- High performance, which is necessary for the computationally intensive simulated annealing and element generation process.
- Memory safety ensures no memory leaks.
- Modern language features.
- Good interoperability with Python.
- Good libraries for generation and visualization, including `geo`, `ndarray`, and `rerun`

Example usage of `citystackgen` to generate a city based on a template is as follows.

```
citystackgen --template washington_template.npz --clusters-dir ./
Washington_Clusters
```

Subsequently, a Rerun (*Rerun-Io/rerun*, 2024) window opens to show the generation process while it’s running. Rerun is a graphical user interface for real-time data analysis. It not only allows for comprehensive insight into the generation process, including tools to plot variables in real-time but it also can be used for visualizing the results.

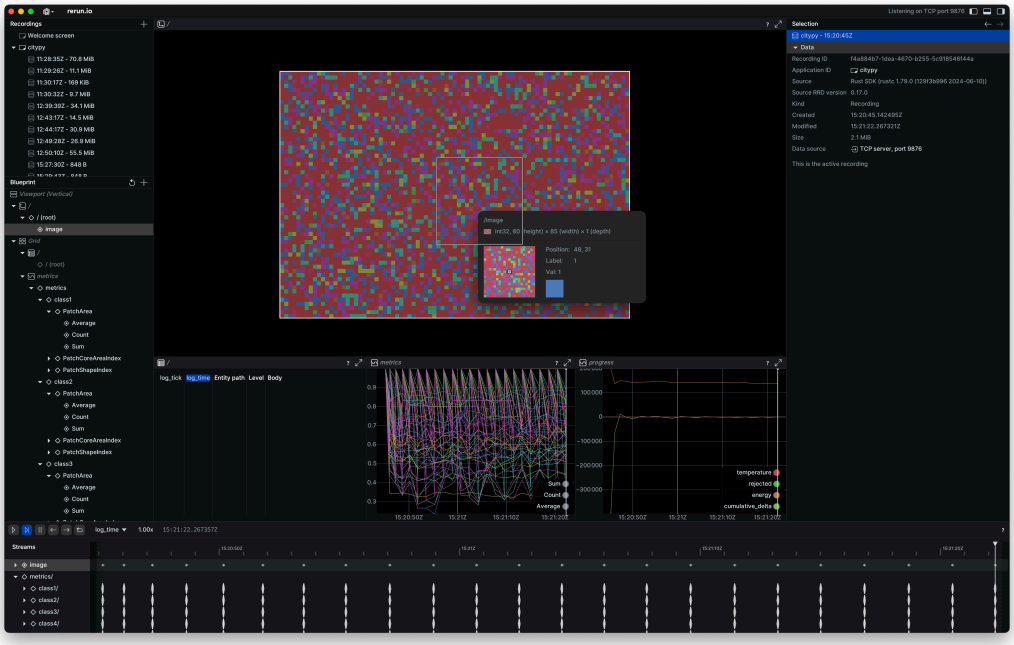


Figure 57 - Example of the Rerun interface for city generation

See Section 5.5 for implementation details.

Data Category	Format	Data Sources
Terrain Height	Raster converted to Points	OpenTopography (2021)
Water	Polygons/LineStrings	OpenStreetMap contributors (2017)
Buildings	Polygons	Overture Maps (2023) + Pesaresi (2023) (Building Height)
Road System	LineStrings/Points	OpenStreetMap contributors (2017)

Table 5 - Data source overview.

5.3 Analysis

5.3.1 Data Download

Table 5 gives an overview of the data sources used.

Height data is programmatically retrieved from the Copernicus GLO-90 OpenTopography API (OpenTopography, 2021). The usage of this API is free, but does require users to generate their own API token. Therefore, `citypy` requires users to provide their own OpenTopograpy API token.

The Python package `osmnx` (Boeing, 2023) is used for the download of the OpenStreetMap data from the publically available Overpass API. This package makes it easy to construct queries for specific data. A limitation of the Overpass API is the 1 Gigabyte download limit per day, which was not sufficient for the amount of data that was planned to be processed for this thesis. Therefore, a local instance of the Overpass API was set up on a TU Delft server to run the queries. This also sped up the download process significantly, as the public Overpass API works with a queue system.

The OvertureMaps building footprint data is also available with a public API, but due to timeout issues with big queries it was decided to download the entire OvertureMaps dataset to the TU Delft server and run the queries locally. For the GeoParquet queries, the `overturemaps` Python module was used.

All vector data is handled internally using the `geopandas` library. This came with many advantages for reading, handling, and modifying the geospatial data, but proved to also contribute significantly to the memory usage of the tool. Memory usage was analyzed using `memray`, and gradually increases during the download command and stays fairly constant during the entire process step. The only data structures that remain in memory for the entirety of this step are the `geopandas` dataframes (as the `networkx` graph is lazily constructed). A city like Berlin reached RAM usage of up to 24 Gigabytes and the Tokyo metropolitan area reached

above 100 Gigabytes. Future development of `geopolars` might bring improvements to this area, but this tool is not production ready.

Raster data is handled in a custom `GeoRaster` class, which uses the `rasterio` library internally.

All downloaded data is saved in the GeoPackage format. This format is based on a database approach. Major advantages include that all layers can be saved in a single file, unlike the Shapefile format, and very good performance due to built-in spatial indexing, unlike a text based format like GeoJSON. Final file sizes of some sample cities are as follows (note that these are file sizes after the completion of the full pipeline, the raw data after the `download` command is significantly lower).

5.3.2 Metric Computation

A system was developed to only compute metrics that are not already present in the GeoPackage or ones that were specifically selected in combination with an *overwrite* flag in the command. This proved to be a major time saver, as it allowed for quick computation of new metrics and recalculation of failed metrics. Also, every metric was wrapped in a `try except` statement, catching any potential errors and reporting them in the summary at the end. This also proved very useful, as edge cases caused certain metrics to error out for some cities, which otherwise would have crashed the computation of all metrics but now could be fixed and only that metric needed to be run again.

STREETS

A starting point for the road system metrics calculation were the metrics from the `osmnx` and `momepy` libraries. In the end many of the metrics were custom implementations of new methods, which sometimes caused them to be more computationally expensive than the polished metrics from these libraries. Upgrading to the latest version of `momepy`, which was released during the timeframe of this thesis, saw further speedups, indicating the advantage of using established algorithms.

A significant challenge in this thesis was the performance of the `right neighbour` metric. Even though the implementation utilizes the spatial index of the geopandas dataframes, and vectorizing the operations in chunks of 1000 segments, it still slowed down significantly with increasing city size. It slowed down to just 3 segments per second on the Tokyo metro area, meaning a total run time of almost 300 hours for this metric. This ultimately led to cancelling the biggest cities that were planned for analysis (Tokyo, New York, Los Angeles, and Jakarta).

Other noteworthy findings include speedups of up to 3000% by storing `networkx` graph attributes in a numpy array beforehand instead of accessing them every iteration. The `momepy` morphological tesselation sees RAM usage spike up to 3 times compared to the baseline, while the enclosed tesselation does not significantly increase RAM usage. Note that these findings were with a `momepy` version before 1.0.

BUILDINGS

The available metrics on building morphology from `momepy` were used extensively in this part. No noteworthy findings.

5.3.3 Neighbourhood Aggregation

The aggregation step proved to be a major computational challenge, both in terms of time complexity and space complexity. For this reason, this is the only part of the `citypy` tool that uses the `polars` library as a replacement for the `geopandas` library. The lazy evaluation of the queries in this library, combined with multi-threading by default, saw significant speedups. Aggregations were applied by joining the data with every other record within the neighborhood of that element.

STREETS

The `networkx` library was used to implement the recursive network growing algorithm. To achieve significant speed-ups, the road graph was first simplified to have one joined edge per road section instead of a separate edge for every road segment. Aggregations were later mapped back to individual segments using the section id attribute.

By performing recursive network growth with 5 steps, a neighborhood of a single segment quickly goes up to 200 segments. For a city like Tokyo this would mean 4.6 billion rows in the joined table. Even if this table just contained one 64-bit float per row, this would mean 34 gigabytes of RAM usage. Therefore, chunked processing of the data is unavoidable. Chunks of 10 thousand segments at a time proved not to significantly raise RAM usage above the baseline.

BUILDINGS

Building neighborhoods are defined using a radius around the current building, in this case, a radius of 300 meters. To efficiently select the buildings belonging to this neighbourhood, a k-dimensional tree from the `scipy` library was used. A much smaller chunk size of 100 had to be used, and still, RAM usage was noticeably higher than the usage in the processing stage of the same city (roughly 50% higher). Further investigation could reveal the cause, but because of the high execution speed of this method it was chosen to simply contextualize fewer cities at the same time.

Originally it was also planned to define a second type of neighbourhood: all buildings in the same enclosure. An oversight was that the maximum size of an enclosure is theoretically only constrained by the city domain, so the neighborhood might consist of an arbitrarily large amount of other elements. Even processing in chunks of 100 still saw very significant RAM increases, and this method was discarded.

5.3.4 Clustering & Classification

STREETS CLUSTERING

Clustering is performed as a separate command, which just outputs the clustering statistic plots and a single cluster summary file for every value of k selected.

```
citypy clusters compute \
    --dir . \
    --layer road_edges \
    --filter "type_category=street" \
    -k 14 -o "streets_k14" \
    -k 16 -o "streets_k16" \
    --on "stretch_curvilinearity::nb::weighted_mean" \
    --on "continuity::nb::weighted_median" \
    --on "angle_entropy::nb" \
    --on "right_neighbour_distance::nb::weighted_mean" \
    --on "right_neighbour_angle_deviation::nb::weighted_mean" \
    --on "intersection_left_angle::nb::std" \
    --on "section_length::nb::std"
```

The cluster summary file can subsequently be used to apply the clusters to any unseen city.

Since clustering is performed on all street segments matching the provided filter, across all analyzed cities, the RAM usage as a `(geo)pandas` dataframe was projected to be too big. Therefore, the code queries the GeoPackages directly using a database connection and SQL statements to only read the selected columns and store the values directly in `numpy` arrays.

BUILDINGS CLASSIFICATION

The `catboost` library was used for gradient boosting classification with support for both `NaN`, `None`, and categorical values. The fitted model could be exported to a single `.cbm` file for applying the classes to unseen cities. The model took an insignificant amount of time to train, presumably due to the small manually created dataset.

5.4 Encoding

5.4.1 The City Template

Typology Grid Encoding

The underlying grid is generated by dividing the domain of the city into 100×100 meter cells. Subsequently, these are saved as polygon geometries in a `geopandas` (Jordahl et al., 2020) dataframe. For every cell, an intersection check is made with all geometries of the layer that is being encoded using the RTree spatial index. The intersecting geometries are grouped by typology, and the one with the highest intersection length/area is chosen as the typology of that cell.

Typology Grid Convolution

The 3×3 mode kernel is applied using the `ndimage` module from `scipy` (Virtanen et al., 2020). A challenge was how to deal with `None` values, as currently the finer details on the outskirts of the city get lost as they are usually a single cell wide and surrounded by `None`. However, if `None` was not considered by the mode kernel, the whole domain gets flooded by typologies, including river and sea areas, giving a wrong representation of the city.

5.4.2 The Typology Template

Typology templates for the roads layer are exported as `JSON` files, with at the top level the three target parameters `foward_angle`, `segment_length` and `next_node_degree`. Then, each target parameter stores a list of bins, each storing the bin bounds (minimum and maximum), bin median, bin probability (since all bins together form a probability histogram), and influencing probability functions. Listing 1 shows an example of a single bin in the typology template. Influence probability distributions are either a Cauchy, LogNormal or Zero (no influence or too little samples) distribution and are determined using the `fitter` (Cokelaer, 2024) Python library. The serialization format used allows for the deserialization of the distributions in the generation step.

```
[
// Previous bins
{
  "bounds": [
        -135.1663886817872,
        -107.56387917170532
    ],
  "median": -114.7550481848244,
  "probability": 0.0009093664746892998,
  "influences": {
      "previous_segment_forward_angle": {
          "type": "Cauchy",
          "scale": 2.8189840800758343,
          "loc": -1.038290809078332
      },
      "previous_segment_length": {
          "type": "Zero"
      },
      "distance_to_last_intersection": {
          "type": "LogNormal",
          "s": 10.610821884860503,
          "loc": 9.952999999999998,
          "scale": 2.1808539933687663
      }
  }
},
// Next bins
]
```

Listing 1 - Example bin from a typology template

5.5 Generation

To accommodate the flexibility of the city stack in the generation process, combined with being able to substitute any generation method, I developed a code architecture based on the builder pattern. Figure 58 shows a simplified version of the architecture, showing only one type of `CityLayer` and `CityLayerBuilder`. The architecture allows for an arbitrary amount of `CityLayers`, each with an arbitrary choice of `CityLayerBuilders`.

The main advantage of this architecture is the low amount of coupling; since layers are stored in a standardized format, the subsequent layers that depend on the "existing" layers only depend on the `CityLayer` interface/trait. Also, most classes are quite focused in regards to their responsibility. A challenge for scaling this architecture is the dependency between different city layers, as currently, the building layer generation algorithm starts from the minor roads `CityLayer`. However, the city stack dictates that any new layer can be added to the stack, and

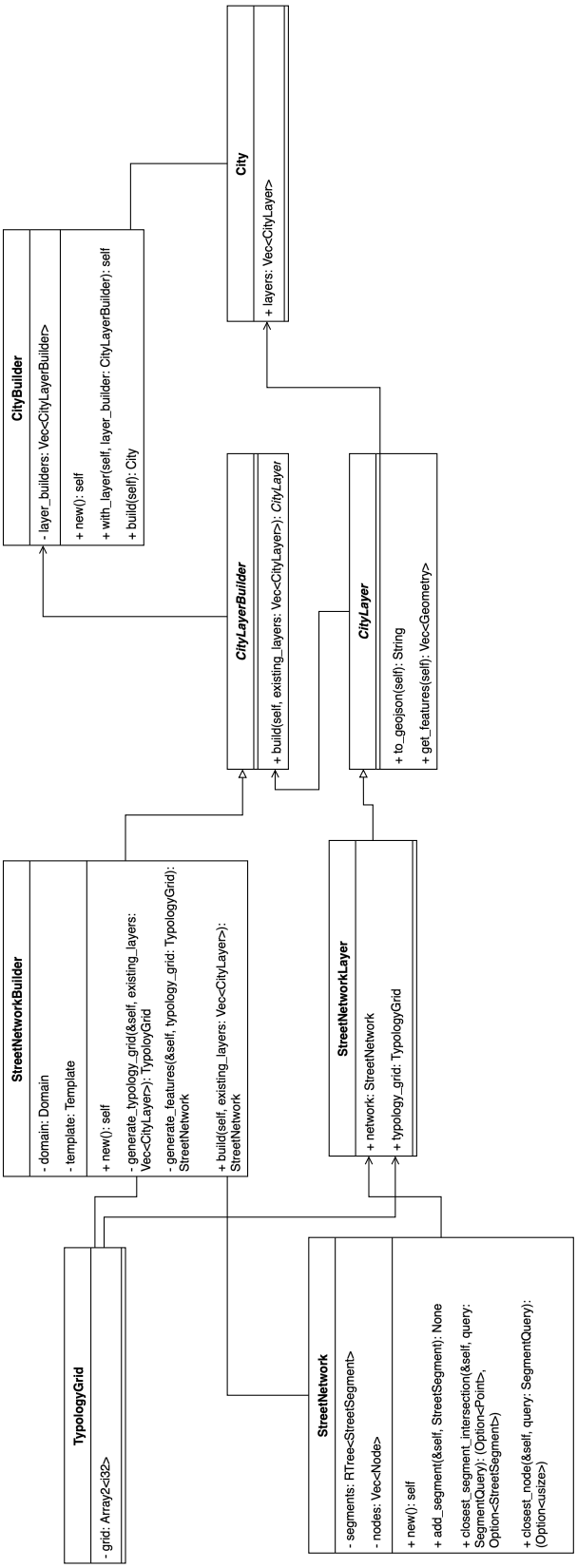


Figure 58 - Simplified architecture of the generation process.

there should also be the possibility to remove layers. If a new "minor points of interest" layer is introduced, the building generation algorithm should also take this layer into account. This kind of extendibility is difficult to implement in the architecture and would need further investigation.

5.5.1 Typology Grid Generation

The biggest challenge for typology grid generation was the performance of the simulated annealing process. The larger the grid, the more iterations are needed for the grid to converge, quickly needing millions of iterations. The following strategies were implemented to improve performance:

- Only the metrics for the modified patches are recomputed per iteration, including the patches that contain the modified cells and any of their four direct neighbors (rooks case).
- The minimum and maximum row and column values for each patch are cached to prevent iterating over the full grid for every patch.

Multi-threading proved to only increase computation time, as the biggest job that can be split up is to update one metric per thread or compare one class per thread. These jobs are so small that the overhead of multi-threading negates any performance gains. The iterations of the simulated annealing cannot be multi-threaded, as the next iteration needs the previous iteration to be finished. Another option to enable multi-threading would be to implement the parallel tempering technique, but this was deemed outside the scope of this thesis.

5.5.2 Street Network Generation

The adapted Parish & Müller (2001) street generation algorithm from Barett (2009) was rewritten in Rust code. Inspiration was taken from the JavaScript implementation by Hoof (2022) and the C++ implementation by Elinder (2017), specifically the separation into a `Segment` and `SegmentQuery` class to separate potential segments and finalized segments. Since every new segment needs to be checked for any intersections with any other segment, the finalized segments are stored in an RTree.

Distributions from the local typology template are deserialized into probability distribution types from the `RV` crate (Eaves & Schmidt, 2024) since these use more similar parameters for the `scipy` distributions than other available crates.

Since the subsequent layers need the street network to form a closed system to derive enclosures, it was very important to already create a closed system while generating. Even though a purely geometry based algorithm was used to generate the enclosure polygons, small imprecisions in the geometry were causing gaps in enclosure borders, leading to multiple enclosures being merged

together. To counteract this, the street network generation algorithm builds a graph on the go, by splitting segments when an intersection is encountered.

5.5.3 Building Generation

The implementation of the 5 building generation methods is quite straightforward, and closely follows the pseudocode algorithms. Geometry operations were implemented using `geo` (GeoRust, 2023). The main challenge that was encountered was the tolerance of the Egenhofer 9-IM operations with the `relate` method of `geo`. The `overlaps` method was often giving false positives for geometries that were only touching. To counteract this, operations that depended on the `overlaps` predicate were slightly offset with a negative distance.

6 Results

This chapter shows examples of the results of both the analysis, encoding, and generation steps.

6.1 Analysis

6.1.1 Road Clustering

The optimum amount of clusters was determined using the same methodology followed by Fleischmann et al. (2022). The goodness of fit of the model is evaluated by identifying the "elbow" of the curve of the Bayesian information criterion (BIC). The gradient of the curve decreased significantly after 13 clusters, so the optimum amount of clusters was determined as 13.

Figure 59 shows an example of the global clustering applied to one of the studied cities.

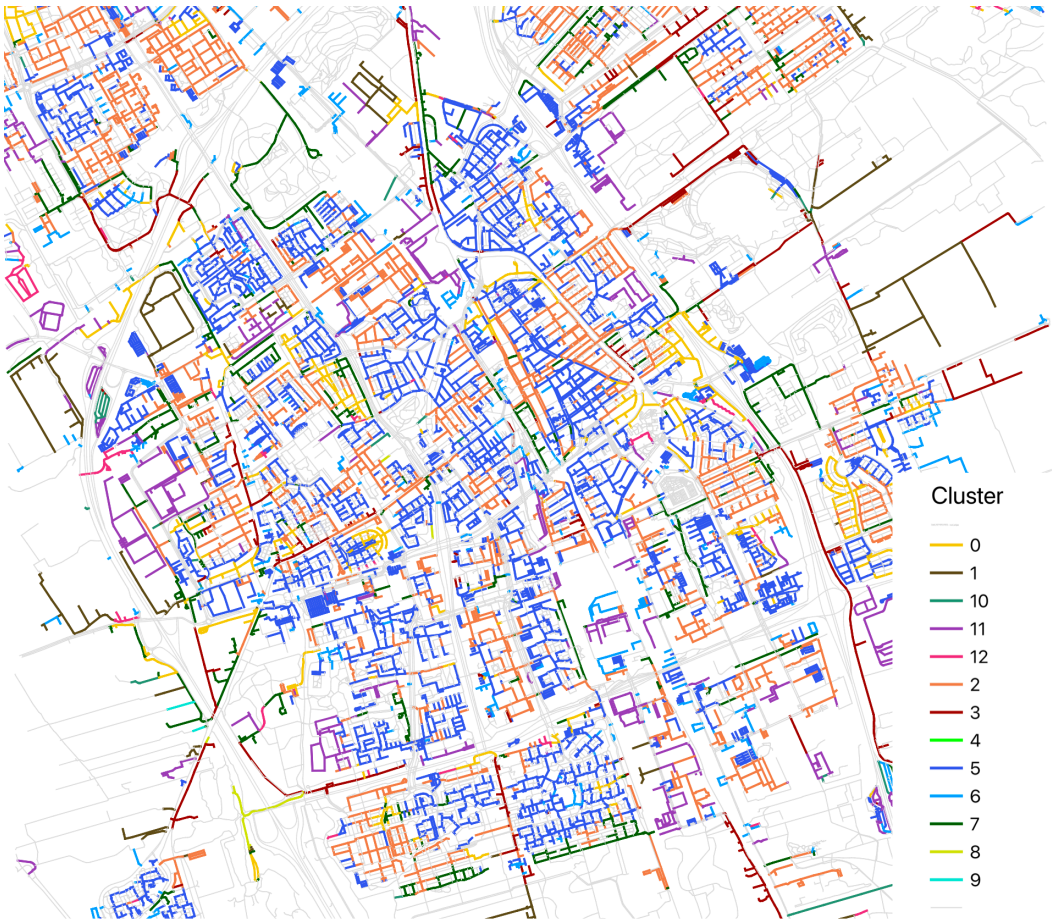
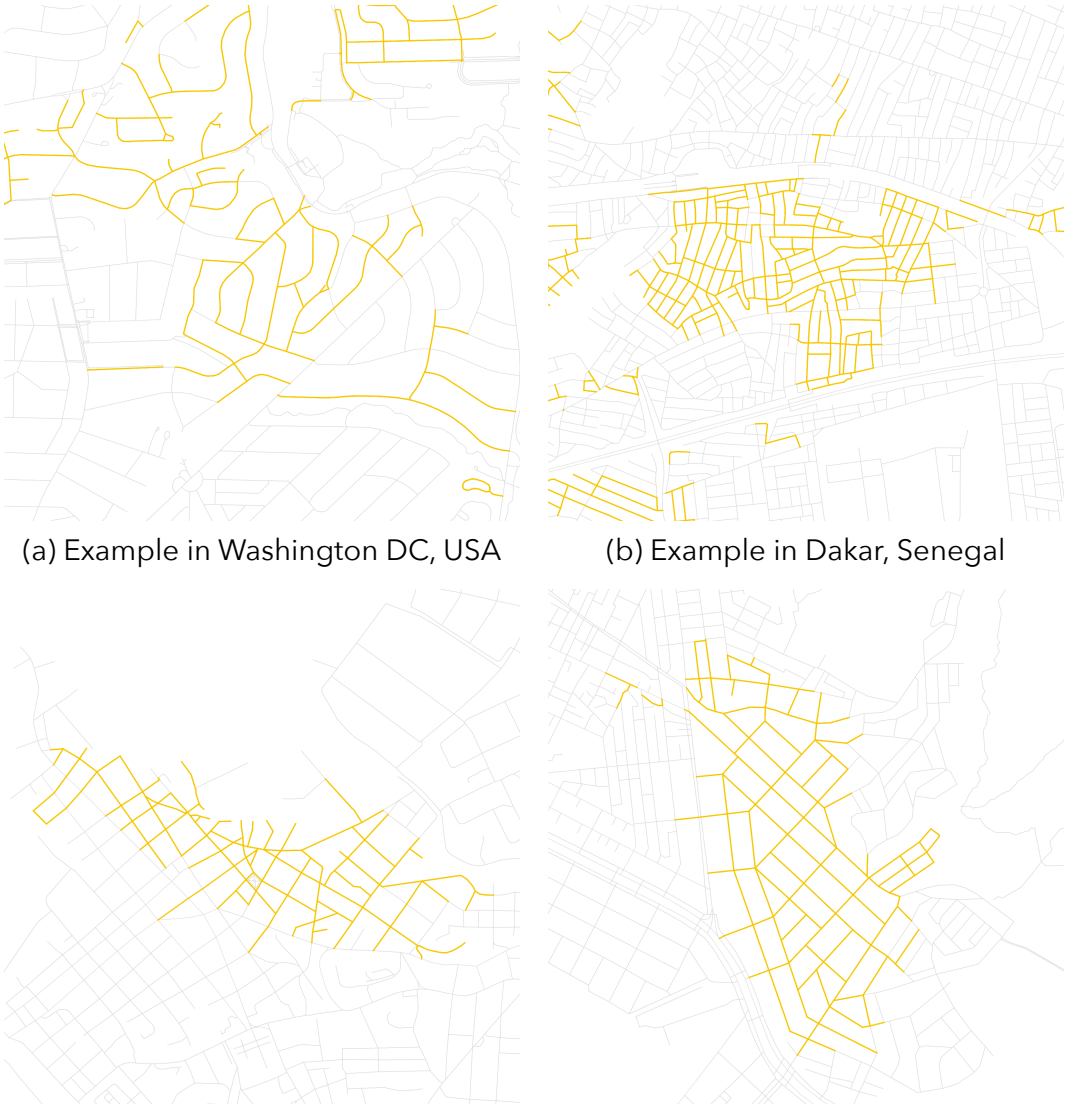


Figure 59 - Example of GMM clustering result applied to Delft, Netherlands. (Source data from OpenStreetMap contributors (2017))

Below, each cluster will be statistically and visually described to assess the validity of the resulting clustering.

CLUSTER 0 "ANGULAR STREETS"

Cluster 0 encompasses 9.9% of all analyzed road segments (by segment length). Of all analyzed cities, Kinshasa, Democratic Republic of the Congo, has the highest percentage of this cluster, with 29.7%. Z-scores (Appendix E) show significantly higher angle entropy and stretch curvilinearity and lower right neighbor distance than average. Visual inspection shows angular street patterns with occasional moderately curved streets.



(a) Example in Washington DC, USA

(b) Example in Dakar, Senegal

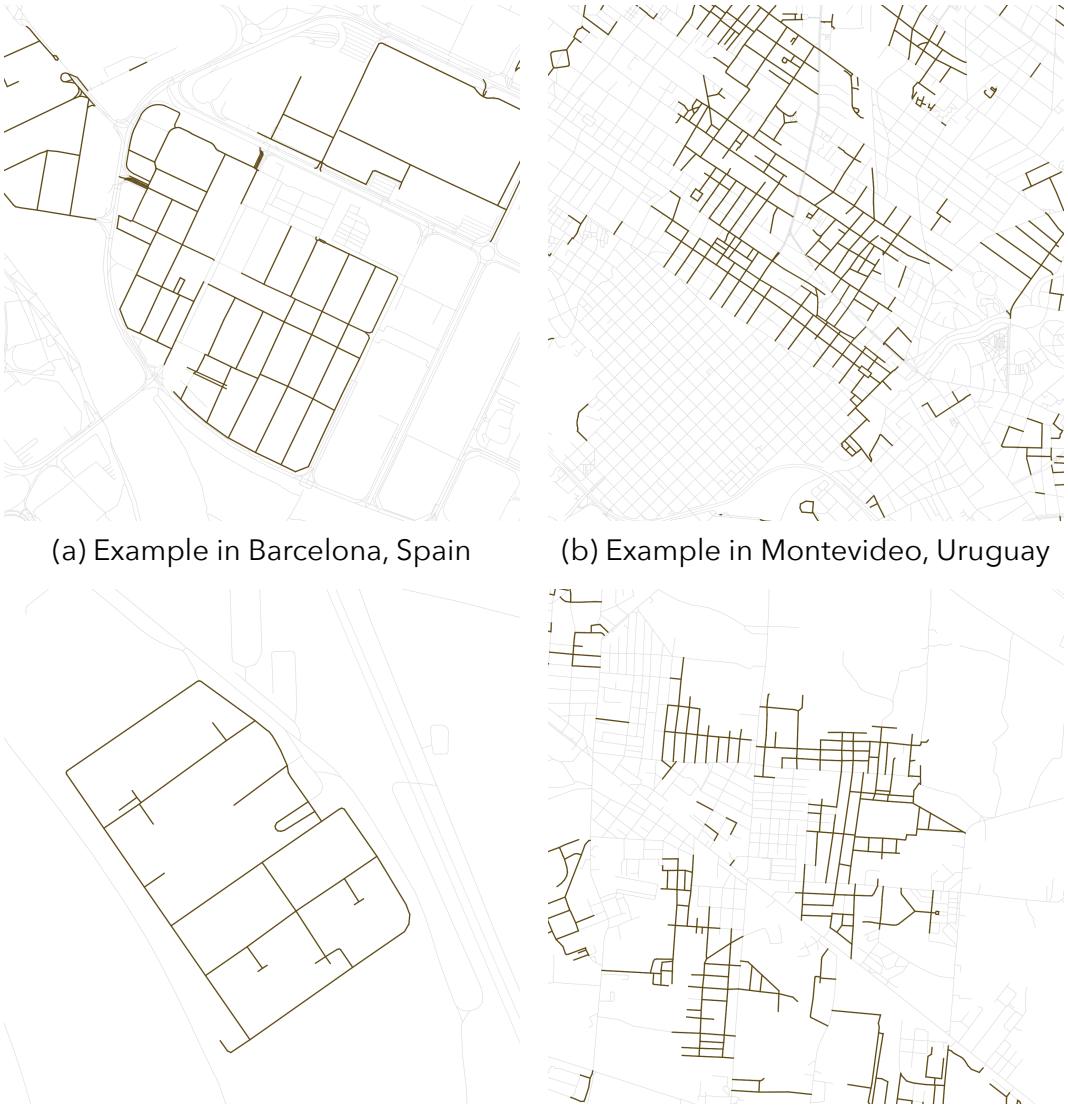
(c) Example in Havana, Cuba

(d) Example in Seoul, South Korea

Figure 60 - Examples of Cluster 0 (Source data from OpenStreetMap contributors (2017))

CLUSTER 1 "IRREGULAR GRID"

Cluster 1 encompasses 14.1% of all analyzed road segments (by segment length). Of all analyzed cities, Beijing, China has with 32.4% the highest percentage of this cluster. Z-scores (Appendix E) show significantly higher right neighbor distance, and significantly lower angle entropy, stretch curvilinearity and right neighbor angle deviation than average. Continuity is only slightly higher than average. Visual inspection shows orthogonal street patterns with moderate but relatively consistent spacing.



(a) Example in Barcelona, Spain (b) Example in Montevideo, Uruguay

(c) Example in Thessaloniki, Greece (d) Example in Havana, Cuba

Figure 61 - Examples of Cluster 1 (Source data from OpenStreetMap contributors (2017))

CLUSTER 2 "INTERUPTED GRID"

Cluster 2 encompasses 13.6% of all analyzed road segments (by segment length). Of all analyzed cities, Chandigarh, India has with 35.0% the highest percentage of this cluster. Z-scores (Appendix E) show very significantly lower angle entropy, stretch curvilinearity and right neighbor angle deviation than average and moderately lower continuity, right neighbor distance, and section length standard deviation than average. Visual inspection shows gridded street patterns with interrupted streets (as suggested by the low continuity). Some outliers were spotted where the pattern did not appear fully orthogonal, for example Figure 62d.

(a) Example in Dakar, Senegal (b) Example in Havana, Cuba

(c) Example in Thessaloniki, Greece (d) Example in Montevideo, Uruguay

Figure 62 - Examples of Cluster 2 (Source data from OpenStreetMap contributors (2017))

CLUSTER 3 "HIGHLY CONTINUOUS STREETS"

Cluster 3 encompasses 11.0% of all analyzed road segments (by segment length). Of all analyzed cities, Montevideo, Uruguay has with 34.9% the highest percentage of this cluster. Z-scores (Appendix E) show very significantly higher right neighbor distance and continuity than average, and significantly higher section length standard deviation. Visual inspection shows very inconsistent patterns, ranging from highly regular grid patterns in Figure 63a, to long stretches of curved (Figure 63b and Figure 63d) or straight (Figure 63c) roads, to parallel curved (Figure 63e) and irregular orthogonal (Figure 63f) patterns. One thing all these patterns have in common is a high median continuity.

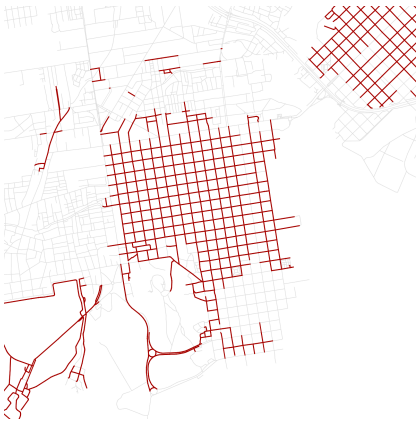
CLUSTER 4 "UNDEFINED 1"

Cluster 4 encompasses 0.02% of all analyzed road segments (by segment length). Of all analyzed cities, Bucharest, Romania has with 0.18% the highest percentage of this cluster. Z-scores (Appendix E) show extremely significantly higher section length standard deviation, and very significantly higher right neighbor distance and continuity than average. Visual inspection yielded only one example in 7 investigated cities, a stretch of road with both long straight segments and slowly curving segments.

Figure 64 - Example of Cluster 4 in Barcelona, Spain (Source data from OpenStreetMap contributors (2017))

CLUSTER 5 "DENSE CORE OR PARKING LOT"

Cluster 5 encompasses 8.1% of all analyzed road segments (by segment length). Of all analyzed cities, Delft, Netherlands has with 36.9% the highest percentage of this cluster. Z-scores (Appendix E) show lower than average scores for all clustering metrics, and significantly lower right neighbor distance and continuity than



(a) Example in Montevideo, Uruguay

(b) Example in Dakar, Senegal

(c) Example in Montevideo, Uruguay

(d) Example in Dakar, Senegal

(e) Example in Washington DC, USA

(f) Example in Thessaloniki, Greece

Figure 63 - Examples of Cluster 3 (Source data from OpenStreetMap contributors (2017))

average. Visual inspection shows dense angular street patterns, often occurring in dense city centers (Figure 65a), but also in parking lots (Figure 65d)

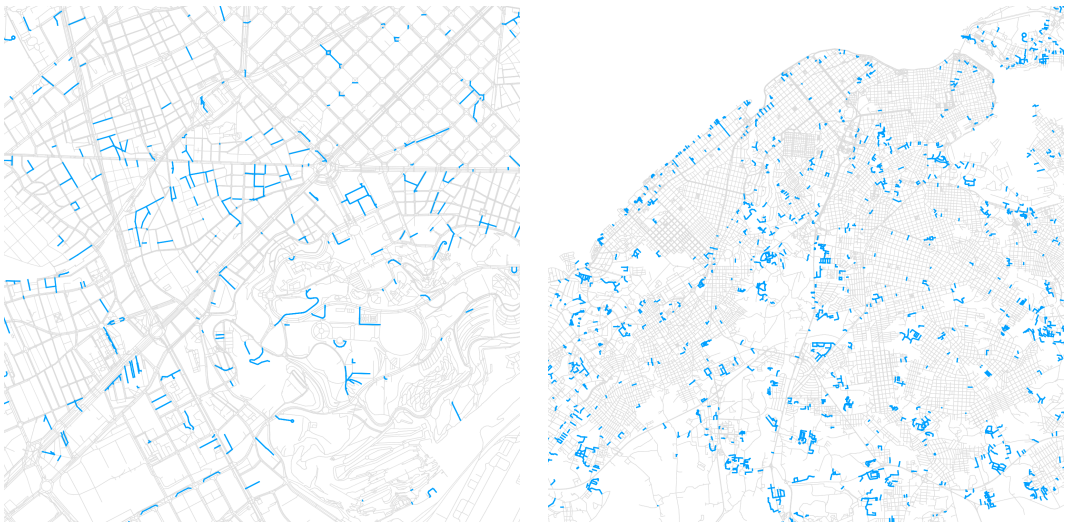
(a) Example in Barcelona, Spain (b) Example in Thessaloniki, Greece

(c) Example in Seoul, South Korea (d) Example in Washington DC, USA

Figure 65 - Examples of Cluster 5 (Source data from OpenStreetMap contributors (2017))

CLUSTER 6 "DISCONNECTED"

Cluster 6 encompasses 11.1% of all analyzed road segments (by segment length). Of all analyzed cities, Hong Kong, China has with 26.6% the highest percentage of this cluster. Z-scores (Appendix E) show significantly lower continuity, lower section length standard deviation, and higher curvilinearity than average. Visual inspection shows scattered segments that are often disconnected from the rest of the network of the same class (they are often only connected to major roads). Especially cities like Hong Kong show this, that have large areas that are mostly connected by major streets.



(a) Example in Barcelona, Spain (b) Example in Havana, Cuba

Figure 66 - Examples of Cluster 6 (Source data from OpenStreetMap contributors (2017))

CLUSTER 7 "REGULAR GRID"

Cluster 7 encompasses 9.0% of all analyzed road segments (by segment length). Of all analyzed cities, Paramaribo, Surinam has with 20.5% the highest percentage of this cluster. Z-scores (Appendix E) show significantly higher continuity, and clearly lower angle entropy and right neighbor angle deviation than average. Visual inspection shows quite regular grid patterns (Figure 67a and Figure 67d) with occasionally more variations in angles (Figure 67b). Outliers were found as seen in figure Figure 67b, possibly due to the choice of weighted median for continuity (the weighted median continuity of this neighborhood would still be high as the small dead end streets have a low weight).

(a) Example in Havana, Cuba (b) Example in Barcelona, Spain

(c) Example in Washington DC, USA (d) Example in Dakar, Senegal

Figure 67 - Examples of Cluster 7 (Source data from OpenStreetMap contributors (2017))

CLUSTER 8 "CURVY ROADS"

Cluster 8 encompasses 1.1% of all analyzed road segments (by segment length). Of all analyzed cities, Sao Paulo, Brazil has with 3.77% the highest percentage of this cluster. Z-scores (Appendix E) show significantly higher angle entropy and right neighbor angle deviation and clearly higher stretch curvilinearity than average. Visual inspection shows highly curved streets, sometimes mixed with straight roads. This specific cluster was not often encountered in the 7 inspected cities, even though they contain many curvy rural roads in the outskirts.

(a) Example in Havana, Cuba (b) Example in Thessaloniki, Greece

Figure 68 - Examples of Cluster 8 (Source data from OpenStreetMap contributors (2017))

CLUSTER 9 "UNDEFINED 2"

Cluster 9 encompasses 0.01% of all analyzed road segments (by segment length). Of all analyzed cities, Lisbon, Portugal has with 0.34% the highest percentage of this cluster. Z-scores (Appendix E) show extremely significantly higher section length standard deviation, and significantly higher angle entropy and right neighbor distance than average. Visual inspection only yielded one example in 7 analyzed cities, which showed two small pieces of erratically curved stretches.

Figure 69 - Example of Cluster 9 in Seoul, South Korea (Source data from OpenStreetMap contributors (2017))

CLUSTER 10 "SPACIOUS WINDING ROADS"

Cluster 10 encompasses 12.6% of all analyzed road segments (by segment length). Of all analyzed cities, Lagos, Nigeria has with 38.1% the highest percentage of this cluster. Z-scores (Appendix E) show significantly higher angle entropy, right neighbor distance, and stretch curvilinearity, and higher right neighbor distance and section length standard deviation than average. Visual inspection shows winding streets with relatively much space between the streets.

(a) Example in Washington DC, USA

(b) Example in Seoul, South Korea

(c) Example in Dakar, Senegal

(d) Example in Thessaloniki, Greece

Figure 70 - Examples of Cluster 10 (Source data from OpenStreetMap contributors (2017))

CLUSTER 11 "SCATTERED AND ORTHOGONAL ROADS"

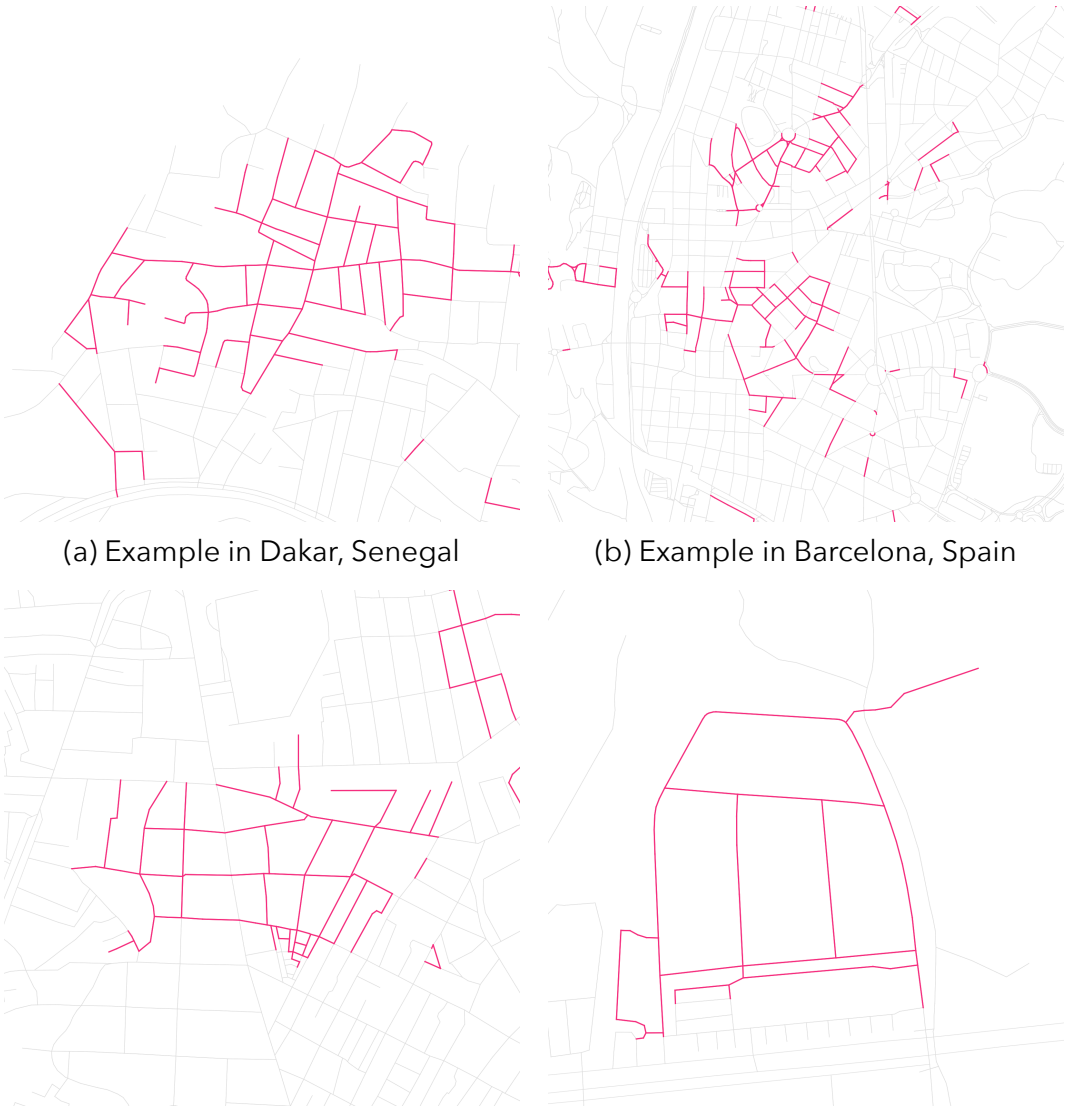
Cluster 11 encompasses 4.8% of all analyzed road segments (by segment length). From all analyzed cities, Bangkok, Thailand, has with 10.3% the highest percentage of this cluster. Z-scores (Appendix E) show higher angle entropy and lower continuity than average. Visual inspection shows small patches of quite orthogonal road patterns scattered throughout the city, often not very connected to the rest of the road pattern or only connected through major streets.

(a) Example in Havana, Cuba (b) Example in Washington DC, USA

Figure 71 - Examples of Cluster 11 (Source data from OpenStreetMap contributors (2017))

CLUSTER 12 "STRUCTURED ANGULAR"

Cluster 12 encompasses 4.5% of all analyzed road segments (by segment length). Of all analyzed cities, Dakar, Senegal has with 12.4% the highest percentage of this cluster. Z-scores (Appendix E) show higher angle entropy and lower continuity than average. Visual inspection shows angular street patterns quite similar to cluster 0 but with slightly less curvy roads.



(a) Example in Dakar, Senegal (b) Example in Barcelona, Spain

(c) Example in Seoul, South Korea (d) Example in Washington DC, USA

Figure 72 - Examples of Cluster 12 (Source data from OpenStreetMap contributors (2017))

Figure 73 shows the dendrogram of the resulting clustering (by Euclidian distance of the cluster centroids), following the same methodology as Fleischmann et al. (2022). Three distinct groups are visible. Further investigation of these groups show that the ambiguous cluster 3 is grouped with the discardable clusters 4 and 9. The scattered clusters 6 and 9 are grouped with the angular cluster 5 and the highly gridded cluster 2. The third group contains both very orthogonal grids, angular networks, and organic networks. Cluster 5 is in a different group than 0 and 12, which are visually quite similar. Therefore, I concluded that the dendrogram grouping does not help with the comprehensibility of the clusters and only represents Euclidian distance in a higher dimensional space without

revealing interesting groupings. Grouping and color-coding the clusters by dendrogram branch, as done by Fleischmann et al. (2022), was not implemented because of this rejection of the comprehensibility of the dendrogram.

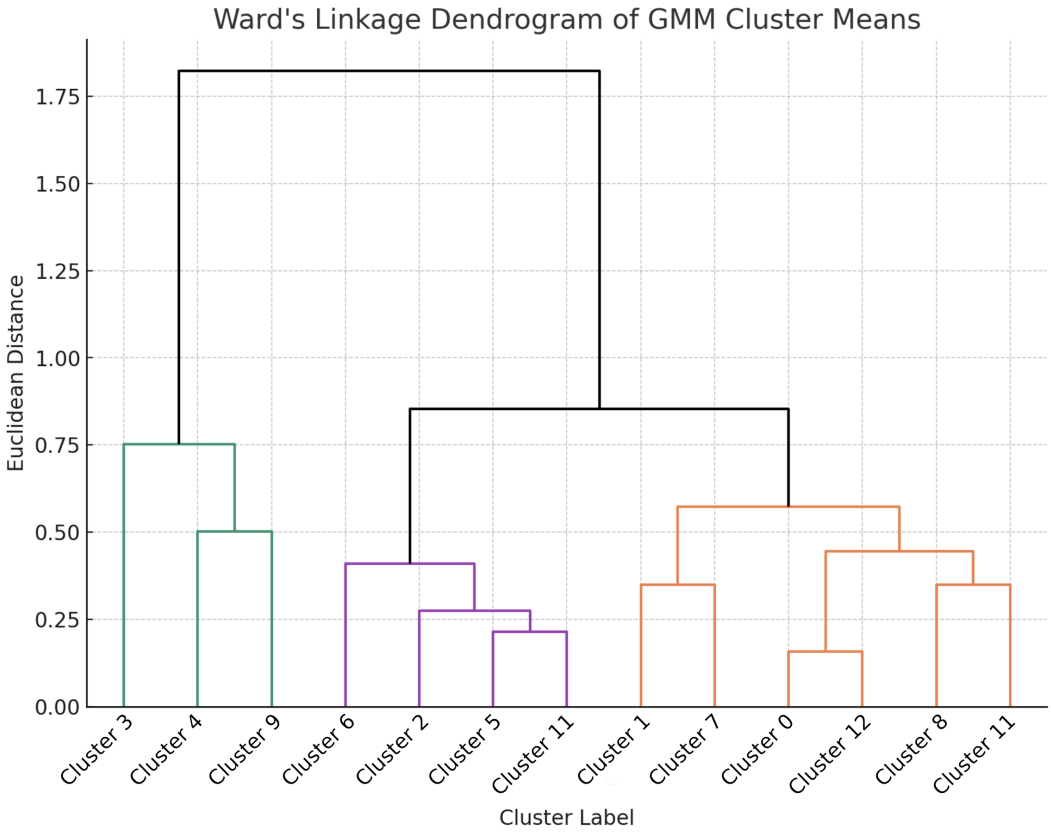


Figure 73 - Ward's linkage dendogram of the means of the computed clusters, showing euclidean distance on the Y axis.

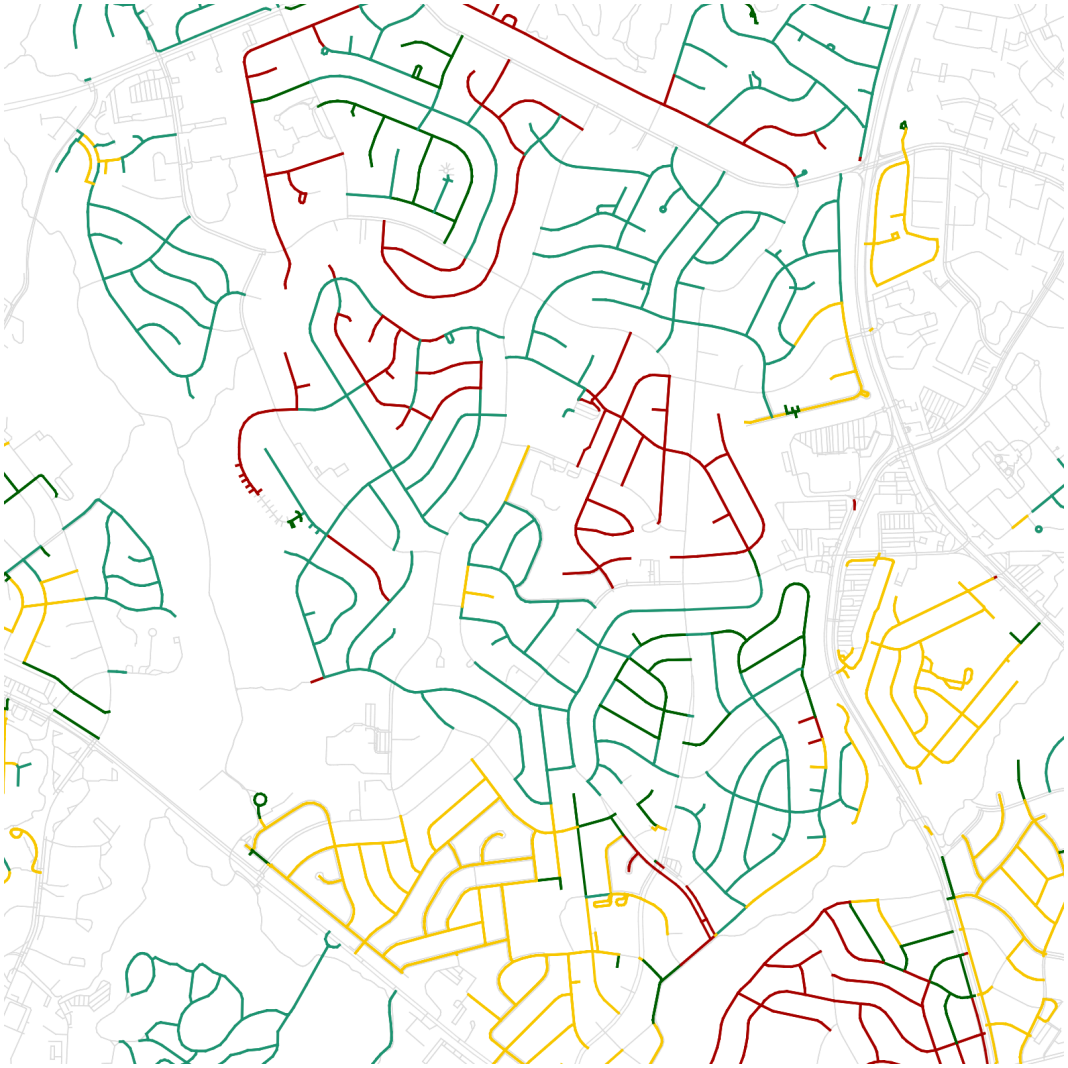


Figure 74 - Example showing three different clusters beig assigned to an area which appears to have a quite homogenous road pattern. (Source data from OpenStreetMap contributors (2017))

6.1.2 Building Classification

All attributes, including local and neighborhood versions, were used for classification input. The only ones left out are references, ids, and other non significant attributes like: `confidence`, `geometry`, `source`, and `name`.

Feature importance scores reveal which input features are the most and least important for the accuracy of the classification. Computed importances (Appendix F) reveal the land use category as the most distinguishing feature. Out of the 20 most important features, 17 describe the aggregated neighborhood. From the 3 non-neighborhood metrics, land use category and subtype often directly

relate to a specific building class (i.e. industrial land use to industrial building or apartment subtype to apartment building). The last one, "covered area" still describes a relationship with the directly neighboring buildings, so it does not describe the building in isolation. From this, it can be concluded that the classification of building type seems to be much more dependent on neighborhood metrics than on individual building metrics.

Training took 4 minutes with 1000 iterations, a learning rate of 0.1, and a tree depth of 6 (the default values).

Figure 75a shows the results of the trained model applied to the city of Prague. As can be seen, clear areas emerge from the resulting pattern. Figure 75b shows the clustering result from Fleischmann et al. (2022), colored according to the unnamed clusters. Clear similarities are visible, with the *filled block* class corresponding closely with the dark red cluster in the historic center and the *perimeter block* class closely corresponding with the blue cluster surrounding the historic center. The *apartments* class has a significant overlap with the yellow cluster, the *detached* class with the light pink cluster, and the *industrial* class with the grey cluster. The classification results (Figure 75a) show a clear difference in the ability to identify single or small groups of buildings within a larger area of another class. Figure 76 shows how the *complex* class is assigned to the central train station (1), the Národní Muzeum (2), and the Prague University of Economics and Business (3), whereas the clustering approach from Fleischmann et al. (2022) assigns these to the clusters of the larger surrounding urban tissue. This has both advantages and disadvantages. The classification approach is able to reflect the presence of unique buildings or areas that have a mix of two or more building classes. The clustering approach is able to create more contiguous areas of urban tissue instead of classifying individual buildings.

(a) Classification results. (Source data from OpenStreetMap contributors (2017), Overture Maps (2023))

(b) Combined clustering results (Image from Fleischmann et al. (2022))

Figure 75 - Comparison between classification results from this thesis (a) and clustering results from Fleischmann et al. (2022) (b).

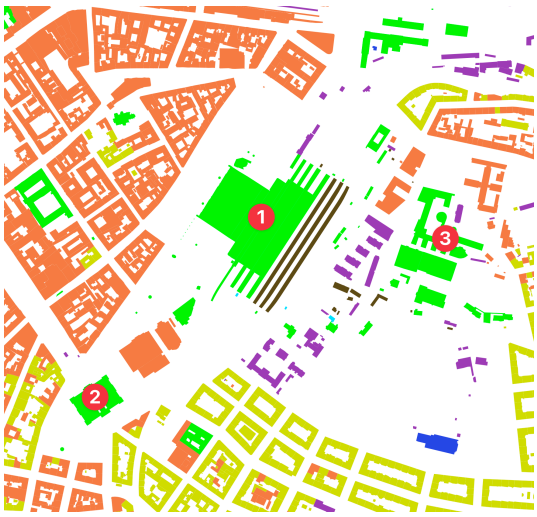


Figure 76 - Examples of classification results for the `complex` typology, highlighting the central train station (1), the Národní Muzeum (2) and the Prague University of Economics and Business (3). (Source data from OpenStreetMap contributors (2017), Overture Maps (2023))

Earlier classification runs included more classes, namely "semi-detached", "agricultural", and "tower". Semi-detached was discarded because creating an accurate ground truth dataset was difficult, detached houses often included attached garages and outbuildings, and corner buildings of terraced housing were getting classified as semi-detached. "Agricultural" was discarded because only Antwerp out of the ground truth city contained greenhouses, industrial and commercial buildings were getting classified as agricultural, and semantically agricultural being marked as industrial was deemed acceptable. Finally, the tower class was discarded because it often did not classify very tall buildings as towers, while often classifying detached houses in the outskirts of cities incorrectly as towers.

Another observed effect is the impact of building footprint data quality on classification results. Figure 77a shows on satellite imagery that the area has a uniform building pattern, but Figure 77b shows the difference OpenStreetMap footprints in the top left and AI generated footprints in the other areas have on classification performance. Many of the blocks get assigned to irregular block, while supposedly they are filled blocks.

(a) Real world area in San Jose, Costa Rica (Airbus, 2024; Google, 2024; Google Earth, 2024; Maxar Technologies, 2024)

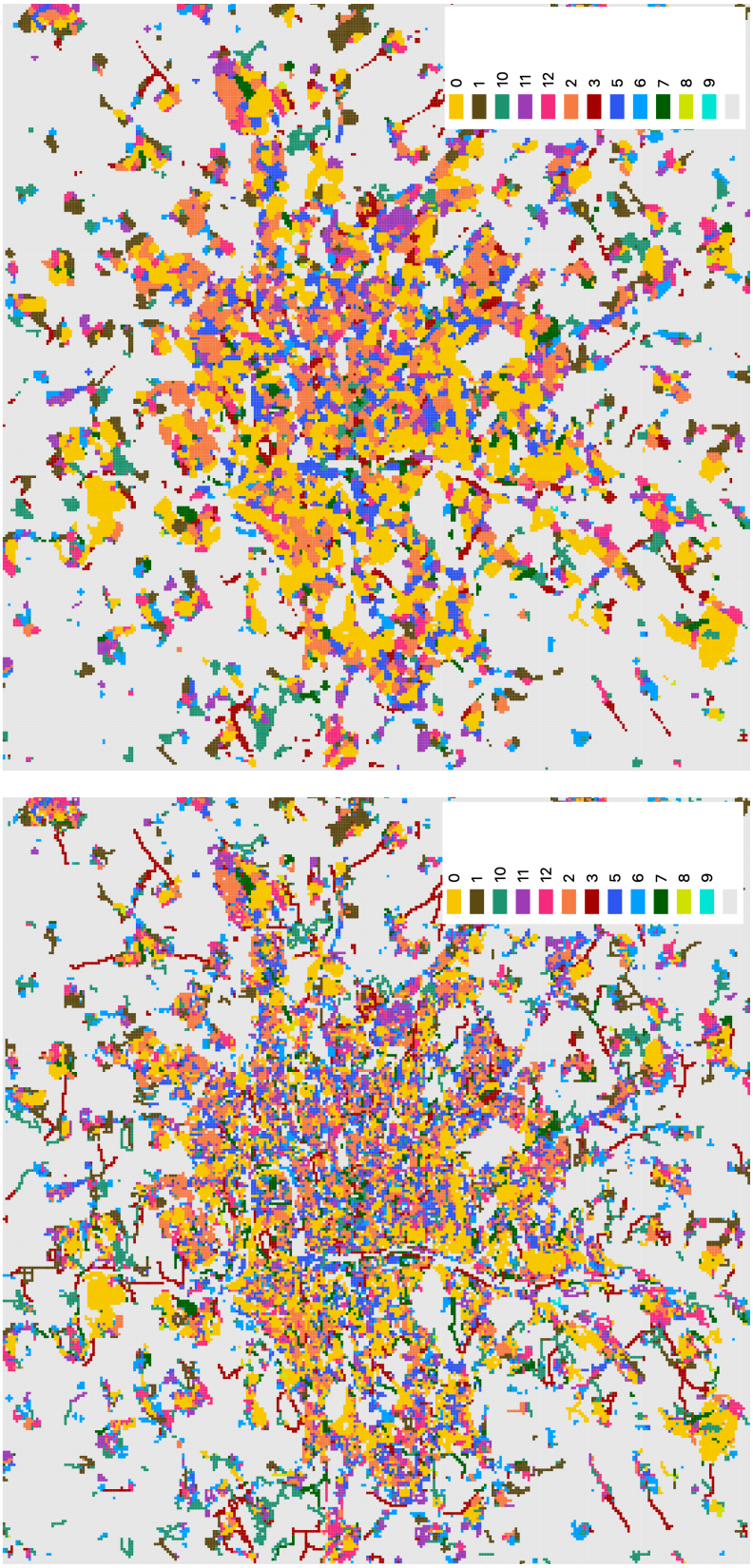
(b) Classified building footprints, with clearly visible effect of inaccurate footprints being assigned a different class. (Source data from OpenStreetMap contributors (2017), Overture Maps (2023))

Figure 77 - Influence of building footprint data quality on classification.

6.2 Encoding

6.2.1 City Templates

Typology grids and convoluted typology grids were created for all 43 analyzed cities. Figure 78a shows an example output typology grid for the roads layer, and Figure 78b shows the same typology grid with the 3×3 mode kernel applied. The convolution process makes the typology grid more readable and reduced noise. However, across all cities, the mode kernel caused 17.5% of all grid cells to change value.



(a) Before convolution.

(b) After convolution with 3×3 mode kernel.

Figure 78 - Example of resulting minor roads *typology grid*

Figure 79 shows a close-up of the same convoluted typology grid as Figure 78b with as overlay the original minor streets. The organic pattern in the historic center is mostly assigned to cluster 0 and 5 and the more orthogonal streets around it to cluster 2. These patterns show a good characterization of the urban tissue. What can also be seen however is the presence of many small patches of other clusters, including small parts of the orthogonal patterns being assigned to cluster 3, 7, and 10. Additionally, tiny patches from cluster 6 and 11 are present. The presence of these small patches harms the comprehensibility, possibly have a negative impact of generation of new grids based on this template, and even if reproduced in a similar pattern might cause sudden changes of pattern in generated road patterns.

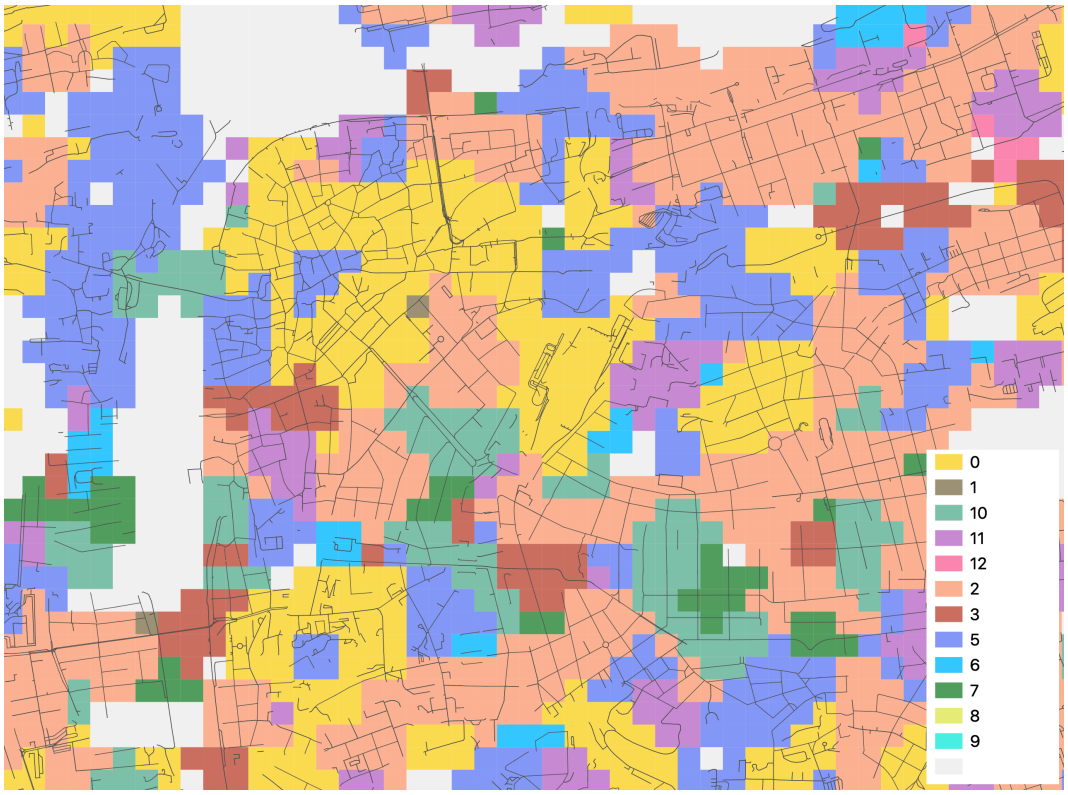


Figure 79 - Example of the minor roads *typology grid of Prague, zoomed in on the city center. The* minor roads *layer is added as a gray overlay. (Road source data from OpenStreetMap contributors (2017))*

Figure 80 shows an example output typology grid for the buildings layer, with mode kernel applied. The buildings grid shows comparatively more contiguous areas than the minor roads grid, with a clear gradient from city blocks, to apartments, to detached buildings and industrial buildings radiating out from the city center.

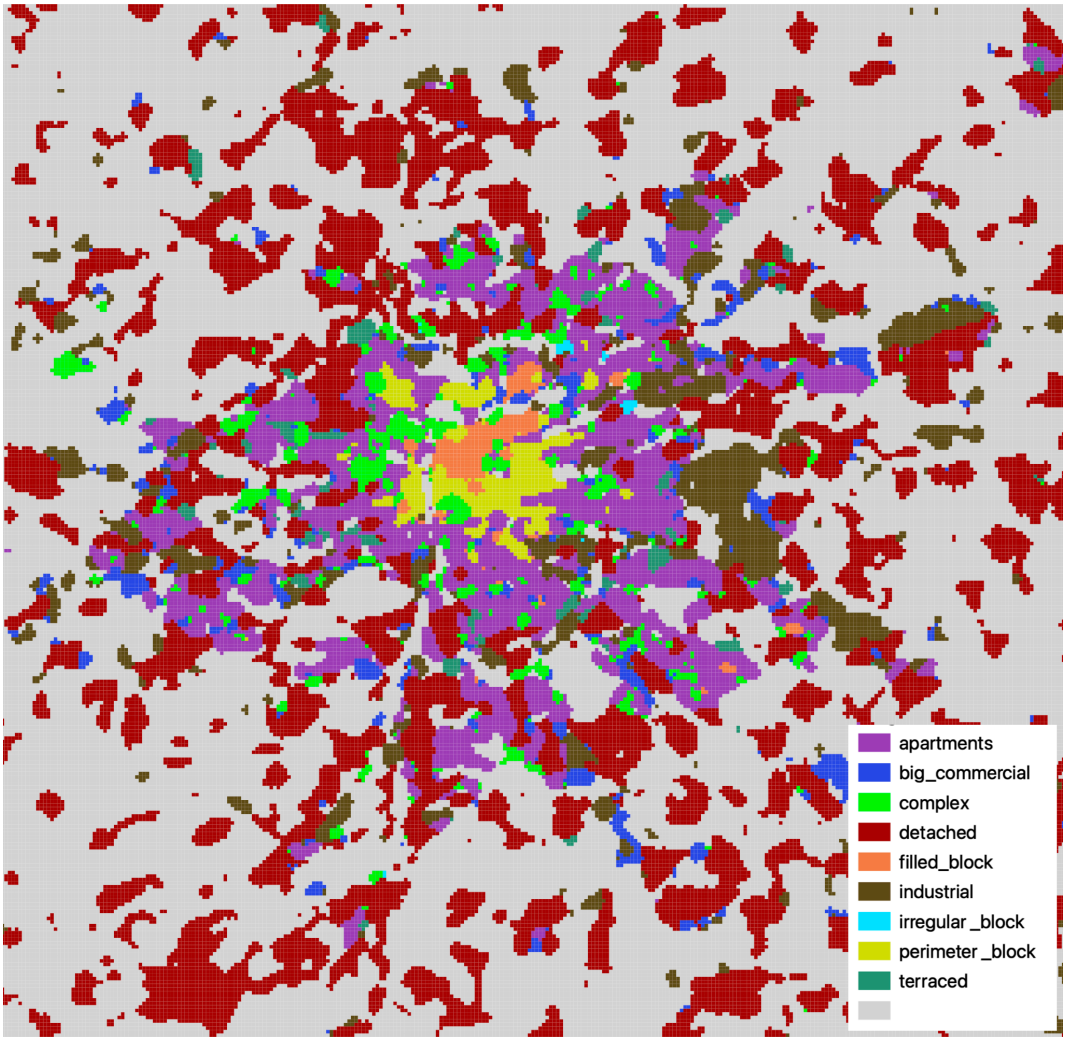


Figure 80 - Example of building *typology grid of the city of Prague after convolution.*

Closer inspection in Figure 81 shows relatively clear and contiguous areas that seem consistent with the buildings within them. Building blocks in the *perimeter block* region show clearly visible courtyards, and the typology immediately switches to *apartments* at the transition from blocks to detached buildings. The areas marked as *complex* sometimes reveal interesting features, for example, the Prague Castle area (1), university area (2), and station area (3). Other smaller patches of the *complex* class do not seem to reveal specific features.

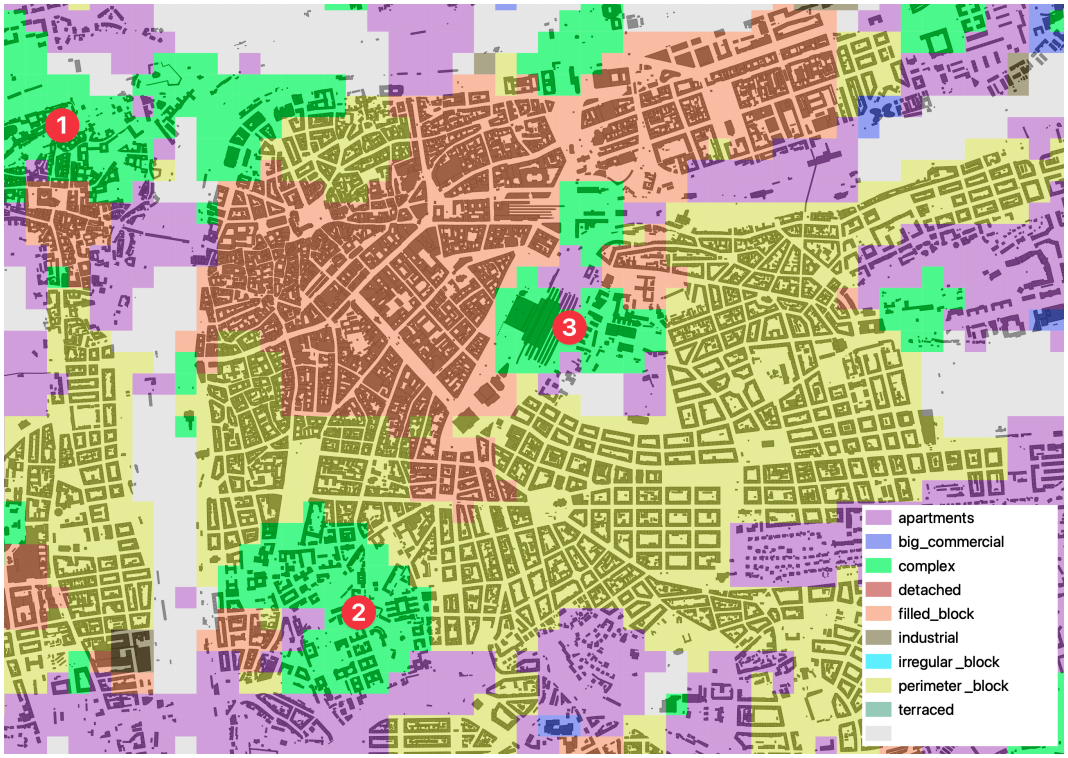


Figure 81 - Example of the building *typology grid of the city of Prague, zoomed in to the city center. The* buildings *layer is added as a gray overlay. (Source data from OpenStreetMap contributors (2017), Overture Maps (2023))*

Besides the generation of new typology grids for morphology-based city generation, the typology grids have the potential to be used in other research applications. Figure 82 and Figure 83 are an example of such an application, showing the distribution of minor road typology and building typology, respectively, for all analyzed cities. The addition of extra parameters on the typology grid can broaden the possibilities for analyses.

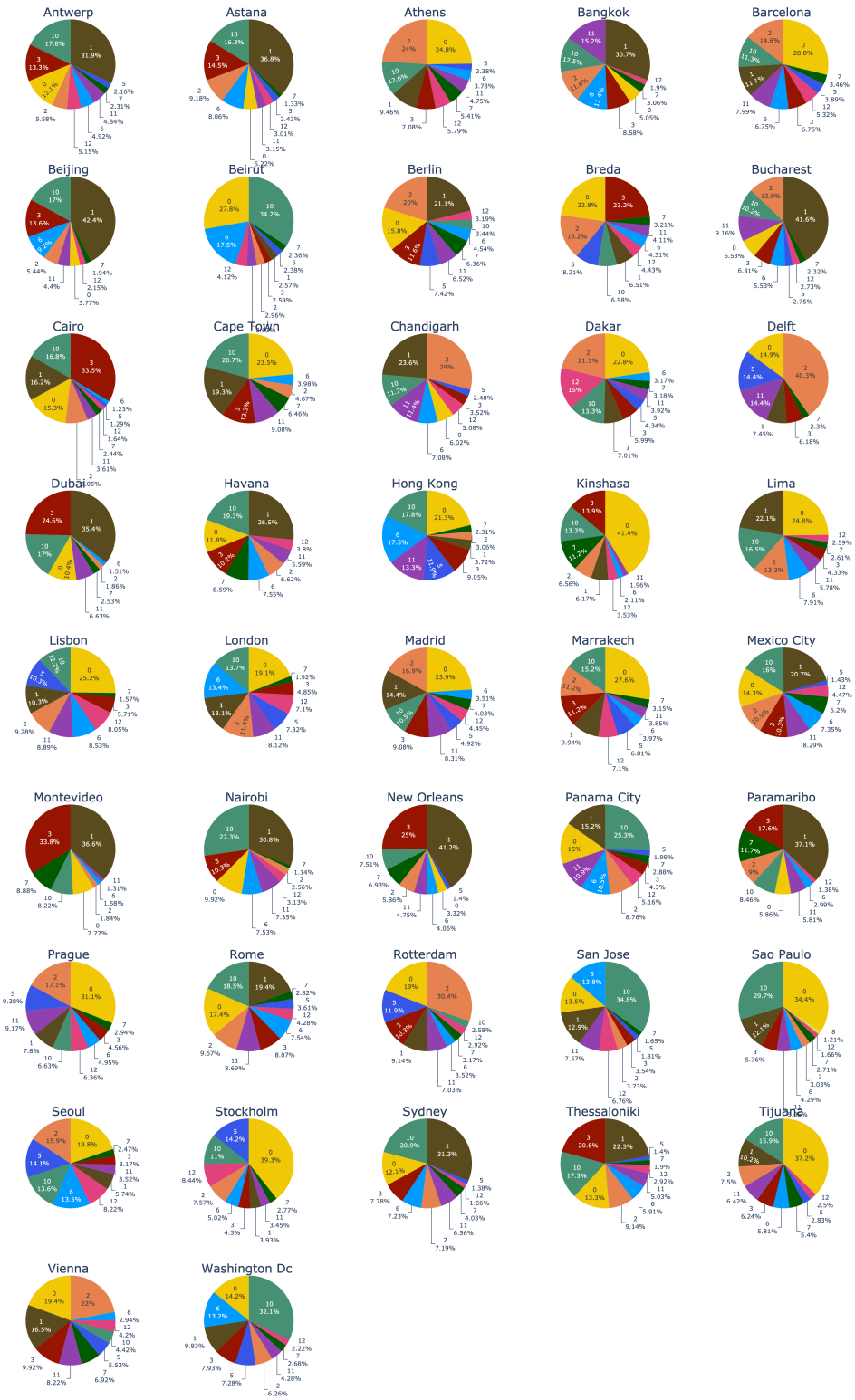


Figure 82 - Percentage of minor roads *typology for the typology grid of each analyzed city. Clusters with less than 1% have been left out.*

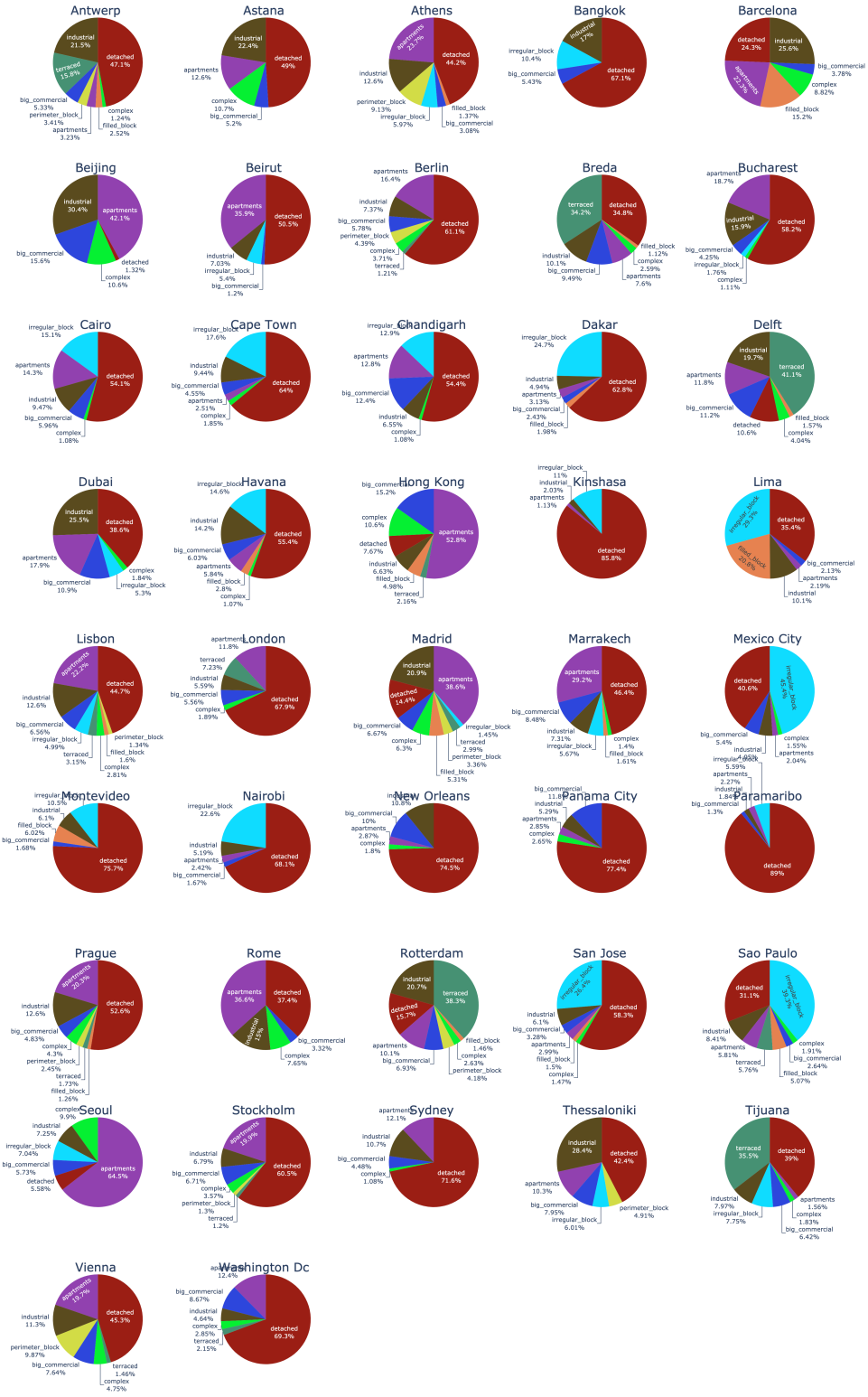


Figure 83 - Percentage of buildings *typology for the typology grid of each analyzed city. Clusters with less than 1% have been left out.*

6.3 Generation

6.3.1 Grids

For performance reasons, simulated annealing was performed at lower resolutions. Two state modification methods were investigated, firstly *random swap*. A resolution of $\frac{1}{9}$th results in a grid with 300×300 meter cells; roughly 7000 variations can be attempted per second of a grid the size of the small city Breda (26 thousand cells). Figure 85 shows convergence at around 15:46, corresponding to roughly 1.3 million iterations. Figure 84d shows the state of the grid shortly after convergence, showing a realistic pattern suddenly appearing from the apparent random pattern of Figure 84c. Another 1 million iterations show marginal improvements, for example the increase in core area for the big terraced housing area (pale green).

Figure 84a shows the starting condition, and Figure 84f shows the final output after 2.5 million iterations with a final similarity of 95%.

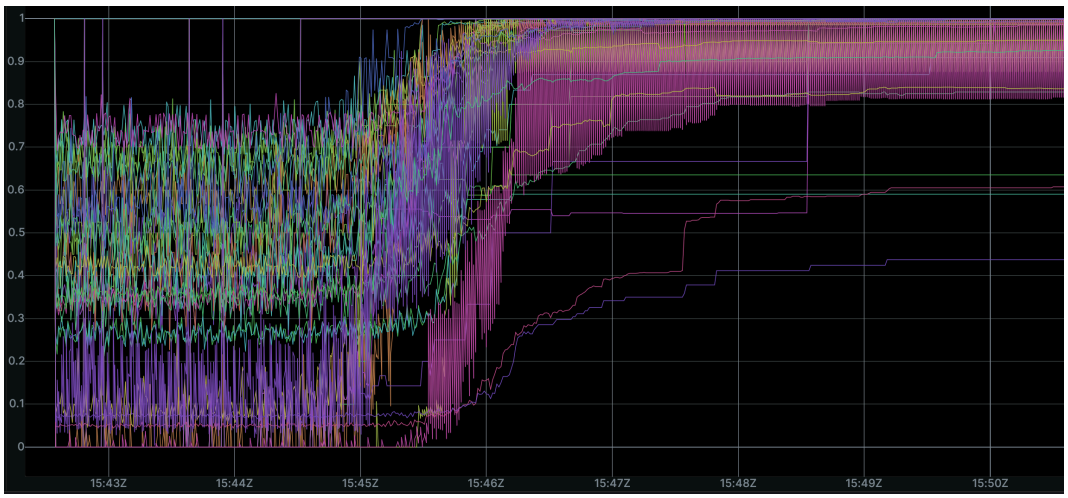
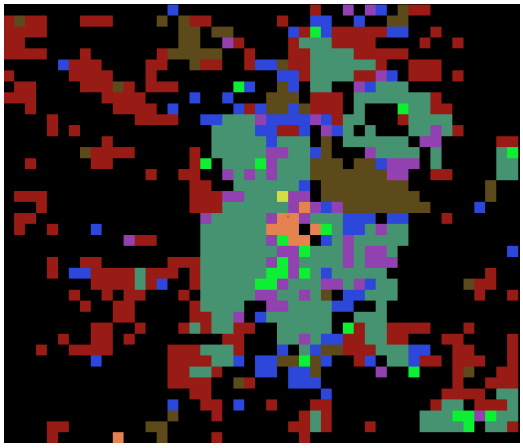


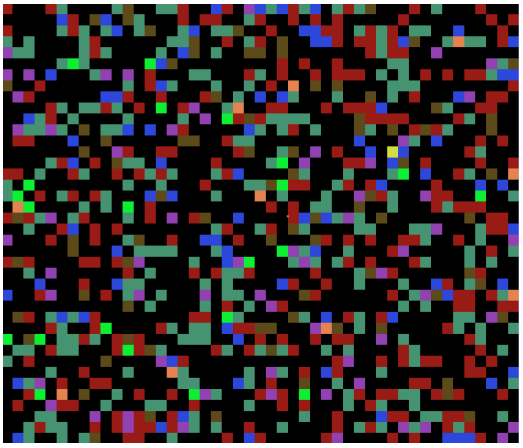
Figure 85 - Statistics of all similarity metrics (0 to 1) for the simulated annealing process.

Statistically speaking, the generated typology grid is very similar to the starting grid. Visual inspection reveals as the main flaw the presence of a lot of empty space or "None" typology around the city center in Figure 84f compared to the starting grid Figure 84a.

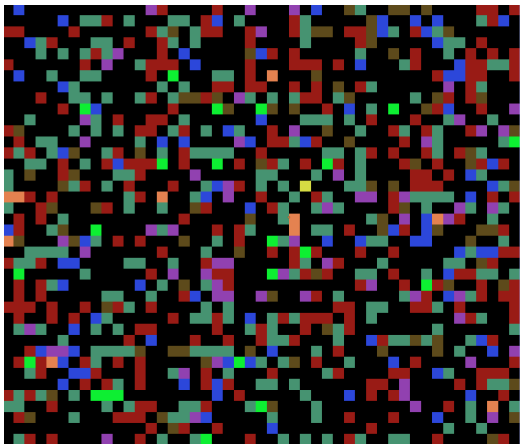
The ability of the simulated annealing process to converge appears to be very much linked with the grid size. Going from $\frac{1}{9}$th resolution to $\frac{1}{4}$th resolution not only slows down the simulation, it also drastically reduces the quality of the results reaching only 63% similarity after 2.5 million iterations. Due to the acceptance criteria being linked with the current temperature, and the cooling



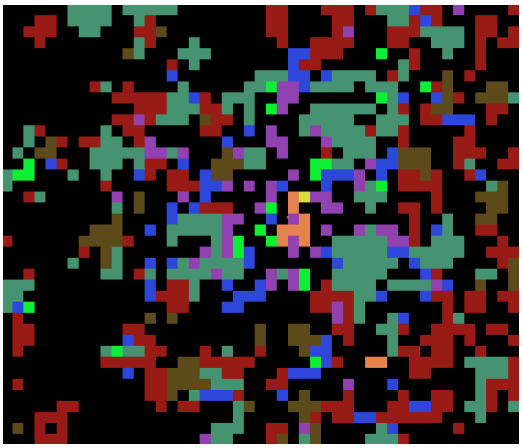
(a) Input building typology grid from Breda (Similarity 100%).



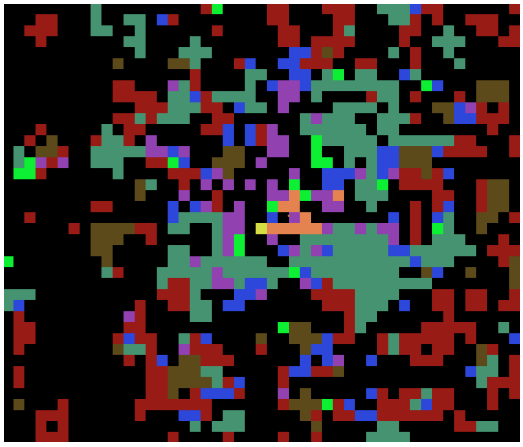
(b) Grid at 500 thousand iterations (Similarity 50%).



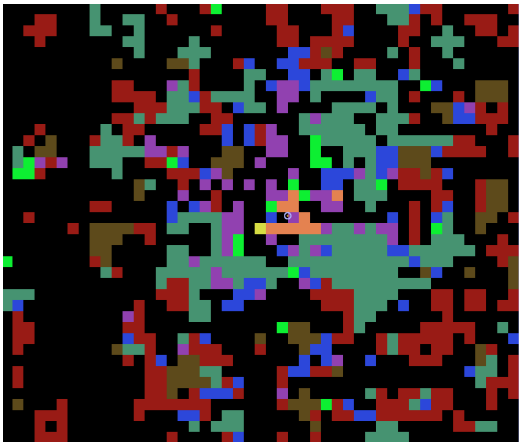
(c) Grid at 1 million iterations (Similarity 64%).



(d) Grid at 1.5 million iterations (Similarity 86%).



(e) Grid at 2 million iterations (Similarity 94%).



(f) Generated building typology grid (Similarity 95%).

Figure 84 - Results from typology grid generation.

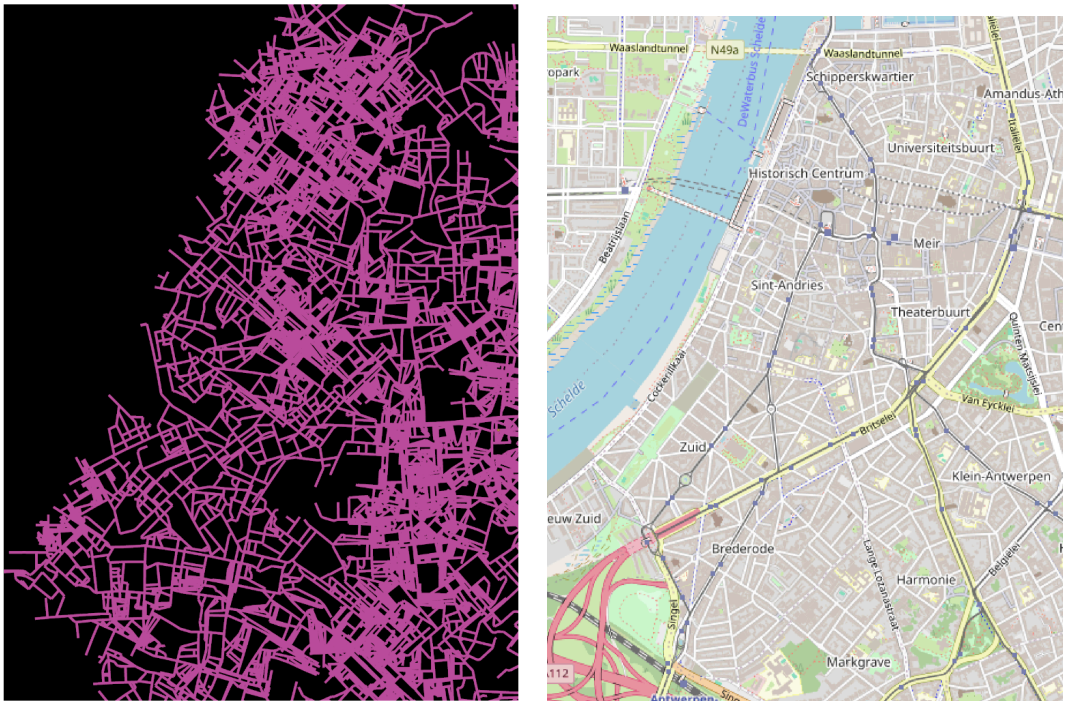
schedule remaining the same, an interesting effect is observed. After roughly 1.5 million iterations, the grid appears to have reached a local minimum, which it only escapes once only to reach another local minimum. The rejection rate is close to 100% for the remaining 1 million iterations.

The *random walk* method where a randomly selected cell is swapped with one if its neighbors (that has a different class) performed worse than the *random swap* method.

In the $\frac{1}{9}$th resolution test it ended up with 78% similarity and in the $\frac{1}{4}$th resolution test, it ended up with 56% similarity. What could be contributing to this is the inability of this method to escape local minima. Additionally, since influencing metrics like distance to the city center are compared in bins, small steps towards or away from the city center will often still fall within the same bin and, therefore, not produce a higher similarity.

6.3.2 Roads System

The performance of the road generation system was excellent, generating 300 thousand segments in mere seconds. Figure 86 shows an example area of the output of road generation.



(a) Generated street pattern with Antwerp's typology grid as input.

(b) Real street pattern of the same area. (OpenStreetMap contributors, 2017)

Figure 86 - Generated street patterns compared to real data.

Figure 86 shows an example of the results of road generation next to the street pattern of the area that was used as an input typology grid. The urban tissue of some areas is similar, but overall, the character of the generation results is quite different from the real street pattern. Especially the area "Historisch Centrum" is vastly different, with a very dense grid pattern generated in the location of the historical center.

Borders between different typologies in the typology grid are visible, with sudden changes in density as a result. However, even though the transition strategy used here is very simple (no transition strategy, it switches to a different typology as soon as the segment enters a typology grid cell of another typology), the underlying typology grid pattern is not jarringly visible. The visibility between changing from one to another typology seems more linked to a big difference in density between the two typologies.

It should be noted that road generation results shown here are based purely on the aggregated data in the typology template of the cluster across the whole city. This was identified in Section 4.1.1 as an oversimplification, with as the proposed solution the parameters of the typology grid cell. Since the typology grid cell parameters are out of scope for this thesis, the road generation will not be evaluated based on how similar a generated area is to a real life area. Instead, each cluster is analyzed separately to assess if the road generation method is able to generate the general character that said cluster encompasses.

Each road cluster will be generated based on 3 different local typology templates of Thessaloniki, Montevideo, and Havana respectively. These results can then be compared with the analyzed clusters in Figure 86, as these 3 cities were among the 7 cities analyzed in that section.

CLUSTER 0 "ANGULAR STREETS"

Generation of typology 0 seems to lead to wildly varying results, with Figure 87a showing a high amount of curvy streets, Figure 87b with a very orthogonal pattern, (reflecting the orthogonal character of Montevideo). and Figure 87c most closely resembling the "angular streets" of cluster 0. Both Figure 87a and Figure 87b look quite realistic, while Figure 87c shows some unrealistically small city blocks.

(a) Based on Thessaloniki, Greece (b) Based on Montevideo, Uruguay (c) Based on Havana, Cuba

Figure 87 - Road patterns generated based on local cluster 0 typology templates from different cities.

CLUSTER 1 "IRREGULAR GRID"

Generation of typology 1 also shows variation, with Figure 88a being quite distinct from the "irregular grids" in Figure 88b and Figure 88c. Figure 88a shows some unrealistic angles and sliver city blocks.

(a) Based on Thessaloniki, Greece (b) Based on Montevideo, Uruguay (c) Based on Havana, Cuba

Figure 88 - Road patterns generated based on local cluster 1 typology templates from different cities.

CLUSTER 2 "INTERUPTED GRID"

Cluster 2 generation results seem to capture the "interrupted grid" character up to a certain degree. Figure 89a shows oriented grids that are interrupted upon a change of orientation. Figure 89c shows a very cardinal version, with more interruptions than the "irregular grid" from Figure 88c. Figure 89b surprisingly is the least orthogonal of all 3 patterns, even though Montevideo consists of mostly very orthogonal patterns.

(a) Based on Thessaloniki, Greece (b) Based on Montevideo, Uruguay (c) Based on Havana, Cuba

Figure 89 - Road patterns generated based on local cluster 2 typology templates from different cities.

Clusters 3 and 4 are left out intentionally, as they were deemed invalid.

CLUSTER 5 "DENSE CORE OR PARKING LOT"

Generation results for typology 5 seem very consistent across the 3 analyzed cities. In all cases, they appear more curvilinear than the patterns they are based on, with big variations in density.

(a) Based on Thessaloniki, Greece (b) Based on Montevideo, Uruguay (c) Based on Havana, Cuba

Figure 90 - Road patterns generated based on local cluster 5 typology templates from different cities.

Cluster 6 is left out intentionally, as it was deemed invalid.

CLUSTER 7 "REGULAR GRID"

Generation of typology 7 seems to resemble the character of the analyzed clusters, with orthogonal patterns and seemingly a high median continuity. Figure 91a shows the lowest orthogonality, with occasional curvilinear roads,

while Figure 91b and Figure 91c show a more consistent orthogonal pattern. This corresponds with the presence of consistent grids in these cities.

(a) Based on Thessaloniki, Greece (b) Based on Montevideo, Uruguay (c) Based on Havana, Cuba

Figure 91 - Road patterns generated based on local cluster 7 typology templates from different cities.

Clusters 8 and 9 are left out intentionally, as 8 occurs only sporadically, and 9 was deemed invalid.

CLUSTER 10 "SPACIOUS WINDING ROADS"

Generation of typology 10 yields interesting results. Figure 92a shows a very high probability of dead ends, resulting in quick termination of the road generation process in all 5 generation runs. Figure 92b starts as a pattern with high cardinality, but then contains more irregular grids within this pattern. Figure 92c most resembles the observed "spacious winding roads" from this typology, although the density is still quite high.

(a) Based on Thessaloniki, Greece (b) Based on Montevideo, Uruguay (c) Based on Havana, Cuba

Figure 92 - Road patterns generated based on local cluster 10 typology templates from different cities.

Cluster 11 is left out intentionally, as it was deemed invalid.

CLUSTER 12 "STRUCTURED ANGULAR"

Results from generating typology 12 are quite consistent across the 3 cities and seem to resemble the character of the analyzed clusters but with higher curvilinearity. Patterns seem to resemble realistic organic patterns, and would also be a good fit for cluster 5.

(a) Based on Thessaloniki, Greece

(b) Based on Montevideo, Uruguay

(c) Based on Havana, Cuba

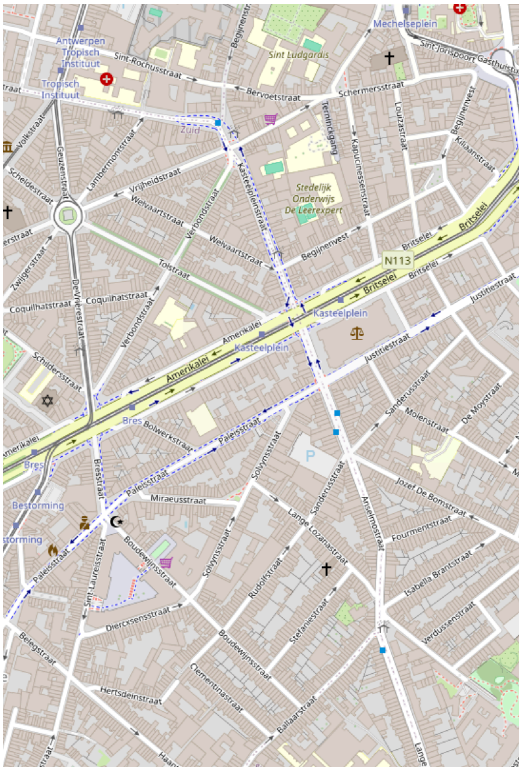
Figure 93 - Road patterns generated based on local cluster 12 typology templates from different cities.

6.3.3 Buildings

The performance of building generation was also excellent, generating 178 thousand buildings (and many more rejected buildings due to intersection checks) in less than a minute.

Since the implemented building generation methods are mostly meant as proof of concept and have not been developed to create highly realistic results, I will not do a deep dive into the realism of the generated buildings. Figure 94a shows that borders between different typologies are very sudden, whereas the real-life area shown in Figure 94b has a slower transition from one to the other type. Generation results, as implemented for this thesis, are able to show a change in urban tissue, but not in a realistic manner. The typology grid combined with proof of concept building generation methods do, however, provide a good starting point to develop a realistic implementation.

(a) Generated building footprints based on `filled_block` (top and right) and `perimeter_block` (bottom left) typologies in fictional street patterns.



(b) Real street pattern of Antwerp including `filled_block` (top) and `perimeter_block` (bottom) areas. (OpenStreetMap contributors, 2017)

Figure 94 - Generated building footprints compared to real data.

7 Discussion

7.1 Concept

The representation of urban tissues as distinct areas within the typology grid can be seen as a comprehensive simplification of a complex real-world phenomenon. This simplification is both a strength and a weakness of this thesis. Its major advantage is the understandability of the approach, and that it creates a uniform framework that applies to any city across the world. For example, Figure 82 shows that the road patterns of a global set of cities can be described in just one page. Also, typology grids are sufficiently understandable that they can be created both programmatically and by hand by a designer. A downside is that it simplifies urban tissue to such a degree that regeneration of a city is challenging, as generation is performed with just a small set of typologies based on aggregated data across the whole city or even the whole globe. Figure 86 shows that attempts to generate a road pattern similar to a real-world location are not very successful. The fact that the cluster assigned to the historic center (Cluster 5) describes both dense urban cores and parking lots highlights the importance of high-quality clustering. Currently, the historic center is generated as a dense, elongated, and chaotic grid.

Future research could be done into creating a higher quality set of road pattern typologies, either by improving the clustering methodology or by attempting a different strategy, like using supervised learning on an expert-created set of global typologies.

7.2 Analysis

This thesis had a clear focus on reproducibility of the results, and making it easy to generate results on unseen cities. This was achieved in the form of the `citypy` command line tool, which can reproduce all results shown in the analysis and encode step of this thesis with a few simple commands. A downside of the current implementation is the time and space complexity, where analyzing a major metropolis can take hundreds of hours and up to 100GB of RAM, meaning that professional computing power is required.

A strength of this research is the sheer amount of data crunched to produce the results, with hundreds of millions of road segments and building footprints being individually analyzed and contextualized.

7.2.1 Metrics

A clear contribution of this thesis is the introduction of many road system metrics that are either novel or built upon existing metrics. These metrics aim to add to the already existing set of numerical urban morphology. Validating the full

applicability and quality of these metrics is outside the scope of this thesis and would need further research.

7.2.2 Contextualizing

Analyzing the effect of the contextualizing method on the quality of the clustering and classification is a difficult task. The road contextualizing method was developed to help detect distinct borders between two differing typologies. However, results show many sudden changes in typology, resulting in scattered patches and a noisy typology grid. Additionally, both cluster 6 and cluster 9 seem to specifically detect small patches of roads that only have a small "neighborhood" due to the contextualizing method. Often, these patches are only connected to major roads, which causes the neighborhood to stop growing. Further research is needed in the effect of contextualizing methodologies on the quality of the clustering, also including the use of different aggregation functions like interquartile mean, interquartile range, interdecile Theil index and Simpson's diversity index as used by Fleischmann et al. (2022).

7.2.3 Clustering

The clustering approach aimed to produce a set of global road typologies where such a set did not exist yet. This was achieved based on global data, considering a wide range of street patterns in cities across the world.

The validity of the final set of clusters is, however, debatable. Clusters 1, 2, and 7 appear to identify different types of grids with decent consistency. Cluster 5, if split in two to represent dense urban cores and parking lots seperately, also appears distinct and unambiguous. Clusters 0 and 12 are quite similar, but if combined, they form a clear typology. Cluster 10 is clear and quite distinct, but visual inspection shows many roads that appear to belong to Cluster 10 being assigned a different cluster. Cluster 8 seems quite restrictive, encompassing only 1% of all roads, seemingly identified by highly curvy roads. Clusters 6 and 11 are common but scattered in small patches throughout the cities. They appear to be caused by disconnected networks and low continuity. Neither seems to clearly identify a specific road pattern. Cluster 3 is promising but very inconsistent. It accurately captures the strict grid in Figure 63a but is also assigned to Figure 63d, which looks more like Cluster 10, Figure 63e, which looks more like Cluster 0 or 12, and other starkly different patterns. The culprit here seems to be the continuity metric introduced in this thesis, combined with the weighted median aggregation method used. Investigation of the attributes show a continuity weighted-median of 1 on the segments assigned to cluster 3, and a drastically lower value to directly neighboring segments belonging to a different cluster. Finally, clusters 4 and 9 are completely invalid and only describe less than 0.1%

of all segments. Here, the culprit appears to be the section length standard deviation, with Z-scores ranging as high as 9. This suggests that standard deviation is not an appropriate aggregation method for clustering.

Besides investigating how well each cluster describes a certain road pattern, another important aspect is the quality of detected areas with similar urban tissue. Visual investigation showed that transitions between clusters often happen quickly, even in areas of seemingly similar character. For example, Figure 74 shows 3 different typologies within the same, quite similar area. This is even excluding tiny patches of even different clusters of the small surrounding streets, not shown in this image. There is definitely room for improvement when it comes to these transitions and minimizing these tiny "noise" patches. When it comes to urban tissue analysis, more generalization is required for the bigger overarching patterns to become visible.

All in all, results show potential for higher quality clustering by tweaking the contextualizing methodology and clustering metrics. Future research could be done into other clustering methods, including subspace clustering to identify clusters that only exist in certain feature subspaces. The effect of using different metrics as input for clustering can be tested. Finally, classification instead of clustering for road typologies can also be attempted to compare the quality of the results. The integration of deep learning techniques is promising, as shown by Wang et al. (2024).

7.2.4 Classification

Classifying buildings using supervised machine learning is not novel, but this thesis investigates the applicability of a global set of building typologies on cities worldwide.

Results show potential for this approach. Visual investigation showed accurate results at a smaller scale. At a bigger scale, results revealed individual incorrectly classified buildings and ambiguousness between typologies. Feature importance statistics highlight the importance of including aggregated neighborhood metrics in the classification features. Visual investigation also revealed the impact of data quality on the classification performance, as footprints generated using machine learning lead to many incorrect classifications.

Comparison with results of the combined clustering of Fleischmann et al. (2022) show stark similarities. This highlights the viability of both approaches, as two different methods reached similar results. It also highlights that the combined clustering of Fleischmann et al. is predominantly based on buildings instead of also on street patterns, validating my hypothesis that analyzing each layer separately adds additional value.

A major point of concern with the classification approach is the way test and training data are split. Figure 47e shows ground truth buildings being selected in groups of multiple buildings in the same direct area. Since test and training data is simply split using random selection, it is inevitable that buildings from the same selection area are both used for training and validation. This makes the resulting accuracy metric not valid and might cause overfitting. Implementing a better splitting method is non-trivial, as not all classes exist in every city, so one could not simply use separate cities for training and validation without a significantly bigger ground truth dataset. This causes this thesis not to have a solid validation of the classification performance, leading to the inability to accurately describe how suitable classification is for building typology detection.

Also, feature importances reveal *land use category* as the most important feature, indicating that the availability of land use data has a big impact on the quality of the classification.

All in all, the quality of the results of the current implementation is deemed as suitable for city generation, as aggregated at a smaller scale (in the typology grid), the classification properly captures the urban tissue of the city.

Future research could be done into a more robust classification and creating a better dataset.

7.3 Encoding

This thesis contributes to publicly available urban morphology data by aggregating hundreds of Gigabytes of data into relatively lightweight grid representations and making these publicly available. The complex layout of a city can be captured in one diagram per city layer. The typology grid concept also enables quick and comprehensive analysis, in a format that is suitable for both vector and raster based processing. In this, it has improved usability above the purely vector based approach from Fleischmann et al. (2022), while losing granularity. See Appendix H for an overview of all generated typology grids.

7.4 Generation

7.4.1 Typology Grid Generation

Simulated annealing results in Figure 84 show strong potential for this technique to be used to generate new typology grids. Even though the objective function is just based on the patch area, core area index, and shape index, it still produces a typology grid that plausibly resembles the starting grid. The binning method-

ology seems able to account for complex relationships and could be tested with more binning metrics that incorporate context and city stack constraints.

Another strength of the typology grid methodology for city generation is that it can be edited or even created from scratch by designers or researchers. The comprehensive nature of the grid, consisting of a set of named typologies, makes it easy to understand and adjust.

A big challenge lies in the performance and parameters of the simulated annealing technique. At $\frac{1}{9}$th resolution, convergence occurs after 1.3 million iterations in the second smallest city of the analyzed dataset (Breda is 43$\times$ smaller in area than the largest city in the dataset). The cooling schedule and acceptance criteria need to be adapted for each grid size, as running the same city at $\frac{1}{4}$th resolution with the same parameters does not converge. Even after heavy optimization, speeds could not be increased past 7000 iterations per second on the $\frac{1}{9}$ th resolution version of Breda. Increases in grid size correspond to big decreases in speed. From this can be concluded that simulated annealing will not be able to produce real-time results.

Besides looking into incorporating complex relationships in the objective function, future research could investigate the use of machine learning models to generate typology grids. The `citypy` tool combined with the tensor-like nature of the typology grid and its parameters could be used to generate a big dataset of typology grids across the globe.

Everything considered, the simulated annealing approach combined with the typology grid contributes a morphology-based methodology to existing land-use based city layout generation methods by Groenewegen & Smelik (2009) and Lechner et al. (2006).

7.4.2 Road Generation

Figure 86 shows that, even with local typology templates, the resulting road pattern is quite different than the real data. This could either be attributed to invalid clusters, or to missing further parameterization of the typology grid. Visual inspection of generation results based on a single cluster reveals more plausible results and highlights the differences between different cities. In some cases, the generated pattern seems very different from the typology it is based on. This can either be explained by the typology not being distinct enough, it being incorrectly assigned in said city, areas with wildly varying statistics (like urban cores vs parking lots for cluster 5), and small sample sizes. In terms of realism, the examples also show room for improvement, as they contain big changes in density and sometimes unrealistically small enclosures. All in all, the variation

between the examples highlights the additional value a morphological approach provides for the street generation algorithm of Parish & Müller (2001).

Future research could look at generating road patterns based on manually selected clusters instead of programmatically derived clusters to isolate the quality of the generation from the quality of the clustering. Deep learning approaches could be investigated for determining input parameters for road segment creation to incorporate even more complex relationships. Additionally, different road pattern generation methodologies could be tested, like example-based generation for an entire patch that smoothly transitions to another typology at the patch border.

7.4.3 Building Generation

The building generation methods in this thesis are proof of concepts for their corresponding building classes and are not parameterized at all at the moment. This means that the ability of the building generation system to recreate real patterns is hard to judge based on the results. Future research could attempt inverse procedural modeling or even deep learning-based footprint generation based on the patches on the typology grid.

All in all, procedural generation for both roads and buildings shows promising results. City generation using "black box" artificial intelligence was ruled out of scope for this thesis, but these models have the potential to represent reality more closely (Breiman, 2001) and should therefore be investigated.

8 Conclusion

This thesis introduces the city stack framework and typology grid concept, both from the analysis and generation side.

How can the urban form of real-world cities be captured using publicly available geospatial data?

This thesis introduces a set of new urban morphology metrics aimed at creating a minimal set of metrics to distinguish real-world street patterns, contributing to the urban morphology field. To test this approach, hundreds of millions of road segments across 43 cities worldwide were programmatically analyzed.

Unsupervised learning using a Gradient Mixture Model on this dataset appears to be able to derive a global set of road pattern typologies. The validity of the derived typologies is difficult to establish. Some clusters appear to describe a clear road typology, showing consistent results across multiple cities. Other clusters are either invalid or not clear (Cluster 4, 6, 9, 11). Additionally, the clustering approach sometimes groups patterns together that one would not expect, like dense European city centers and North American parking lot service roads (Cluster 5). Some of the newly introduced metrics appear to create too much separation in feature space, causing visually starkly different patterns to belong to the same cluster (Cluster 3). Future research can investigate improving the quality of the clustering or test out the viability of using supervised learning for road pattern classification.

Supervised learning based on Gradient Boosting appears to be adequately suitable for detecting building typologies for the use case of city generation. The proposed set of building typologies combined with this classification approach produced highly similar results as the combined morphological clustering approach by Fleischmann et al. (2022), with the distinct advantages that the classification approach includes descriptive names and can single out individual complex buildings. This also highlights the value of analyzing the road layer separately from the buildings layer to capture more detailed urban tissue information, since combined clustering mainly seems to capture the urban tissue of buildings.

How can the captured urban form be encoded in a way that allows for the comparison of different cities and generation of new cities with a similar character?

The city stack combined with the typology grid proves to be a comprehensible and effective way of connecting the analysis phase with the generation phase. The encoding allows for easy comparison across cities, and new typology grids

can statistically be compared with typology grids from real cities to make them as similar as possible.

Typology grid simplification proves to be comprehensible and comparable for analysis but might be too simplified for generation, needing further detail in the form of patch or cell-level parameters like density information.

The open source `citypy` command line tool simplifies the process of analyzing and encoding any city across the world.

How can this encoded data be utilized to procedurally generate a digital city model that resembles the form of the encoded real-life city?

Results show simulated annealing as a promising way of generating typology grids based on an analyzed city. Results show plausible patterns being generated with only a small set of shape metrics, suggesting potential for highly plausible patterns with the inclusion of more metrics, including metrics based on both topology and city stack constraints. However, it can be concluded that simulated annealing is not the appropriate methodology in time-critical applications, as simulation times range from minutes to hours.

The road generation methodology followed in this thesis shows the potential of generating road patterns based on the typology template concept. It also highlights the advantages of using global typologies combined with local typology template statistics.

The building generation methodology lists proof of concept algorithms for generating buildings based on building typologies.

All in all, the city stack framework appears to be a viable approach for city analysis and generation.

References

Aarts, E., & Laarhoven, P. van. (1989). Simulated Annealing: An Introduction. *Statistica Neerlandica*, *43*(1), 31–52. https://doi.org/10.1111/j.1467-9574.1989.tb01245.x

Airbus. (2024,). *Imagery*.

Aized Amin Soofi, & Arshad Awan. (2017). Classification Techniques in Machine Learning: Applications and Issues. *Journal of Basic & Applied Sciences*, *13*, 459–465. https://doi.org/10.6000/1927-5129.2017.13.76

Aliaga, D. G., & Vanegas, C. A. (2008). *Interactive Example-Based Urban Layout Synthesis*.

Ankerst, M., Breunig, M. M., Kriegel, H.-P., & Sander, J. (1999). OPTICS: Ordering Points to Identify the Clustering Structure. *ACM SIGMOD Record*, *28*(2), 49–60. https://doi.org/10.1145/304181.304187

Araldi, A., & Fusco, G. (2019). From the Street to the Metropolitan Region: Pedestrian Perspective in Urban Fabric Analysis. *Environment and Planning B: Urban Analytics and City Science*, *46*(7), 1243–1263. https://doi.org/10.1177/2399808319832612

Araújo De Oliveira, V. M. (2022). *Urban Morphology: An Introduction to the Study of the Physical Form of Cities*. Springer International Publishing. https://doi.org/10.1007/978-3-030-92454-6

Aschwanden, G., Haegler, S., Bosché, F., Gool, L., & Schmitt, G. (2011). Empiric Design Evaluation in Urban Planning. *Automation in Construction*, *20*, 299–310. https://doi.org/10.1016/J.AUTCON.2010.10.007

Badhrudeen, M., Derrible, S., Verma, T., Kermanshah, A., & Furno, A. (2022). A Geometric Classification of World Urban Road Networks. *Urban Science*, *6*(1), 11. https://doi.org/10.3390/urbansci6010011

Badwi, I. M., Ellaithy, H. M., & Youssef, H. E. (2022). 3D-GIS Parametric Modelling for Virtual Urban Simulation Using CityEngine. *Annals of GIS*, *28*, 325–341. https://doi.org/10.1080/19475683.2022.2037019

Barett, S. (2009,). *L-Systems Considered Harmful*. http://nothings.org/gamedev/l_systems.html

Basaraner, M., & Cetinkaya, S. (2017). Performance of Shape Indices and Classification Schemes for Characterising Perceptual Shape Complexity of Building Footprints in GIS. *International Journal of Geographical Information Science*, *31*(10), 1952–1977. https://doi.org/10.1080/13658816.2017.1346257

Bidarra, R., Kraker, K. J. de, Smelik, R. M., & Tutenel, T. (2010,). *Integrating Semantics and Procedural Generation: Key Enabling Factors for Declarative Modeling of Virtual Worlds*.

Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., & Çöltekin, A. (2015). Applications of 3D City Models: State of the Art Review. *ISPRS International Journal of Geo-Information*, *4*(4), 2842–2889. https://doi.org/10.3390/ijgi4042842

Boeing, G. (2019, August 17). *Urban Spatial Order: Street Network Orientation, Configuration, and Entropy*. https://doi.org/10.2139/ssrn.3224723

Boeing, G. (2023, August 2). *OSMnx*. https://github.com/gboeing/osmnx

Bourdic, L., Salat, S., & Nowacki, C. (2012). Assessing Cities: A New System of Cross-Scale Spatial Indicators. *Building Research & Information*, *40*(5), 592–605. https://doi.org/10.1080/09613218.2012.703488

Breiman, L. (2001). Statistical Modeling: The Two Cultures. *Statistical Science*, *16*(3), 199–215. https://www.jstor.org/stable/2676681

Chen, G., Esch, G., Wonka, P., Muller, P., & Zhang, E. (2008). *Interactive Procedural Street Modeling*.

Cokelaer, T. (2024, October 18). *Cokelaer/Fitter*. https://github.com/cokelaer/fitter

Coulston, J. W., Reams, G. A., Wear, D. N., & Brewer, C. K. (2014). An Analysis of Forest Land Use, Forest Land Cover and Change at Policy-Relevant Scales. *Forestry*, *87*(2), 267–276. https://doi.org/10.1093/forestry/cpt056

Dibble, J., Prelorendjos, A., Romice, O., Zanella, M., Strano, E., Pagel, M., & Porta, S. (2019). On the Origin of Spaces: Morphometric Foundations of Urban Form Evolution. *Environment and Planning B: Urban Analytics and City Science*, *46*(4), 707–730. https://doi.org/10.1177/2399808317725075

Dorogush, A. V., Ershov, V., & Gulin, A. (2018). *CatBoost: Gradient Boosting with Categorical Features Support*.

Douglas, D. H., & Peucker, T. K. (1973). ALGORITHMS FOR THE REDUCTION OF THE NUMBER OF POINTS REQUIRED TO REPRESENT A DIGITIZED LINE OR ITS CARICATURE. *Cartographica: The International Journal for Geographic Information and Geovisualization*, *10*(2), 112–122. https://doi.org/10.3138/FM57-6770-U75U-7727

Eaves, B., Jr, & Schmidt, M. (2024, October 17). *RV*. https://github.com/promised-ai/rv

Elinder, T. (2017,). *General Methods for the Generation of Seamless Procedural Cities*. https://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=8929185&fileOId=8929189

Emilien, A., Bernhardt, A., Peytavie, A., Cani, M.-P., & Galin, E. (2012). Procedural Generation of Villages on Arbitrary Terrains. *The Visual Computer*, *28*. https://doi.org/10.1007/s00371-012-0699-7

Episcope. (2014,). *IEE Project TABULA*. https://episcope.eu/iee-project/tabula/

Esri. *Community Maps*. Retrieved September 10, 2024, from https://communitymaps.arcgis.com/home/

Esri R&D Center Zurich. *CityEngine*. https://www.esri.com/en-us/arcgis/products/arcgis-cityengine/overview

European Space Agency, & Airbus. (2022). *Copernicus DEM* [Computer software]. European Space Agency. https://doi.org/10.5270/ESA-c5d3d65

Fleischmann, M., Feliciotti, A., Romice, O., & Porta, S. (2020). Morphological Tessellation as a Way of Partitioning Space: Improving Consistency in Urban Morphology at the Plot Scale. *Computers, Environment and Urban Systems*, *80*, 101441. https://doi.org/10.1016/j.compenvurbsys.2019.101441

Fleischmann, M., Feliciotti, A., Romice, O., & Porta, S. (2022). Methodological Foundation of a Numerical Taxonomy of Urban Form. *Environment and Planning B: Urban Analytics and City Science*, *49*(4), 1283–1299. https://doi.org/10.1177/23998083211059835

Fleischmann, t., & PySAL Developers. (2018,). *Momepy*. https://github.com/martinfleis/momepy

Gebauer, M., & Samuels, I. (1981). Urban Morphology: An Introduction. *Joint Centre for Urban Design, Research Note 8. Oxford Polytechnic, Oxford*.

GeoRust. (2023, November 24). *Geo*. https://github.com/georust/geo

Gil, J., Beirão, J. N., Montenegro, N., & Duarte, J. P. (2011). On the Discovery of Urban Typologies: Data Mining the Many Dimensions of Urban Form. *Urban Morphology*, *16*(1), 27–40. https://doi.org/10.51347/jum.v16i1.3966

Girindran, R., Boyd, D., Rosser, J., Vijayan, D., Long, G., & Robinson, D. (2020). On the Reliable Generation of 3D City Models from Open Data. *Urban Science*. https://doi.org/10.3390/URBANSCI4040047

Google. (2024,). *Map Data*.

Google Earth. (2024,). *Sattelite Imagery*.

Greuter, S., Parker, J., Stewart, N., & Leach, G. (2003). *Real-Time Procedural Generation of `pseudo Infinite' Cities*. *3*, 87–94. https://doi.org/10.1145/604471.604490

Groenewegen, S. A., & Smelik, R. M. (2009). *Procedural City Layout Generation Based on Urban Land Use Models*.

Hamaina, R., Leduc, T., & Moreau, G. (2012). Towards Urban Fabrics Characterization Based on Buildings Footprints. In J. Gensel, D. Josselin, & D. Vandenbroucke (Eds.), *Bridging the Geographic Information Sciences: Bridging the Geographic Information Sciences* (pp. 327–346). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-29063-3_18

Hartmann, A., Meinel, G., Hecht, R., & Behnisch, M. (2016). A Workflow for Automatic Quantification of Structure and Dynamic of the German Building Stock Using Official Spatial Data. *ISPRS International Journal of Geo-Information*, *5*(8), 142. https://doi.org/10.3390/ijgi5080142

He, H. S., Ventura, S. J., & Mladenoff, D. J. (2002). Effects of Spatial Aggregation Approaches on Classified Satellite Imagery. *International Journal of Geographical Information Science*, *16*(1), 93–109. https://doi.org/10.1080/13658810110075978

Herfort, B., Lautenbach, S., Albuquerque, J. Porto de, Anderson, J., & Zipf, A. (2023). A Spatio-Temporal Analysis Investigating Completeness and Inequalities of Global Urban Building Data in OpenStreetMap. *Nature Communications*, *14*(1), 3985. https://doi.org/10.1038/s41467-023-39698-6

Hoof, J. van. (2022, October 6). *Yatoom/City-Generator*. https://github.com/Yatoom/city-generator

Jordahl, K., Bossche, J. V. den, Fleischmann, M., Wasserman, J., McBride, J., Gerard, J., Tratner, J., Perry, M., Badaracco, A. G., Farmer, C., Hjelle, G. A., Snow, A. D., Cochran, M., Gillies, S., Culbertson, L., Bartos, M., Eubank, N., maxalbert, Bilogur, A., … Leblanc, F. (2020, July). *Geopandas/Geopandas: V0.8.1*. https://doi.org/10.5281/zenodo.3946761

Kelly, T. (2014). *Unwritten Procedural Modeling with the Straight Skeleton*.

Kim, J.-S., Kavak, H., & Crooks, A. (2018). Procedural City Generation beyond Game Development. *SIGSPATIAL Special*, *10*(2), 34–41. https://doi.org/10.1145/3292390.3292397

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, *220*(4598), 671–680. https://doi.org/10.1126/science.220.4598.671

Kropf, s. (1996). Urban Tissue and the Character of Towns. *Urban Design International*, *1*, 247–263. https://doi.org/10.1057/udi.1996.32

Larkham, P. (2005). Understanding Urban Form?. *Urban Design*, *93*, 22–24.

Lechner, T., Ren, P., Watson, B., Brozefski, C., & Wilenski, U. (2006). Procedural Modeling of Urban Land Use. *ACM SIGGRAPH 2006 Research Posters on - SIGGRAPH '06*, 135. https://doi.org/10.1145/1179622.1179778

Ledoux, H. (2018). Val3dity: Validation of 3D GIS Primitives According to the International Standards. *Open Geospatial Data, Software and Standards*, *3*(1), 1. https://doi.org/10.1186/s40965-018-0043-x

Lind, P. G., González, M. C., & Herrmann, H. J. (2005). Cycles and Clustering in Bipartite Networks. *Physical Review E*, *72*(5), 56127. https://doi.org/10.1103/PhysRevE.72.056127

Louf, R., & Barthelemy, M. (2014, October 8). *A Typology of Street Patterns*. https://doi.org/10.1098/?rsif.2014.0924

Lynch, K. (1981). *A Theory of Good City Form*. MIT Press (MA).

Madhulatha, T. S. (2012). *An Overview on Clustering Methods*. https://doi.org/10.48550/ARXIV.1205.1117

Maps, O. (2023, December 14). *Overture Buildings Theme Hits 2.3B Buildings With Addition of Google Open Buildings Data*. https://overturemaps.org/overture-buildings-theme-hits-2-3b-buildings-with-addition-of-google-open-buildings-data/

Marshall, S. (2005). *Streets & Patterns* (1st ed). Spon.

Maxar Technologies. (2024,). *Imagery*.

McGarigal, K., & Marks, B. J. (1995). *FRAGSTATS: Spatial Pattern Analysis Program for Quantifying Landscape Structure*. (Issue PNW-GTR-351, p. PNW–GTR–351). https://doi.org/10.2737/PNW-GTR-351

McInnes, L., Healy, J., & Astels, S. (2017). Hdbscan: Hierarchical Density Based Clustering. *Journal of Open Source Software*, *2*(11), 205. https://doi.org/10.21105/joss.00205

Microsoft. (2024, September 9). *GlobalMLBuildingFootprints*. https://github.com/microsoft/GlobalMLBuildingFootprints

Müller, P., Wonka, P., Haegler, S., Ulmer, A., & Van Gool, L. (2006). Procedural Modeling of Buildings. *ACM Transactions on Graphics*, *25*(3), 614–623. https://doi.org/10.1145/1141911.1141931

Nishida, G., Garcia-Dorado, I., & Aliaga, D. G. (2016). Example-Driven Procedural Urban Roads. *Computer Graphics Forum*, *35*(6), 5–17. https://doi.org/10.1111/cgf.12728

OpenStreetMap contributors. (2017,). *Planet Dump Retrieved from https://planet.osm.org*. https://www.openstreetmap.org/

OpenStreetMap Wiki. *Key:Building*. Retrieved September 10, 2024, from https://wiki.openstreetmap.org/wiki/Key:building

OpenTopography. (2021,). *Copernicus GLO-90 Digital Surface Model*. https://doi.org/10.5069/G9028PQB

OSM Contributors. *Nominatim*. Retrieved September 10, 2024, from https://nominatim.org/

Overture Maps. (2023,). *Overture Maps Foundation Releases Its First World-Wide Open Map Dataset – Overture Maps Foundation*. https://overturemaps.org/overture-maps-foundation-releases-first-world-wide-open-map-dataset/

Parish, Y. I. H., & Müller, P. (2001). *Procedural Modeling of Cities*. https://doi.org/10.1145/1185657.1185716

Pesaresi, M. (2023). *GHS-BUILT-H R2023A - GHS Building Height, Derived from AW3D30, SRTM30, and Sentinel2 Composite (2018)* [Computer software]. European Commission, Joint Research Centre (JRC). https://doi.org/10.2905/85005901-3A49-48DD-9D19-6261354F56FE

Peters, R., Dukai, B., Vitalis, S., Liempt, J. van, & Stoter, J. (2022,). *Automated 3D Reconstruction of LoD2 and LoD1 Models for All 10 Million Buildings of the Netherlands* (Vol. 88, Issue 3, pp. 165–170). https://doi.org/10.14358/PERS.21-00032R2

Poon, C. (2024). *Inferring the Residential Building Type from 3DBAG*. https://resolver.tudelft.nl/uuid:3ef77acb-b38b-4fa5-9b1e-31813b00b739

Prusinkiewicz, P., & Lindenmayer, A. (1996). *The Algorithmic Beauty of Plants* (First soft cover printing). Springer.

Rerun-Io/Rerun. (2024, October 25). https://github.com/rerun-io/rerun

Reynolds, D. (2009). Gaussian Mixture Models. In S. Z. Li & A. Jain (Eds.), *Encyclopedia of Biometrics: Encyclopedia of Biometrics* (pp. 659–663). Springer US. https://doi.org/10.1007/978-0-387-73003-5_196

Saxena, A., Prasad, M., Gupta, A., Bharill, N., Patel, O. P., Tiwari, A., Er, M. J., Ding, W., & Lin, C.-T. (2017). A Review of Clustering Techniques and Developments. *Neurocomputing*, *267*, 664–681. https://doi.org/10.1016/j.neucom.2017.06.053

Schirmer, P. M., & Axhausen, K. W. (2015a). A Multiscale Classification of Urban Morphology. *Journal of Transport and Land Use*. https://doi.org/10.5198/jtlu.2015.667

Schirmer, P. M., & Axhausen, K. W. (2015b). A Multiscale Classification of Urban Morphology. *Journal of Transport and Land Use*. https://doi.org/10.5198/jtlu.2015.667

Schwarz, G. (1978). Estimating the Dimension of a Model. *The Annals of Statistics*, *6*(2), 461–464. https://www.jstor.org/stable/2958889

Shannon, C. E. (1948). A Mathematical Theory of Communication. *Bell System Technical Journal*, *27*(3), 379–423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

Sirko, W., Kashubin, S., Ritter, M., Annkah, A., Bouchareb, Y. S. E., Dauphin, Y., Keysers, D., Neumann, M., Cisse, M., & Quinn, J. (2021, July 29). *Continental-Scale Building Detection from High Resolution Satellite Imagery*. http://arxiv.org/abs/2107.12283

Smelik, R. M. (2011). *A Declarative Approach to Procedural Generation of Virtual Worlds*.

Smelik, R., Tutenel, T., Bidarra, R., & Benes, B. (2014,). *A Survey on Procedural Modeling for Virtual Worlds*. https://www.semanticscholar.org/paper/A-Survey-on-Procedural-Modeling-for-Virtual-Worlds-Smelik-Tutenel/05311671fe92fc09fdc153c14654a16fb63b7348

Southworth, M., & Ben-Joseph, E. (2003). *Streets and the Shaping of Towns and Cities*. Island Press.

Steadman, P., Bruhns, H. R., Holtier, S., Gakovic, B., Rickaby, P. A., & Brown, F. E. (2000). A Classification of Built Forms. *Environment and Planning B: Planning and Design*, *27*(1), 73–91. https://doi.org/10.1068/bst7

Steiniger, S., Lange, T., Burghardt, D., & Weibel, R. (2008). An Approach for the Classification of Urban Building Structures Based on Discriminant Analysis Techniques. *Transactions in GIS*, *12*(1), 31–59. https://doi.org/10.1111/j.1467-9671.2008.01085.x

Trosset, M. W. (2001). What Is Simulated Annealing?. *Optimization and Engineering*, *2*(2), 201–213. https://doi.org/10.1023/A:1013193211174

Vanegas, C. A., Garcia-Dorado, I., Aliaga, D. G., Benes, B., & Waddell, P. (2012a). Inverse Design of Urban Procedural Models. *ACM Transactions on Graphics*, *31*(6), 1–11. https://doi.org/10.1145/2366145.2366187

Vanegas, C. A., Kelly, T., Weber, B., Halatsch, J., Aliaga, D. G., & Müller, P. (2012b). Procedural Generation of Parcels in Urban Modeling. *Computer Graphics Forum*, *31*(2pt3), 681–690. https://doi.org/10.1111/j.1467-8659.2012.03047.x

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., Walt, S. J. van der, Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2

Wang, J., Huang, W., & Biljecki, F. (2024). Learning Visual Features from Figure-Ground Maps for Urban Morphology Discovery. *Computers, Environment and Urban Systems*, *109*, 102076. https://doi.org/10.1016/j.compenvurbsys.2024.102076

Weber, B., Müller, P., Wonka, P., & Gross, M. (2009). Interactive Geometric Simulation of 4D Cities. *Computer Graphics Forum*, *28*(2), 481–492. https://doi.org/10.1111/j.1467-8659.2009.01387.x

Wurm, M., Schmitt, A., & Taubenbock, H. (2016). Building Types' Classification Using Shape-Based Features and Linear Discriminant Functions. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *9*(5), 1901–1912. https://doi.org/10.1109/JSTARS.2015.2465131

Zanaga, D., Van De Kerchove, R., De Keersmaecker, W., Souverijns, N., Brockmann, C., Quast, R., Wevers, J., Grosu, A., Paccini, A., Vergnaud, S., Cartus, O., Santoro, M., Fritz, S., Georgieva, I., Lesiv, M., Carter, S., Herold, M., Li, L., Tsendbazar, N.-E., … Arino, O. (2021). *ESA WorldCover 10 m 2020 V100* (Version v100) [Computer software]. Zenodo. https://doi.org/10.5281/ZENODO.5571936

Zhou, Q., Zhang, Y., Chang, K., & Brovelli, M. A. (2022). Assessing OSM Building Completeness for Almost 13,000 Cities Globally. *International Journal of Digital Earth*, *15*(1), 2400–2421. https://doi.org/10.1080/17538947.2022.2159550

9 Appendices

Appendix A Road Node Metrics Overview

Node Metric	Description	Reason(s)	Unit	Source(s)
`is intersection`	See Section A	• Prerequisite • Generation statistics	-	New
`is major intersection`	See Section A	• Prerequisite	-	New
`node degree`	See Section A	• Prerequisite	-	New
`major node degree`	See Section A	• Prerequisite	-	New
`clustering`	The fraction of possible squares that exist at each node	• Clustering metric	-	(Lind et al., 2005)

Table 6 - Road node metrics overview

Appendix B Road Segment Metrics Overview

Segment Metric	Description	Reason(s)	Unit	Source(s)
`next segment`	Next segment in the forward direction, if any. See Section A.	• Prerequisite	-	New
`previous segment`	Segment that has the current segment as `next segment`, if any.	• Prerequisite	-	New
`right neighbour`	See Section F	• Prerequisite	-	New
`major right neighbour`	See Section F	• Prerequisite	-	New
`road section id`	See Section B	• Prerequisite	-	New
`road major section id`	See Section B	• Prerequisite	-	New
`distance to last intersection`	Distance from the end point of this segment to the last intersection.	• Generation influence	m	New

Next Segment

Right Neighbour

Sections

Distance

Segment Metric	Description	Reason(s)	Unit	Source(s)
`distance to last major intersection`	Distance from the end point of this segment to the last major intersection.	• Generation influence	m	New
`previous segment length`	Length of the `previous segment`	• Generation influence	-	New
`section length`	Total length of the road section, see `road section id`	• Clustering Metric • Generation statistic	m	New
`major section length`	Total length of the major road section, see `major section id`.	• Clustering metric • Generation statistic	m	New
`right neighbour distance`	Distance to `right neighbour`.	• Cluster metric	-	New
`right major neighbour distance`	Distance to `major right neighbour`.	• Cluster metric	-	New
Angles				
`forward angle`	See Section A	• Generation statistic • Prerequisite • Clustering metric	°	New
`previous segment forward angle`	Forward angle of `previous_segment`, if any.	• Generation influence	-	New
`bearing`	Compass bearing of this segment	• Generation statistic	°	-
`intersection left angle`	See Section E	• Clustering metric • Generation statistic	°	New
`major intersection left angle`	See Section E	• Clusterin metric • Generation statistic	°	New

Segment Metric	Description	Reason(s)	Unit	Source(s)	
`right neighbour angle deviation`	See Section F	• Cluster metric	-	New	
`right major neighbour angle deviation`	See Section F	• Cluster metric	-	New	
`next node degree`	Degree of the end node of this segment.	• Generation statistic	-	New	Intersections
`next node major degree`	Major degree of the end node of this segment.	• Generation statistic	-	New	
`dead end section`	True if this road section has a dead end on either side	• Clustering metric • Generation statistic	-	New	
`continuity`	See Section C	• Clustering metric	-	New	Shape
`stretch linearity`	See Section D	• Prerequisite	-	New	
`stretch curvilinearity`	See Section D	• Clustering metric • Generation statistic	-	New	
`on domain border`	True if within set distance of domain border	• Prerequisite	-	New	Misc.

Table 7 - Road segment metrics overview

Appendix C Building Metrics Overview

Segment Metric	Description	Reason(s)	Unit	Source(s)
`area`	Area of the building footprint.	• Classification metric • Generation statistic	m^2	-
`perimeter`	Perimeter of the building footprint	• Classification metric	m	-
`equivalent rectangular index`	Version of rectangularity that compensates for protrusions.	• Classification metric	-	(Basaraner & Cetinkaya, 2017; Fleischmann & PySAL Developers, 2018)
`elongation`	The elongation of the minimum bounding rectangle	• Classification metric	-	(Fleischmann & PySAL Developers, 2018; Gil et al., 2011)
`shared walls ratio`	Shared walls ratio of adjacent buildings	• Classification metric	-	(Fleischmann & PySAL Developers, 2018; Hamaina et al., 2012)
`neighbour distance`	Mean distance to adjacent buildings	• Classification metric	m	(Fleischmann & PySAL Developers, 2018; Schirmer & Axhausen, 2015a)
`closest road edge id`	Unique ID of closest road edge segment	• Prerequisite	-	(Fleischmann & PySAL Developers, 2018)
`building group id`	See Section A	• Prerequisite	-	New

Segment Metric	Description	Reason(s)	Unit	Source(s)
`building group enclosure perimeter coverage`	See Section B	• Classification metric	-	New
`building group area std`	Standard deviation of the area of buildings within this building group	• Classification metric	-	New
`building group elongation std`	Standard deviation of the elongation of buldings within this building group	• Classification metric	-	New & (Fleischmann & PySAL Developers, 2018; Gil et al., 2011)
`building group courtyard index`	Courtyard Index of the combined shape of the building group	• Classification metric	-	New & (Fleischmann & PySAL Developers, 2018; Schirmer & Axhausen, 2015a)
`building enclosure id`	Unique ID of the enclosure this building is in	• Prerequisite	-	-
`form factor`	Form factor of the building	• Classification metric	-	(Bourdic et al., 2012; Fleischmann & PySAL Developers, 2018)
`land use category`	Category of the land use area this building is in	• Classification metric	-	-
`shape index`	Shape metric related to area	• Classification metric	-	(Fleischmann & PySAL

Segment Metric	Description	Reason(s)	Unit	Source(s)
	and longest axis.			Developers, 2018)
`orientation`	Deviation of orientation of the cardinal axes (0 to 45)	• Generation statistic	°	(Fleischmann & PySAL Developers, 2018; Schirmer & Axhausen, 2015a)
`alignment`	The mean deviation of solar orientation of this building with neighbouring buildings.	• Classification metric	-	(Fleischmann & PySAL Developers, 2018)
`squareness`	Statistical measure of footprint corners deviating from 90 degrees.	• Classification metric	-	(Dibble et al., 2019; Fleischmann & PySAL Developers, 2018)
`approximate height`	See Section C	• Classification metric • Generation statistic	m	New
`covered area`	The area covered by the footprints of the building and its neighbours	• Clustering metric	m^2	(Fleischmann & PySAL Developers, 2018)

Table 8 - Building metrics overview

Appendix D Considered road contextualizing methods

Name	Description	Pros	Cons
Radius	The neighborhood is defined as all segments intersecting with a circle of radius r around the centroid of the current segment.	• Simple	• Unpredictable • Scale sensitive, the amount of selected features is highly dependant on the scale of the segment and the neighbourhood.
IDW	Same as *Radius*, but with inverse distance weighting for neighbourhood features.	• Simple • More importance for closer features, leading to smoother data	• Scale sensitive • Smooth results do not capture borders between typologies
Constrained IDW	Version of *IDW* where the selection circle cannot cross major roads, so features on the other side of the major road are not selected.	• Importance pro from IDW • Due to constraints, more chance of abrupt style change borders	• Scale Sensitive • Complex • Still no clear style borders if two styles within the same major road enclosure
Grid	Select all features contained in the same grid cell as neighbourhood.	• Simple • Fast	• Scale sensitive • Unnatural changes due to grid cell change
N closest features	Select a specific amount of closest segments from	• Guarantee of sufficient sample size • Simple	• Resulting network might be disconnected

Name	Description	Pros	Cons
	the current segment.	• High level of control	• Unrepresentative for sparse network, where there are no segments nearby
Recursive network growing	Starting from the current segment, recursively grow n levels deep in all possible directions in the road graph. Constrained to not grow past major road intersections. This methodology was later found to be similar to the "2 minute driving distance" from Schirmer & Axhausen (2015b)	• Resulting network guaranteed to be connected • Not scale sensitive • High level of control • Captures streets that logically "belong" together	• Streets that are not connected within n levels, but still close, will not be captured. • Size of resulting neighbourhood sensitive to segment length

Table 9 - Potential methods for road neighbourhood selection.

Appendix E Road Clusters Z-Scores

Z-scores calculated for all 13 clusters based on a subset of all data. The subset was created by taking 10 thousand random segments from each city, resulting in a total of 45 thousand segments.

Metric	0	1	2	3	4	5	6	7	8	9	10	11	12
1	0.57	−0.46	−0.82	−0.02	0.18	−0.17	−0.01	−0.49	0.90	1.32	0.73	−0.75	0.41
2	−0.38	0.75	−0.47	1.03	2.21	−0.80	0.04	−0.27	0.10	1.83	1.08	−0.14	−0.14
3	−0.06	0.24	−0.40	1.91	1.91	−0.84	−0.86	0.83	0.17	0.38	0.15	−0.65	−0.53
4	0.43	−0.60	−0.96	−0.35	−0.48	−0.34	0.38	−0.27	0.56	0.65	0.70	0.26	−0.06
5	−0.20	−0.09	−0.34	0.59	6.72	−0.38	−0.25	−0.16	−0.18	7.12	0.32	−0.24	−0.26
6	−0.16	−0.06	−0.31	0.72	9.46	−0.37	−0.31	−0.13	−0.17	8.52	0.31	−0.30	−0.22
7	0.05	−0.50	−0.92	−0.02	0.40	−0.31	0.16	−0.60	0.92	0.58	0.58	−0.46	0.20

1: `angle_entropy::nb`
2: `right_neighbour_distance::nb::weighted_mean`
3: `continuity::nb::weighted_median`
4: `stretch_curvilinearity::nb::weighted_mean`
5: `section_length::nb::mean`
6: `section_length::nb::std`
7: `right_neighbour_angle_deviation::nb::weighted_mean`

Appendix F Building Classification Feature Importance

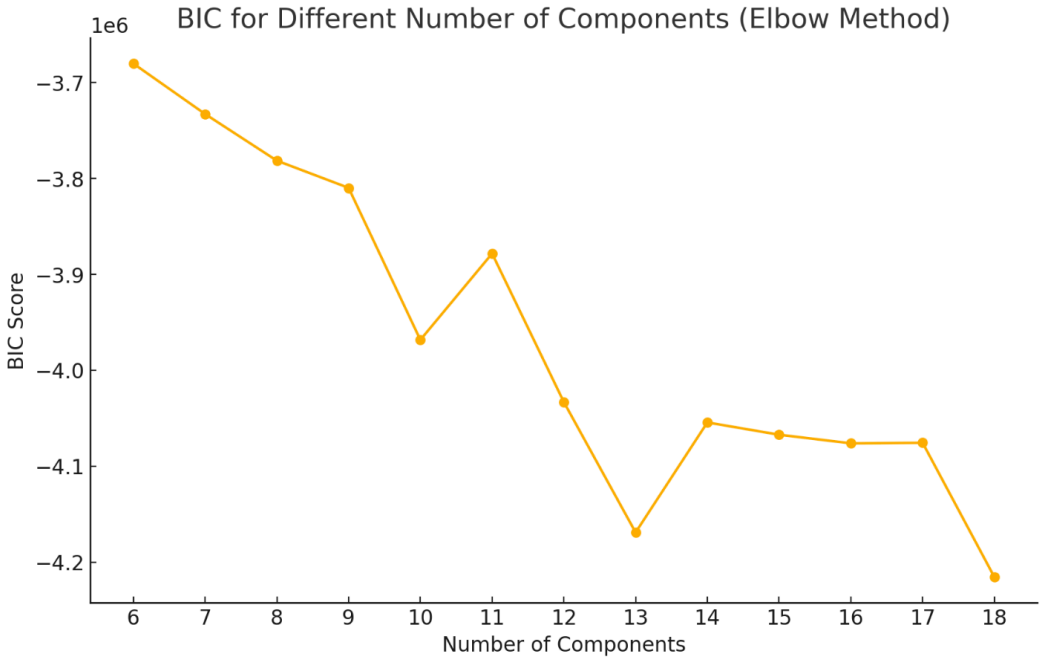
The PredictionValueChange metric from `catboost` shows how much on average the prediction changes if the feature value changes. High values indicate that this metric is very important for distinguishing between the classes, and low values mean it doesn't have much impact.

Feature Id	Importance
`land_use_category`	12.67
`shared_walls_ratio::nb_radius_300::mean`	8.28
`enclosurue_building_area_ratio`	5.62
`approximate_height::nb_radius_300::mean`	4.39
`shape_index::nb_radius_300::std`	4.35
`building_group_enclosure_perimeter_coverage::nb_radius_300::mean`	4.25
`covered_area`	4.05

covered_area::nb_radius_300::mean	3.64
equivalent_rectangular_index::nb_radius_300::std	3.15
elongation::nb_radius_300::mean	2.98
area::nb_radius_300::mean	2.72
land_use_category::nb_radius_300::mode	2.71
equivalent_rectangular_index::nb_radius_300::mean	2.57
building_group_elongation_std::nb_radius_300::mean	2.57
area::nb_radius_300::std	2.50
elongation::nb_radius_300::std	2.38
subtype	2.28
approximate_height::nb_radius_300::std	2.24
shared_walls_ratio::nb_radius_300::std	2.17
building_group_enclosure_perimeter_coverage::nb_radius_300::std	1.94
shape_index::nb_radius_300::mean	1.74
squareness::nb_radius_300::mean	1.72
approximate_height::nb_radius_300::max	1.71
orientation	1.57
area	1.55
covered_area::nb_radius_300::std	1.51
shared_walls_ratio	1.36
building_group_area_std::nb_radius_300::mean	1.22
alignment::nb_radius_300::std	1.10
building_group_enclosure_perimeter_coverage	1.09
alignment::nb_radius_300::mean	1.00
perimeter	0.98
building_group_courtyard_index	0.97
squareness	0.91
squareness::nb_radius_300::std	0.85
alignment	0.65
elongation	0.33
real_height	0.31
form_factor	0.30
shape_index	0.29
building_group_area_std	0.29

`approximate_height`	0.26
`equivalent_rectangular_index`	0.23
`class`	0.20
`building_group_elongation_std`	0.15
`real_levels`	0.14
`roof_shape`	0.08

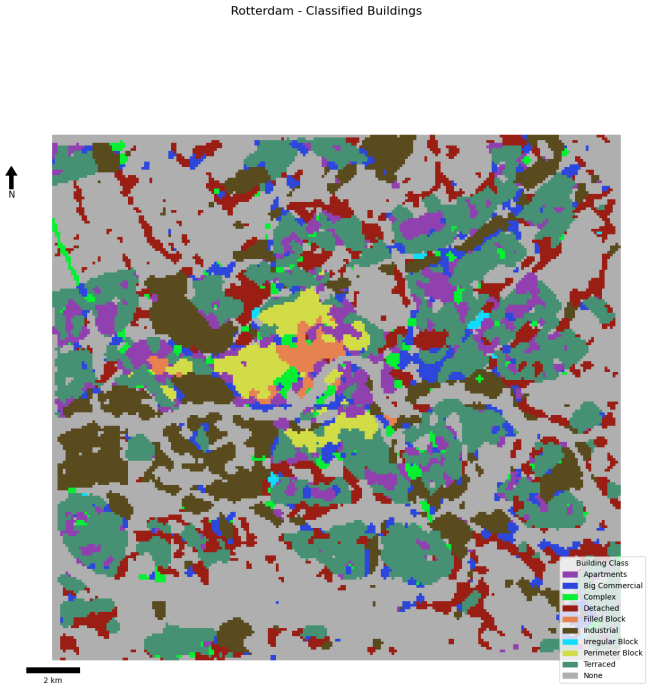
Appendix G Road Clustering Elbow Method



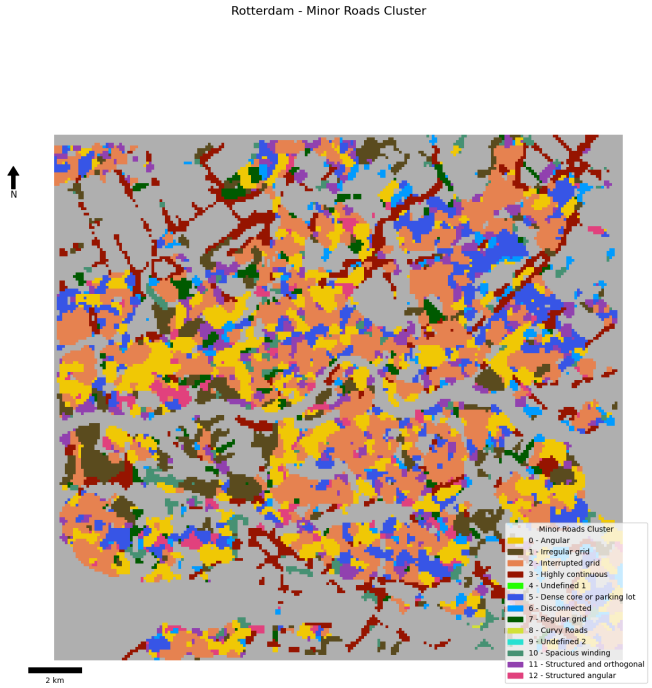
The optimal amount of clusters was determined by detecting highest change in the second-order difference (maximum curvature). Clustering with 13 components was detected as the optimal amount.

Appendix H Final Typology Grids

(Grids start on the next page)



(a) Building Classes

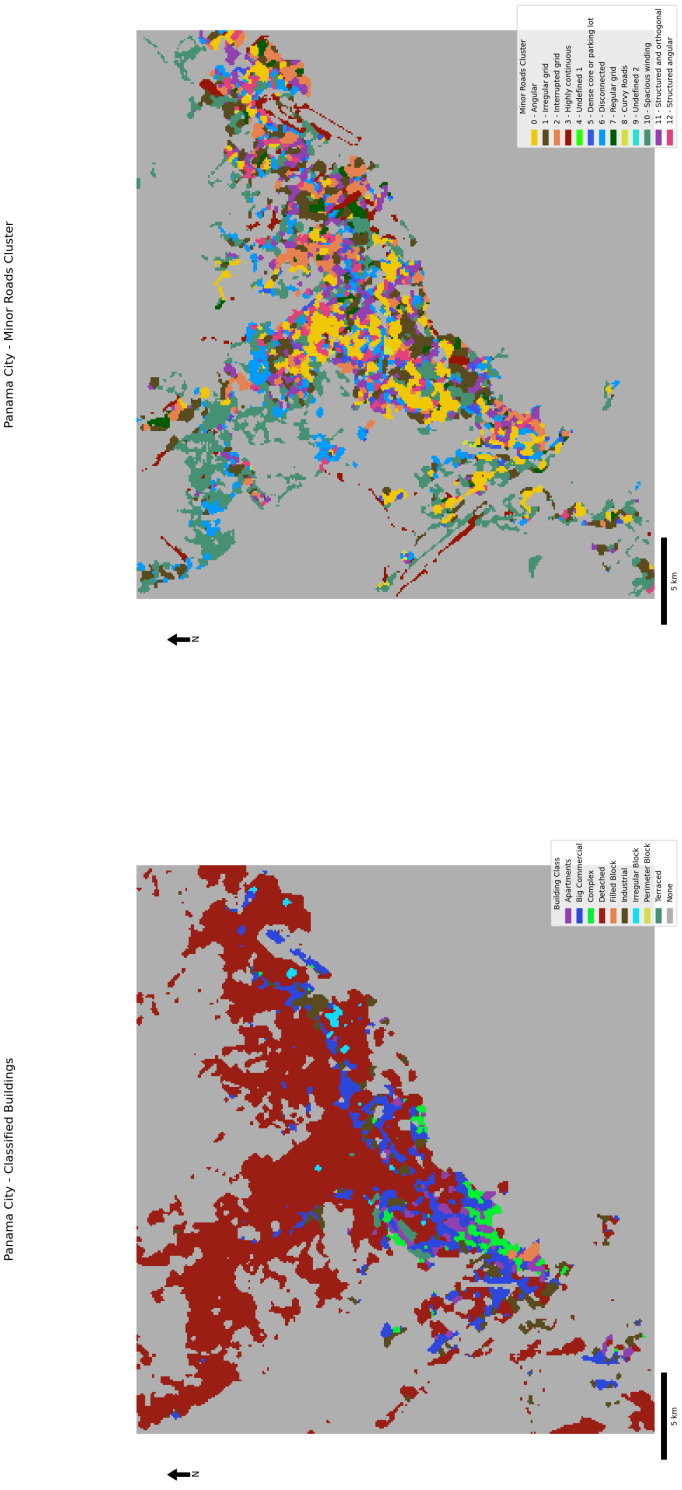


(b) Minor Roads Clusters

Figure 95 - Final typology grids for Rotterdam

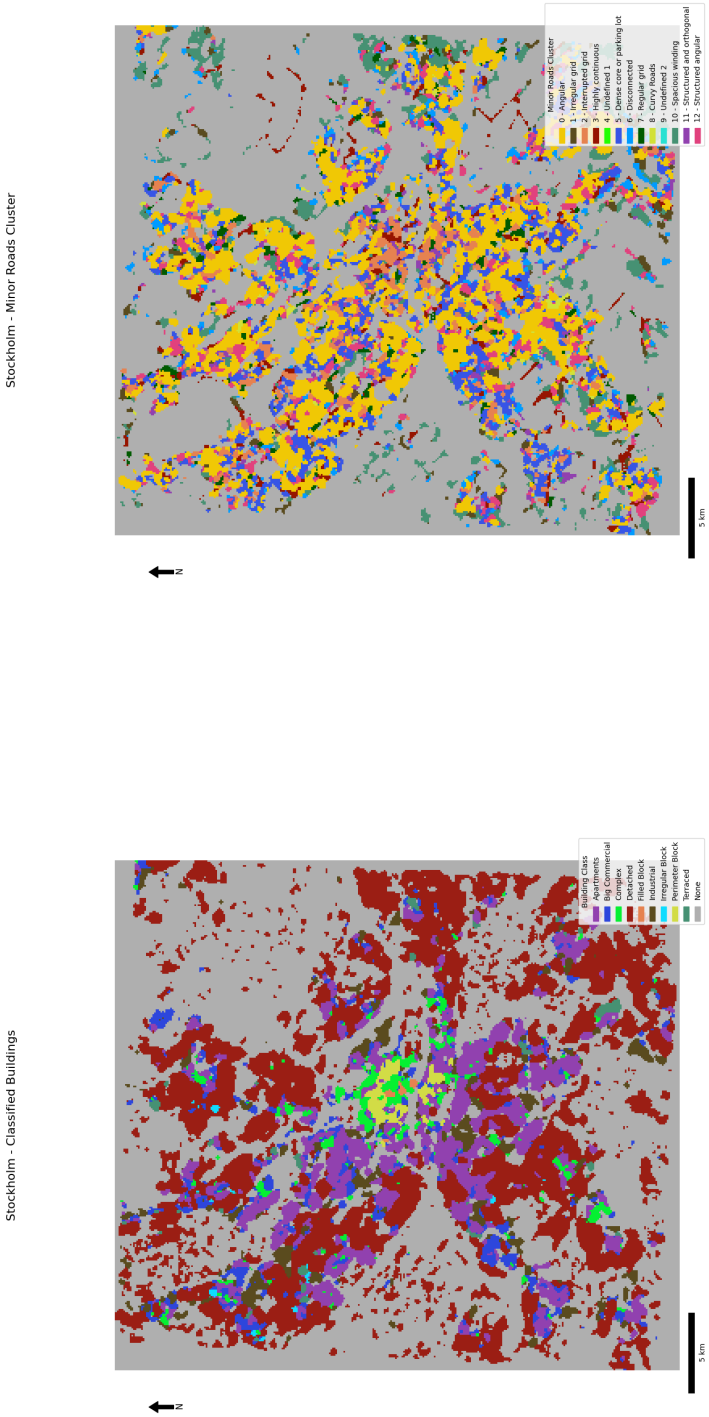
(a) Building Classes (b) Minor Roads Clusters

Figure 96 - Final typology grids for Havana



(a) Building Classes (b) Minor Roads Clusters

Figure 97 - Final typology grids for Panama city

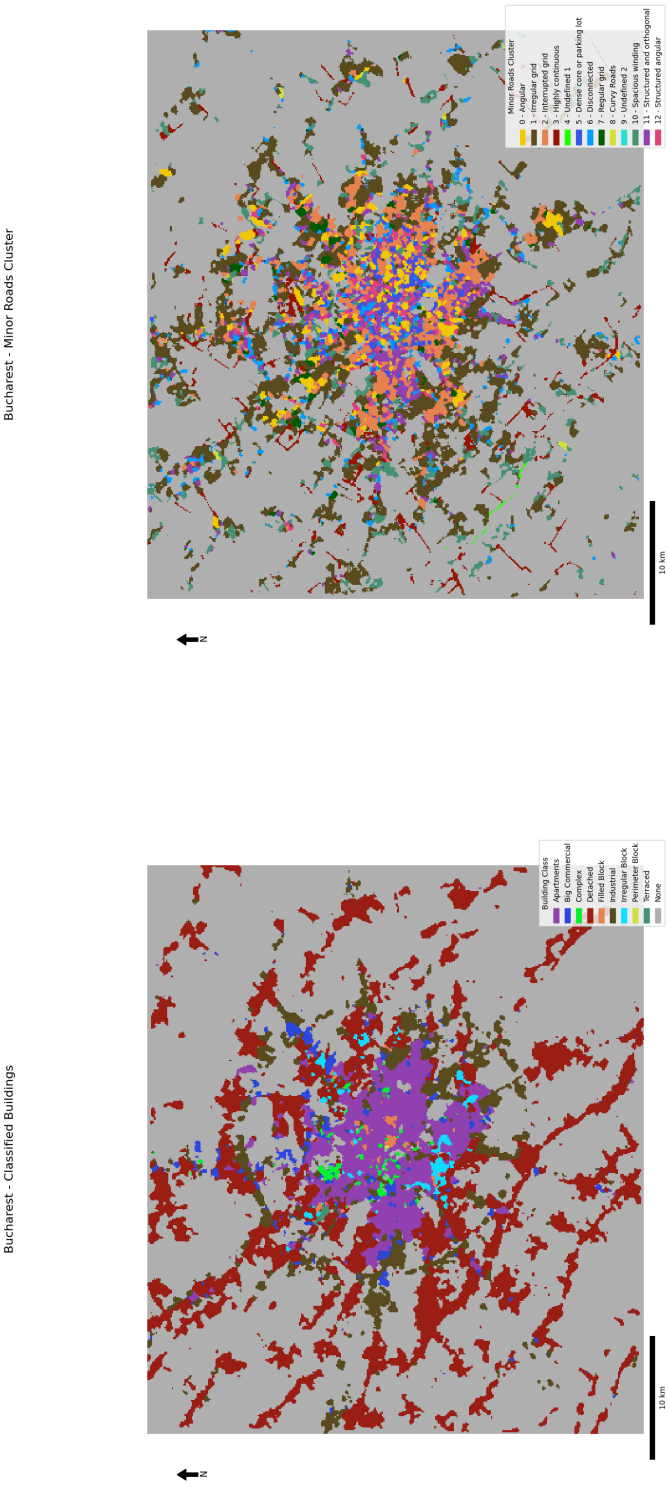


(a) Building Classes (b) Minor Roads Clusters

Figure 98 - Final typology grids for Stockholm

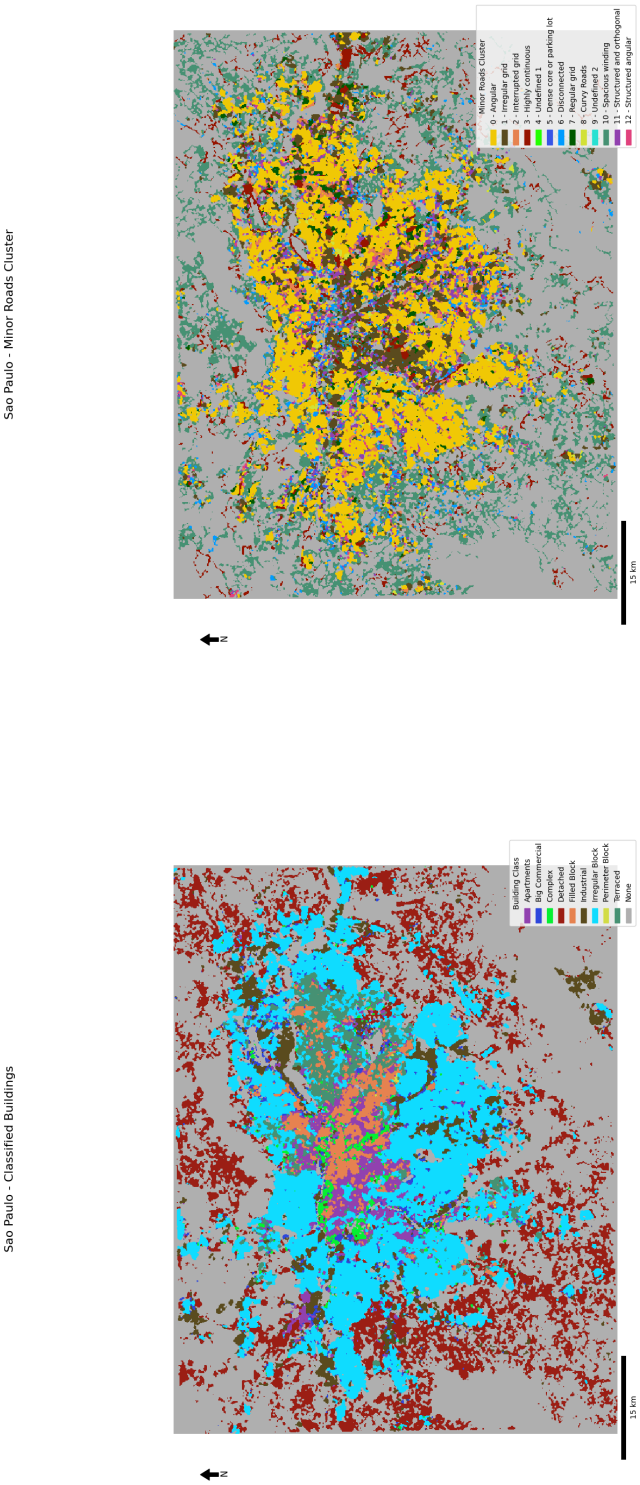
(a) Building Classes (b) Minor Roads Clusters

Figure 99 - Final typology grids for Dakar



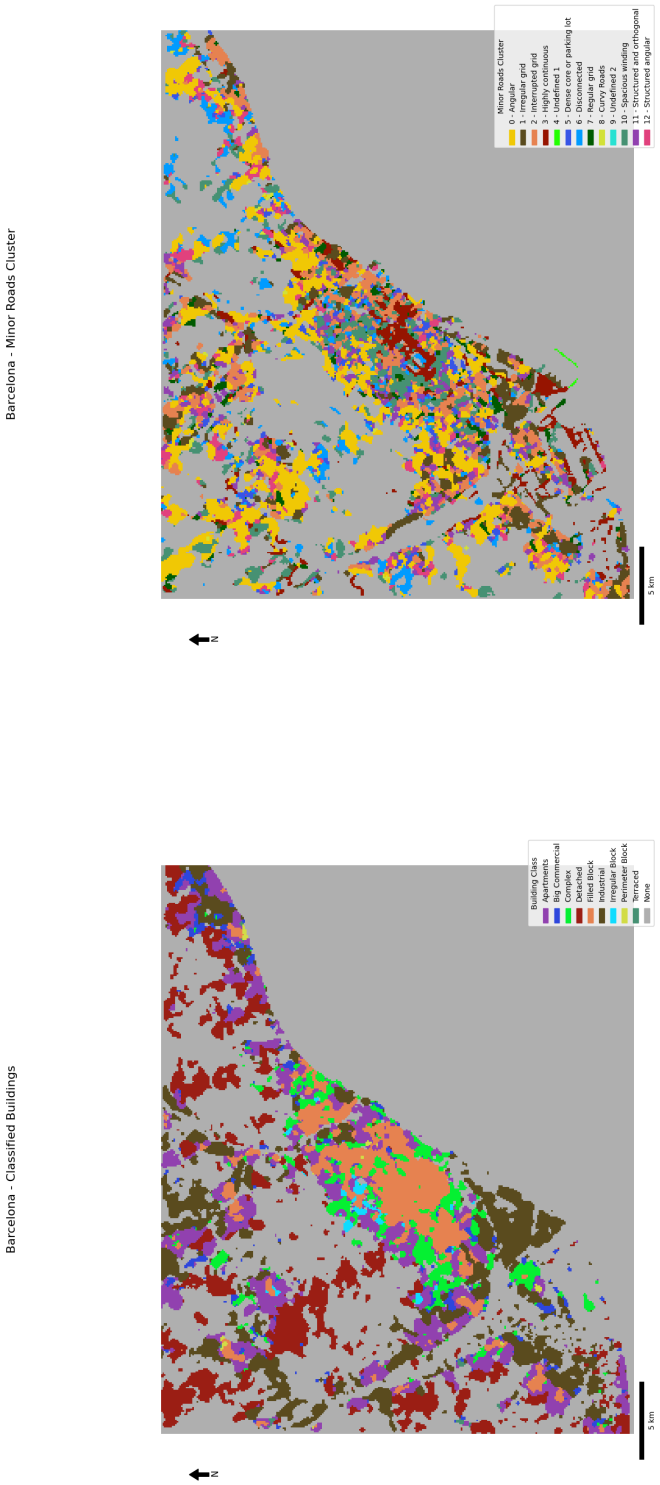
(a) Building Classes (b) Minor Roads Clusters

Figure 100 - Final typology grids for Bucharest



(a) Building Classes (b) Minor Roads Clusters

Figure 101 - Final typology grids for Sao paulo

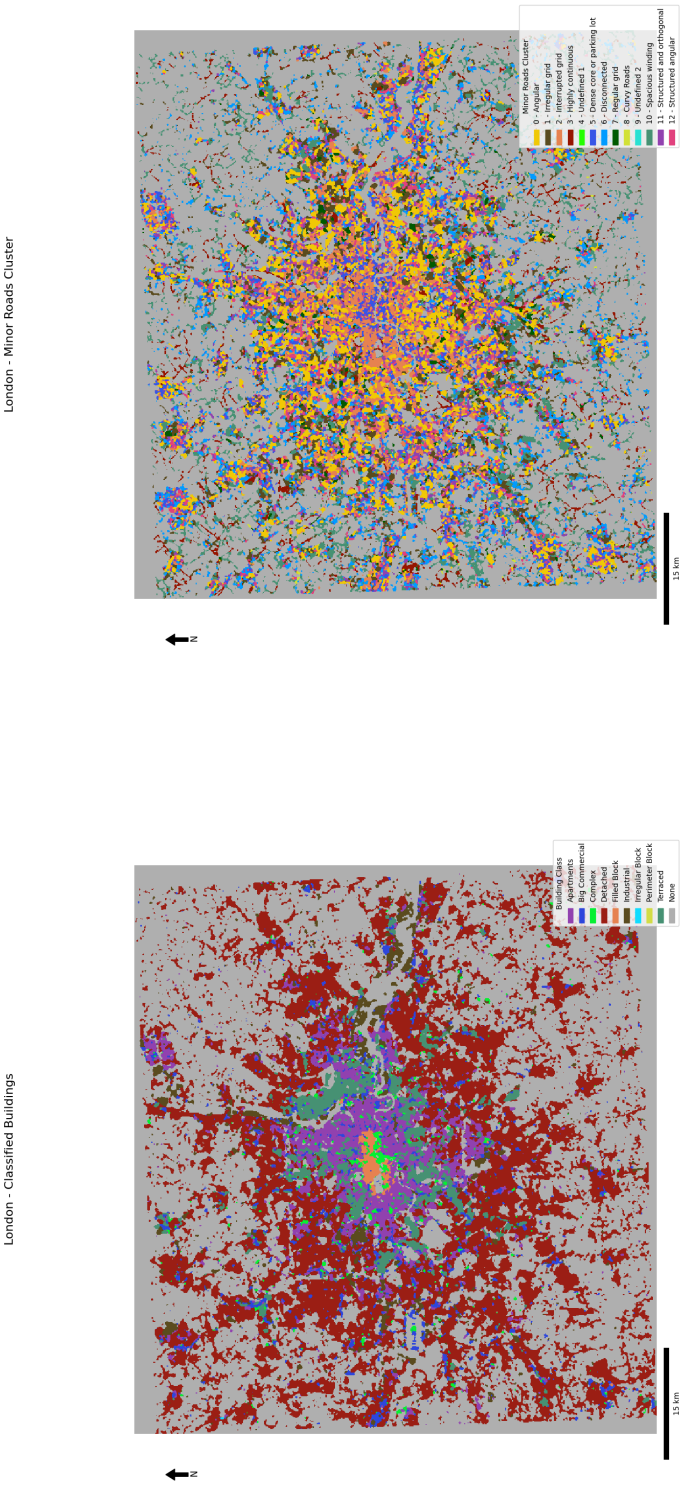


(a) Building Classes (b) Minor Roads Clusters

Figure 102 - Final typology grids for Barcelona

(a) Building Classes (b) Minor Roads Clusters

Figure 103 - Final typology grids for Lisbon

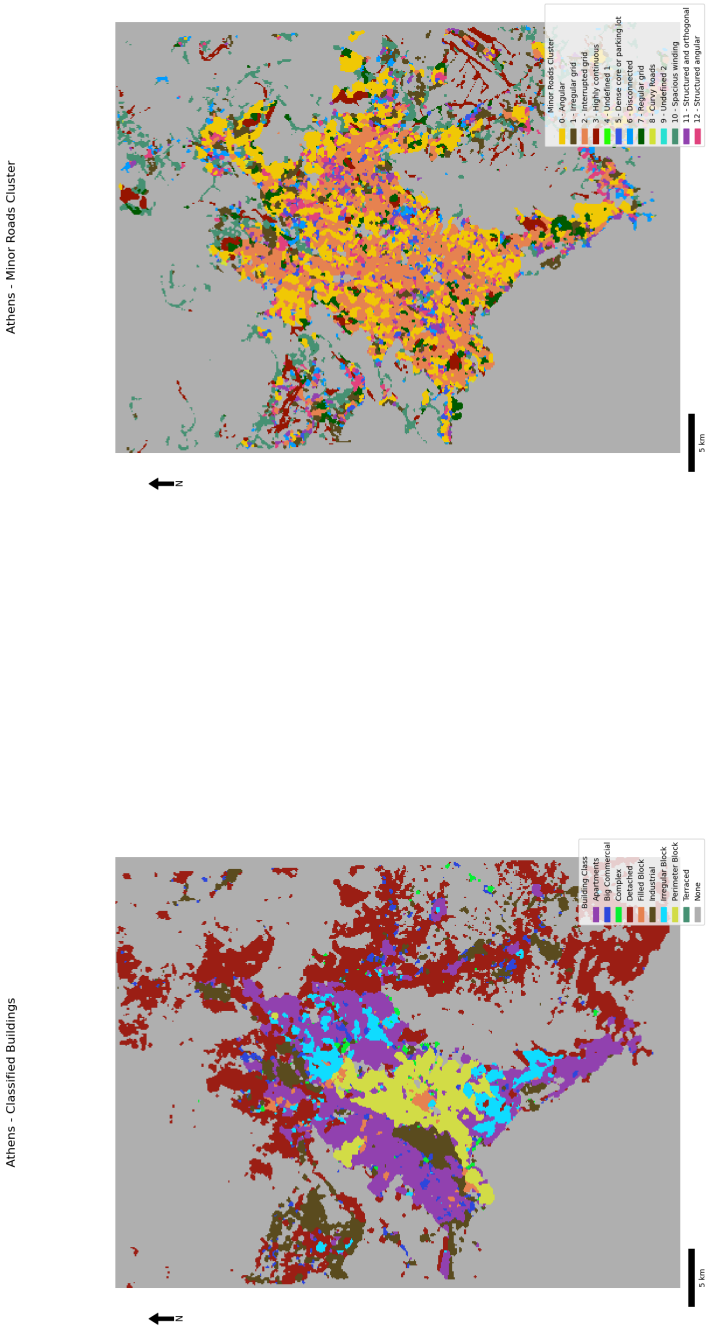


(a) Building Classes (b) Minor Roads Clusters

Figure 104 - Final typology grids for London

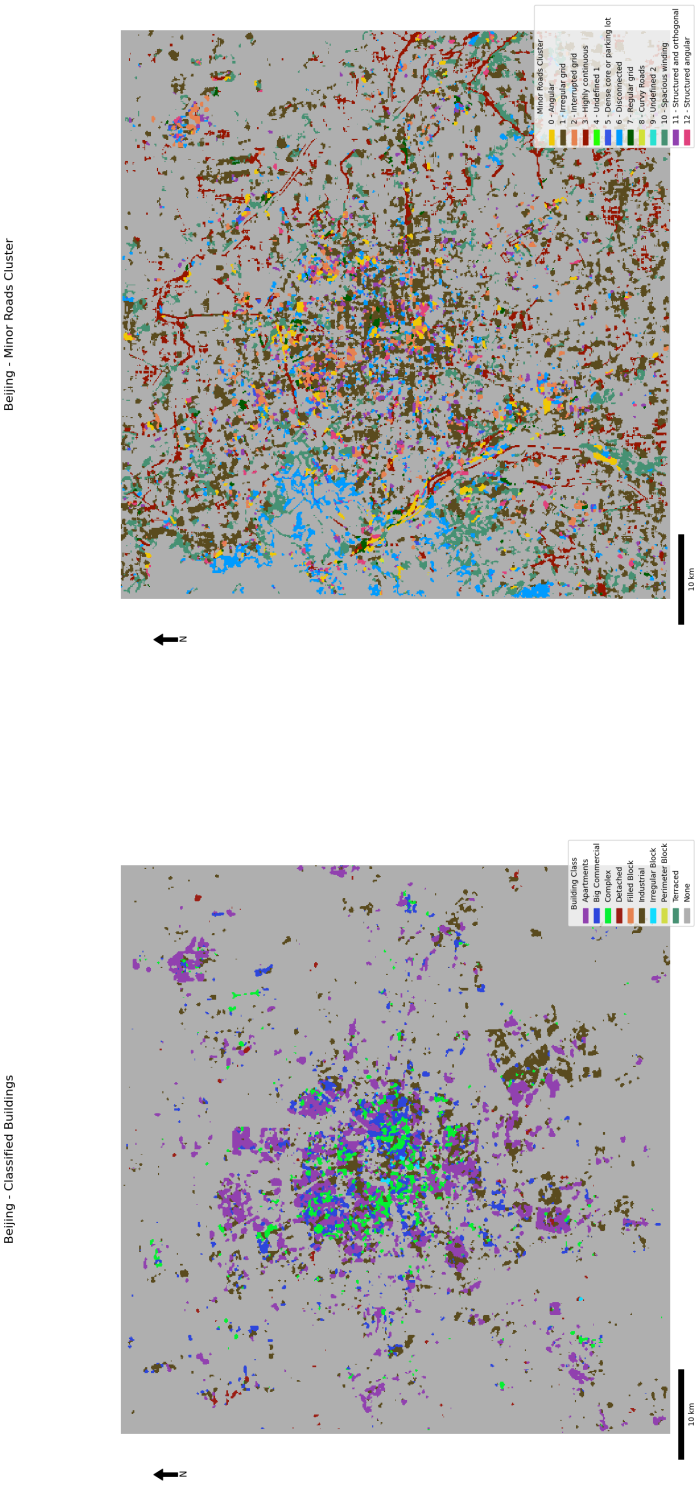
(a) Building Classes (b) Minor Roads Clusters

Figure 105 - Final typology grids for Paramaribo



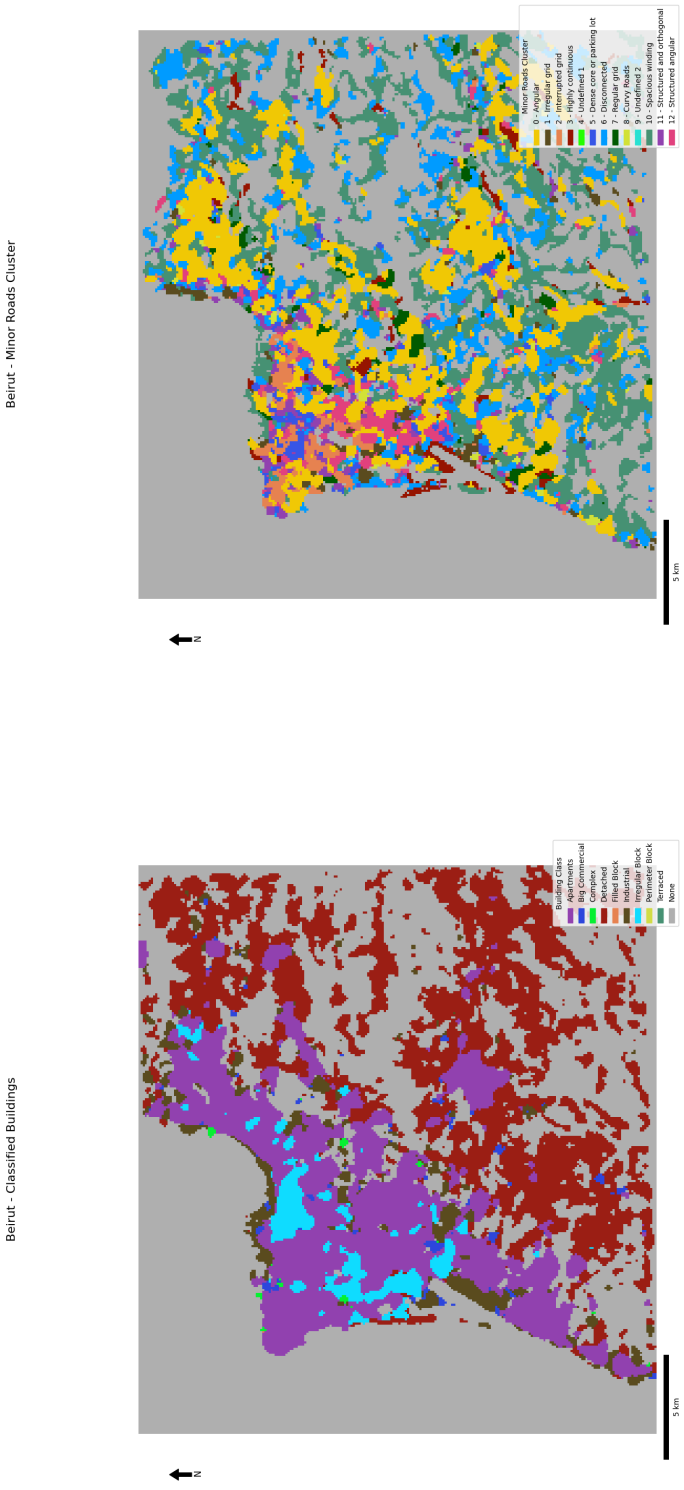
(a) Building Classes (b) Minor Roads Clusters

Figure 106 - Final typology grids for Athens



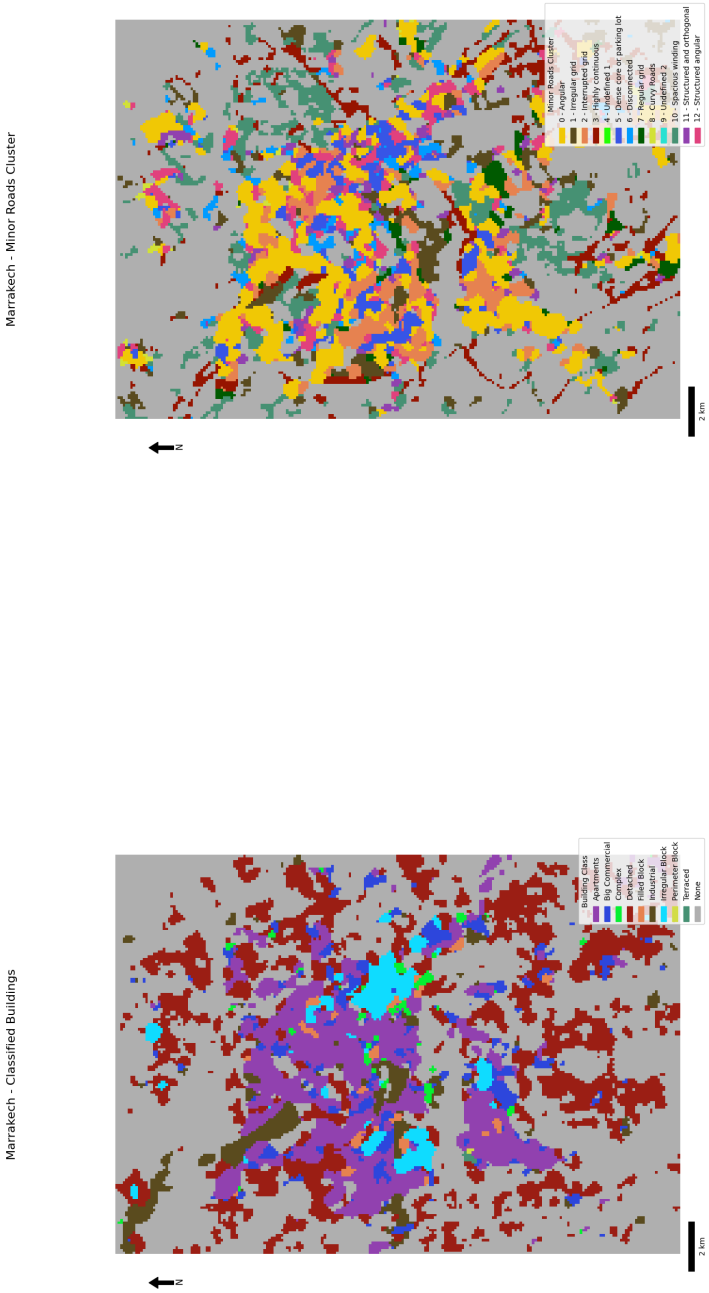
(a) Building Classes (b) Minor Roads Clusters

Figure 107 - Final typology grids for Beijing



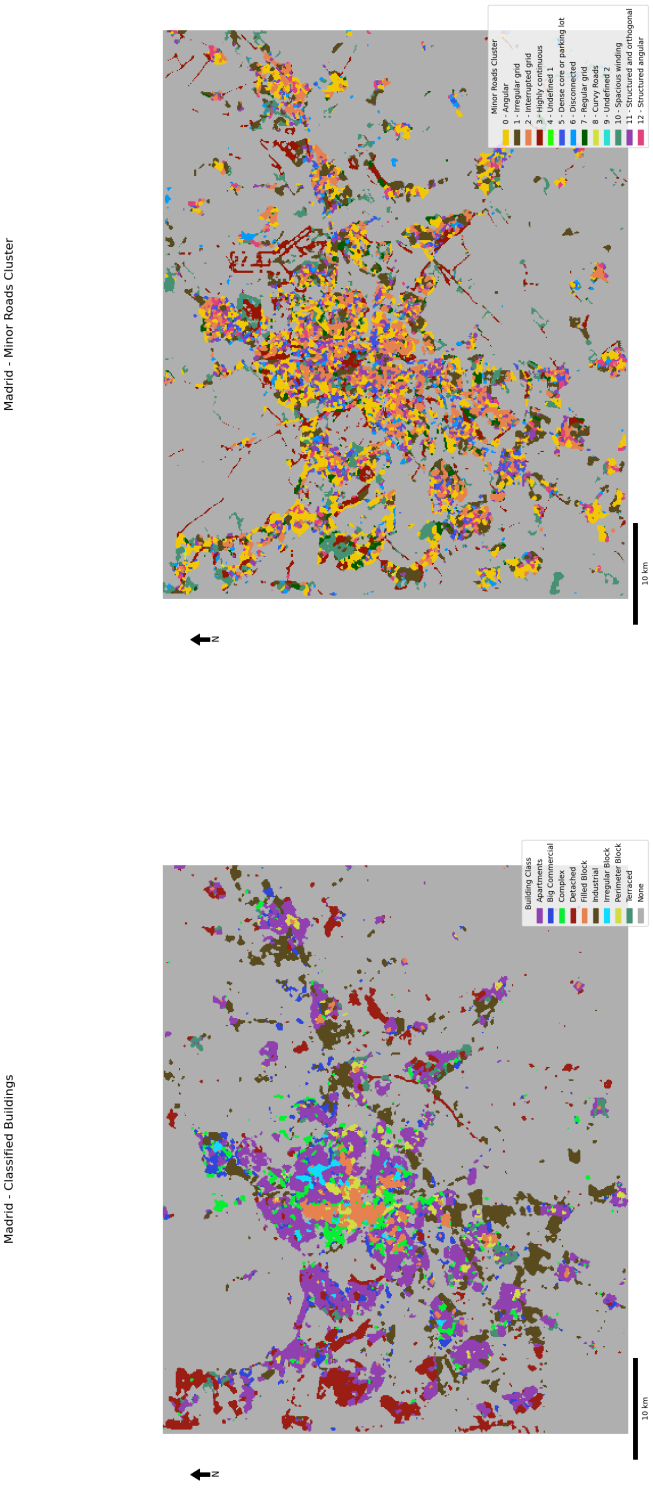
(a) Building Classes (b) Minor Roads Clusters

Figure 108 - Final typology grids for Beirut



(a) Building Classes (b) Minor Roads Clusters

Figure 109 - Final typology grids for Marrakech

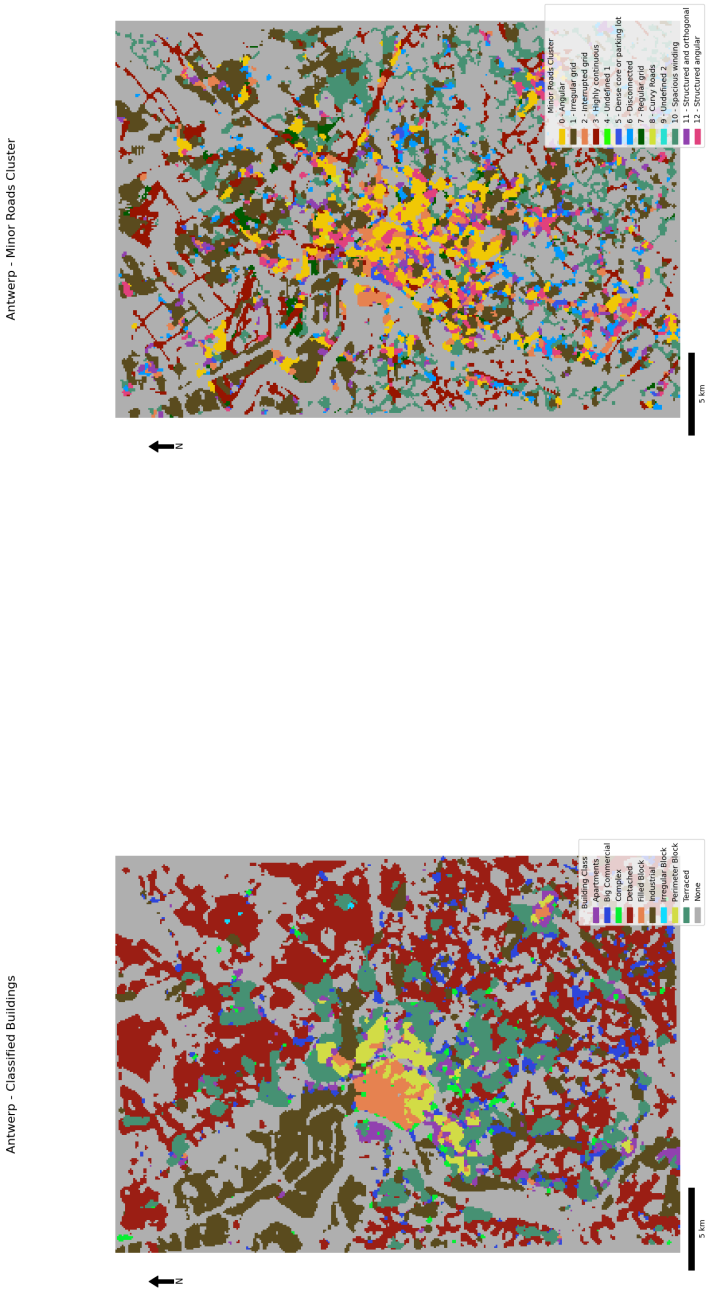


(a) Building Classes (b) Minor Roads Clusters

Figure 110 - Final typology grids for Madrid

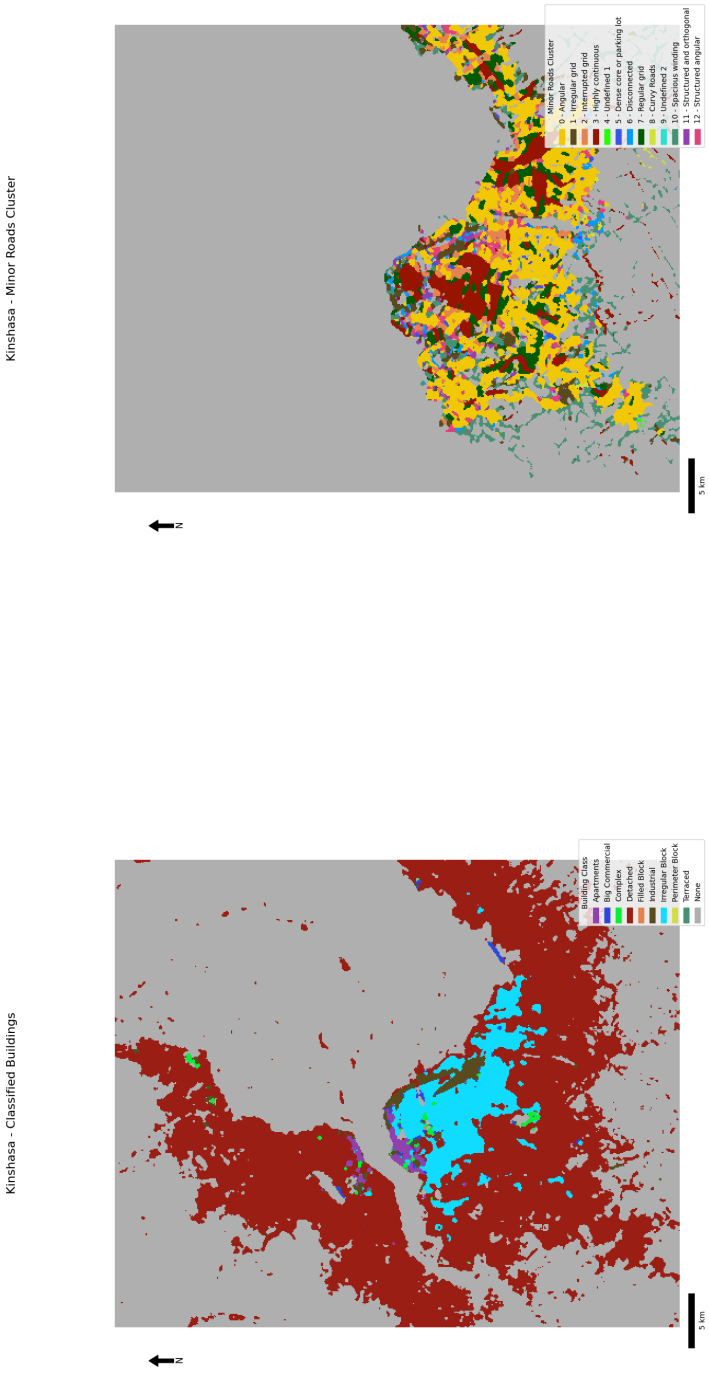
(a) Building Classes (b) Minor Roads Clusters

Figure 111 - Final typology grids for Washington dc



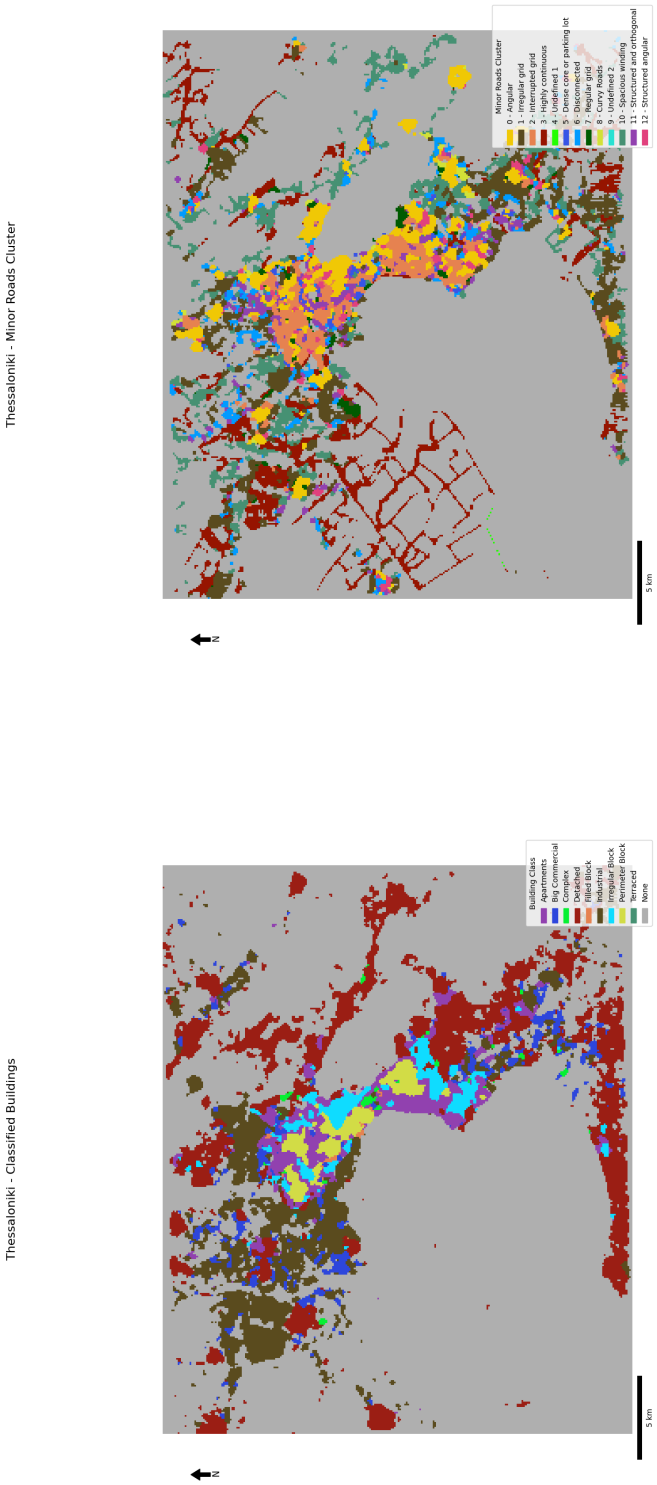
(a) Building Classes (b) Minor Roads Clusters

Figure 112 - Final typology grids for Antwerp



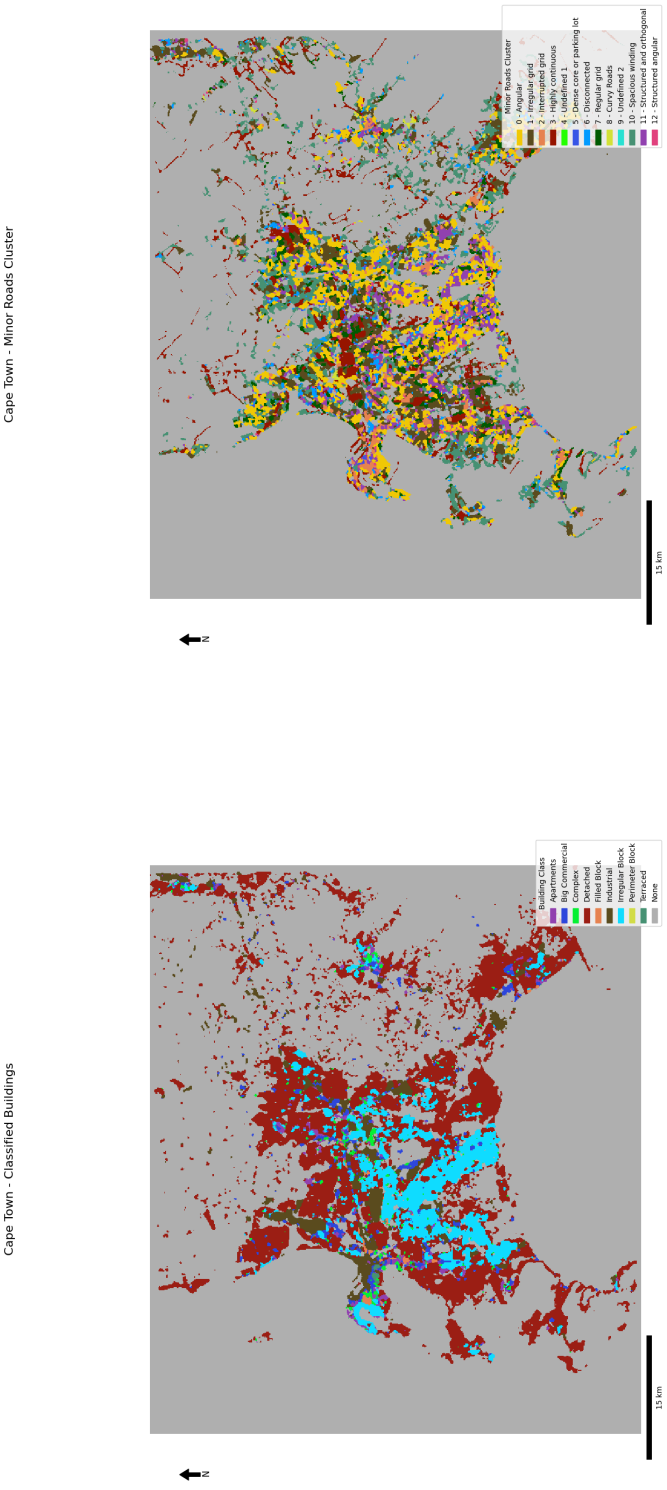
(a) Building Classes (b) Minor Roads Clusters

Figure 113 - Final typology grids for Kinshasa



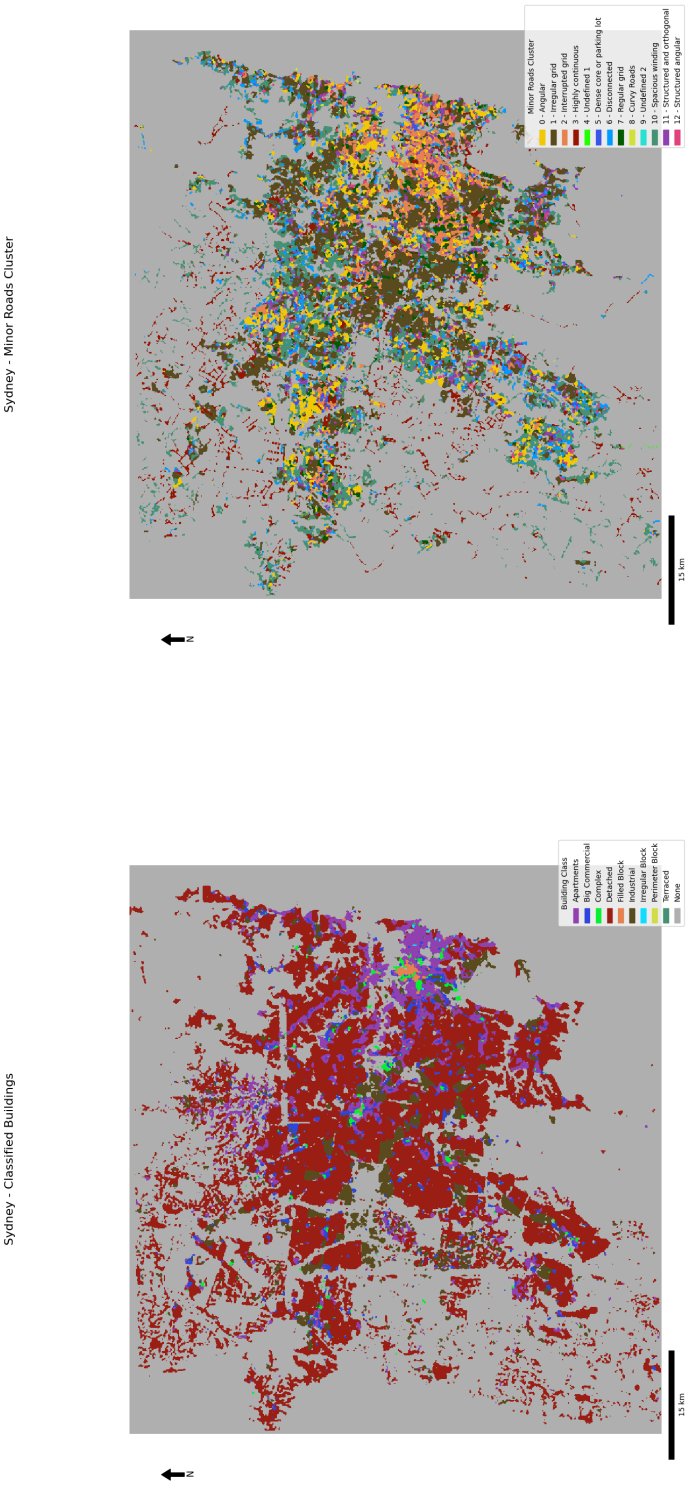
(a) Building Classes (b) Minor Roads Clusters

Figure 114 - Final typology grids for Thessaloniki



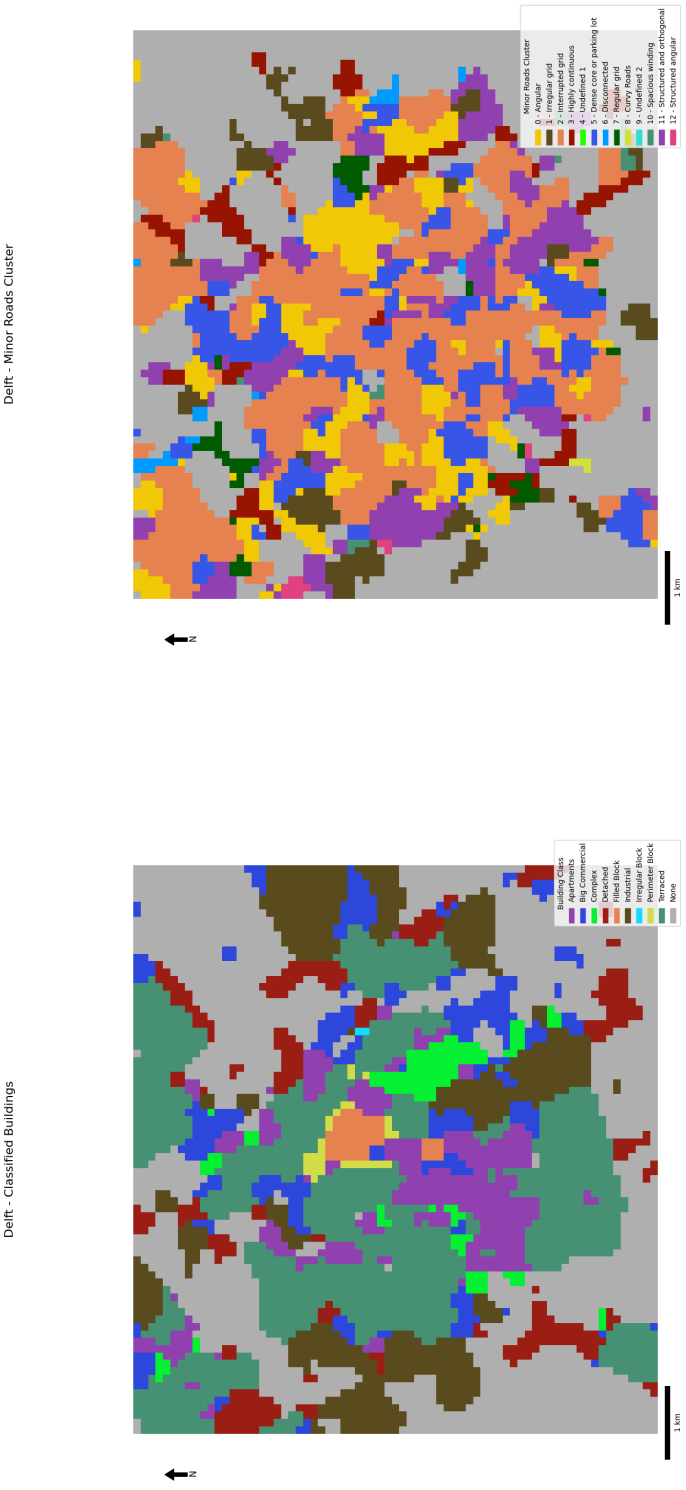
(a) Building Classes (b) Minor Roads Clusters

Figure 115 - Final typology grids for Cape town



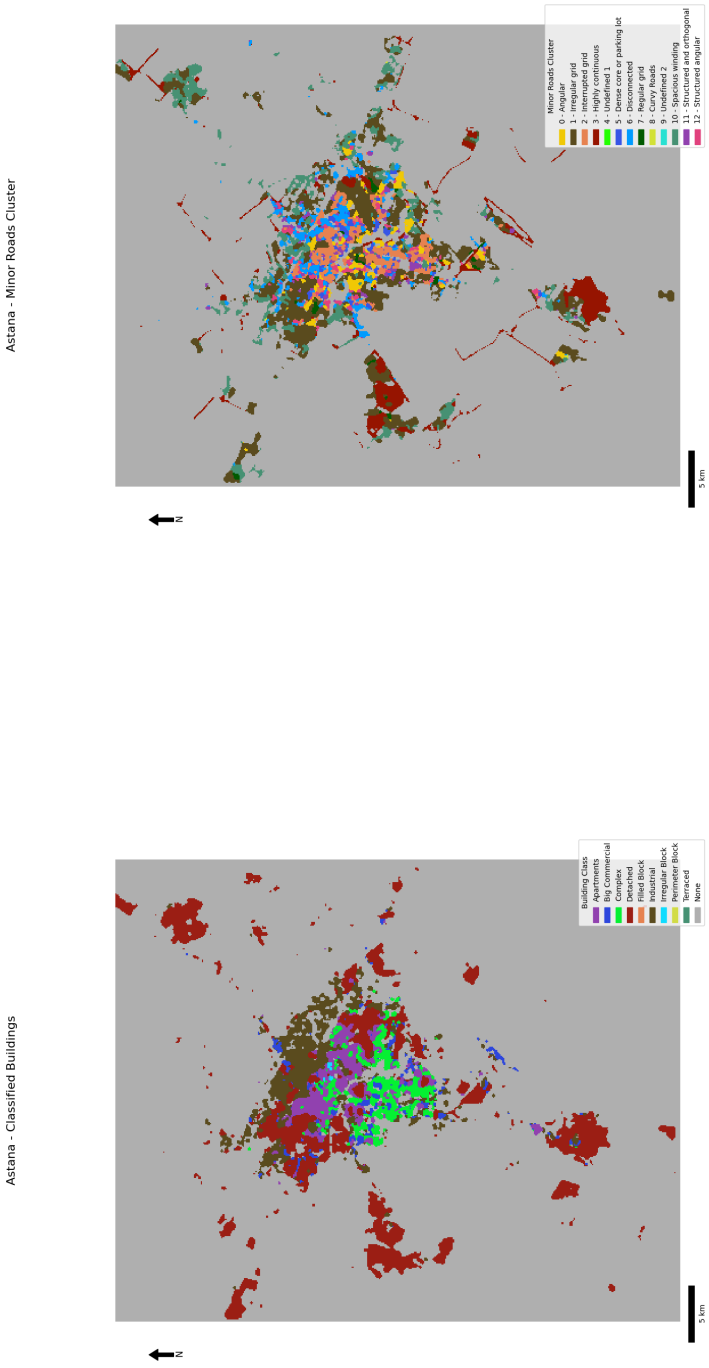
(a) Building Classes (b) Minor Roads Clusters

Figure 116 - Final typology grids for Sydney



(a) Building Classes (b) Minor Roads Clusters

Figure 117 - Final typology grids for Delft

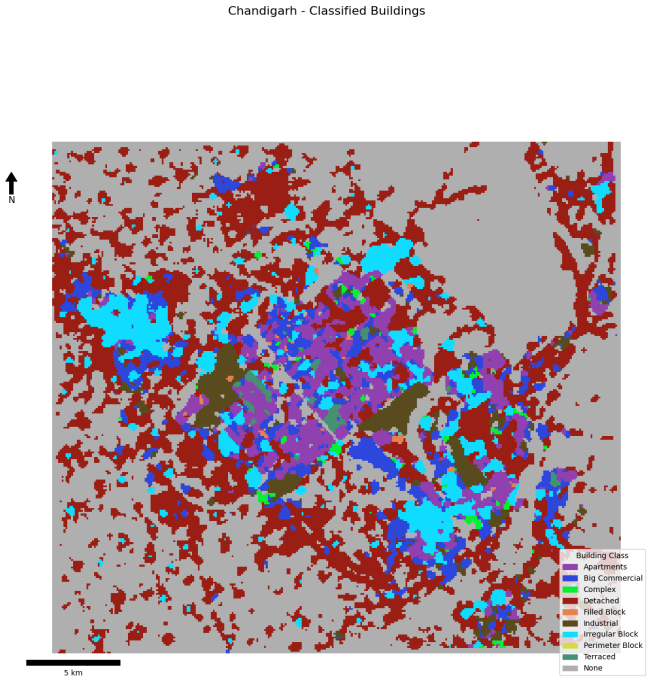


(a) Building Classes (b) Minor Roads Clusters

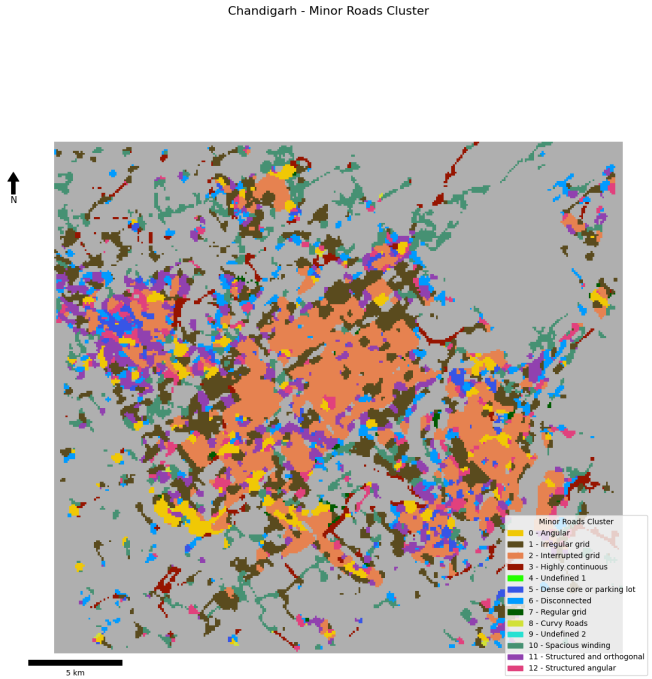
Figure 118 - Final typology grids for Astana

(a) Building Classes (b) Minor Roads Clusters

Figure 119 - Final typology grids for Mexico city

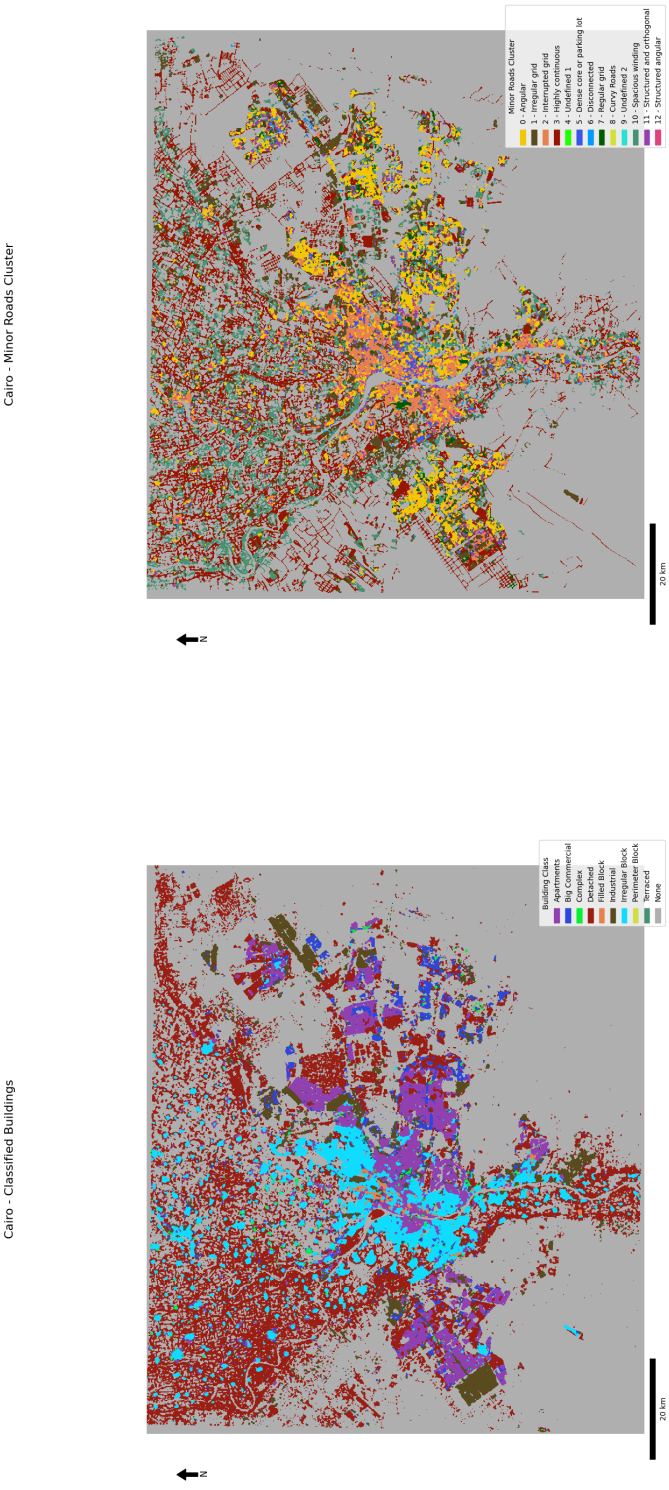


(a) Building Classes



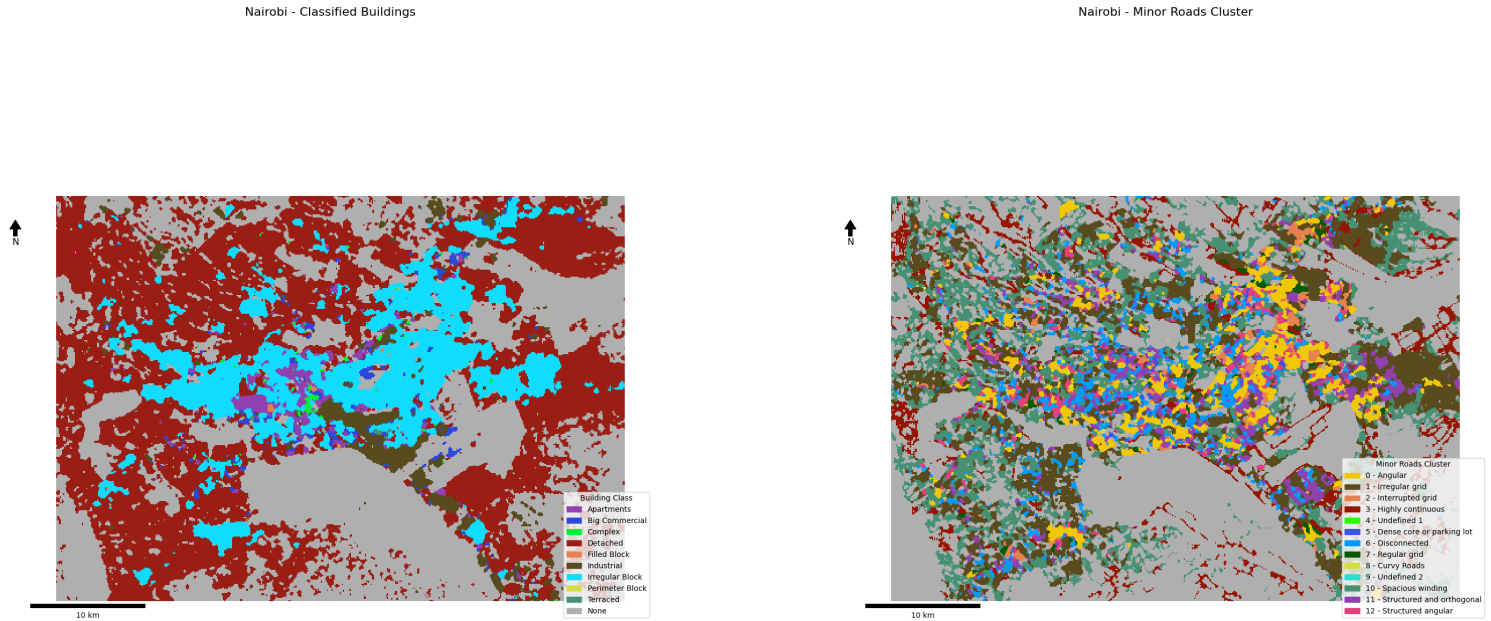
(b) Minor Roads Clusters

Figure 120 - Final typology grids for Chandigarh



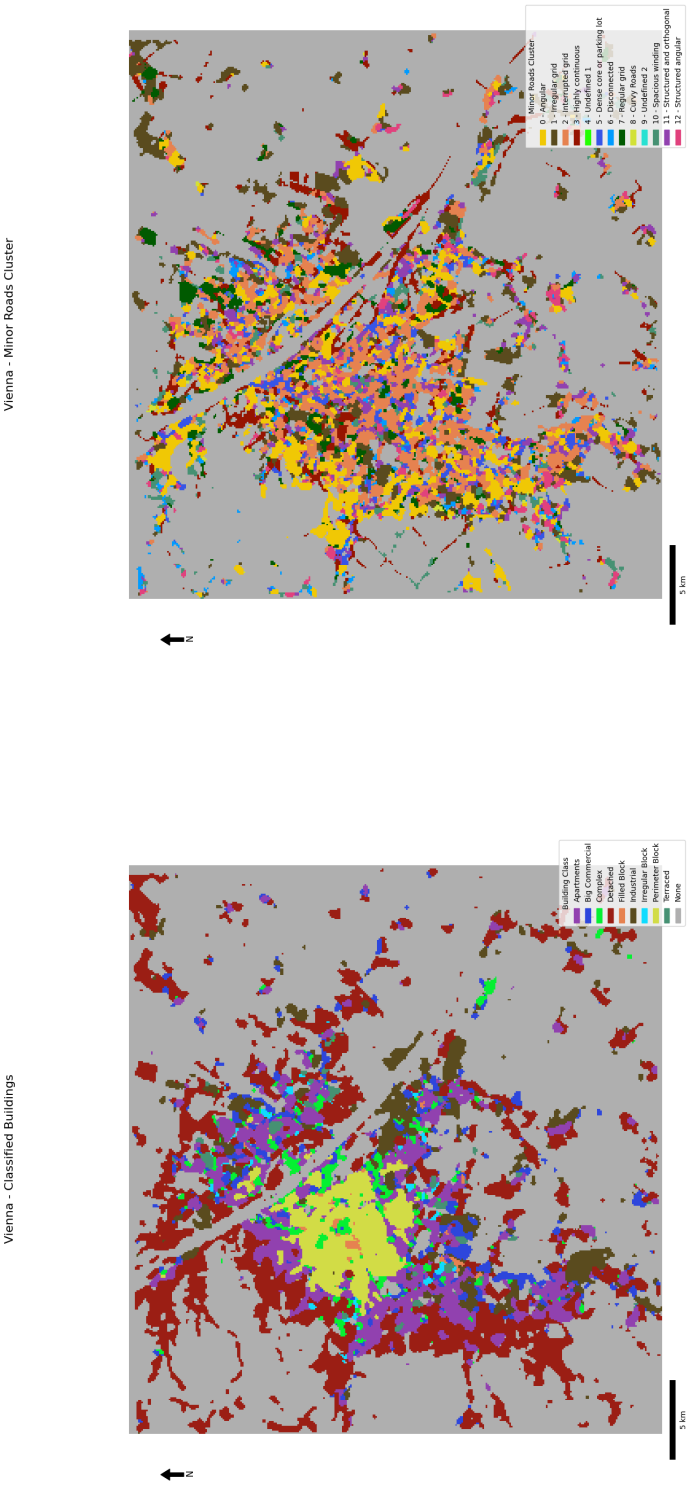
(a) Building Classes (b) Minor Roads Clusters

Figure 121 - Final typology grids for Cairo



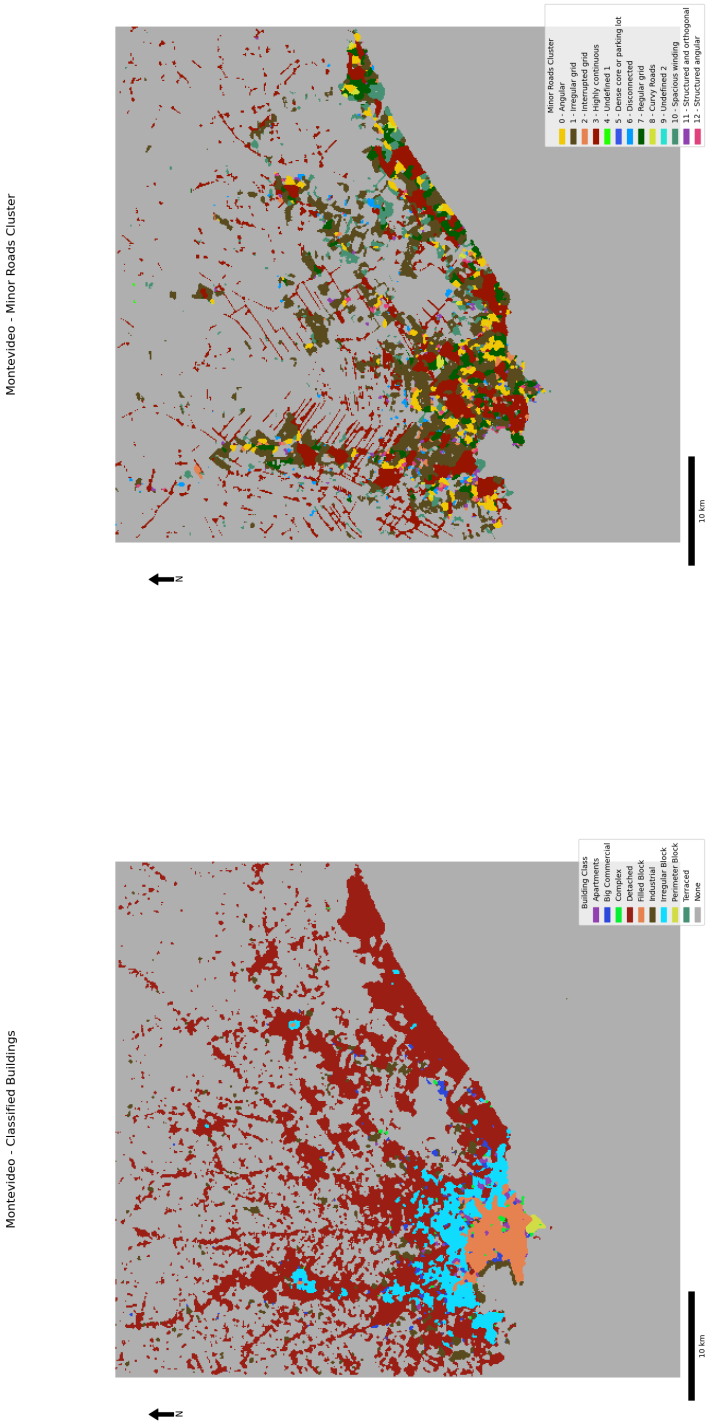
(a) Building Classes (b) Minor Roads Clusters

Figure 122 - Final typology grids for Nairobi



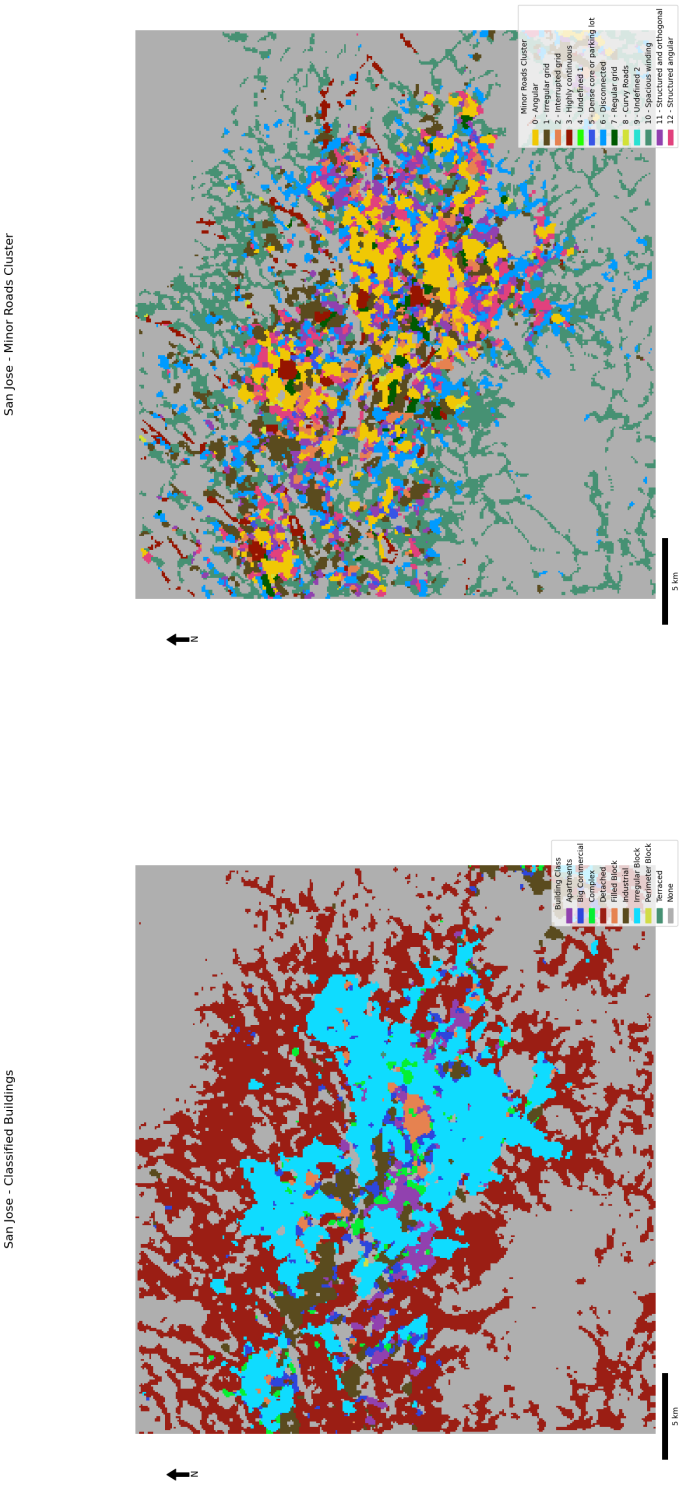
(a) Building Classes (b) Minor Roads Clusters

Figure 123 - Final typology grids for Vienna



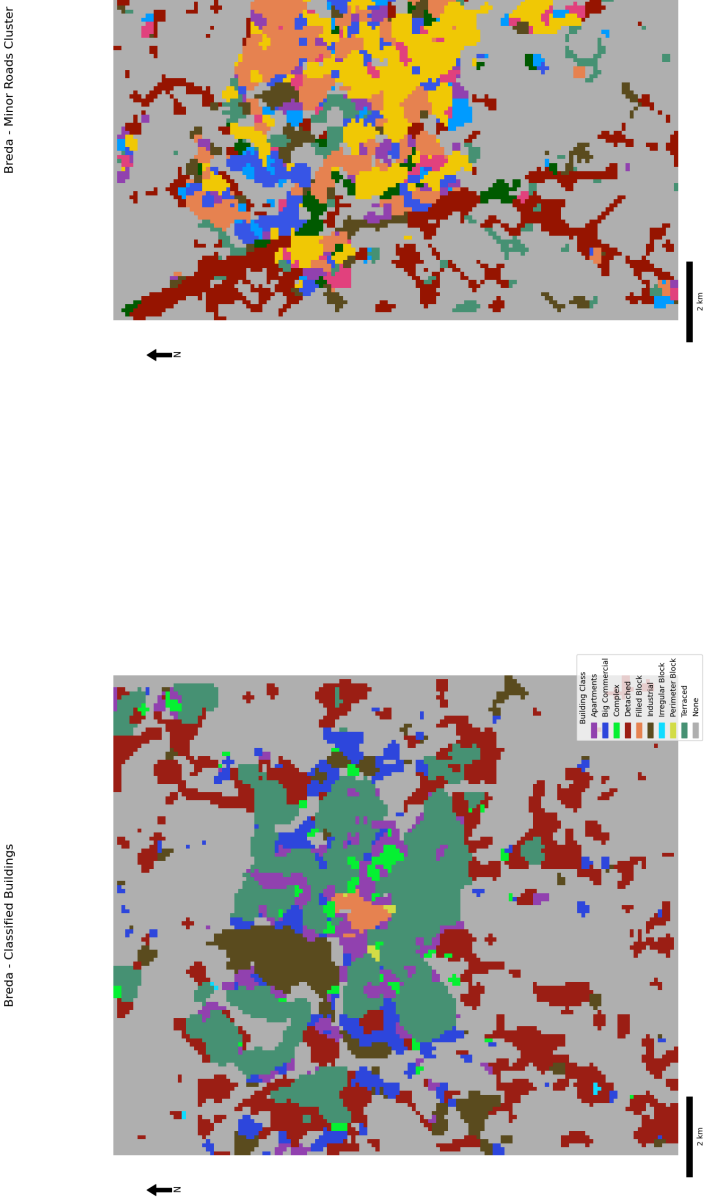
(a) Building Classes (b) Minor Roads Clusters

Figure 124 - Final typology grids for Montevideo



(a) Building Classes (b) Minor Roads Clusters

Figure 125 - Final typology grids for San jose



(a) Building Classes (b) Minor Roads Clusters

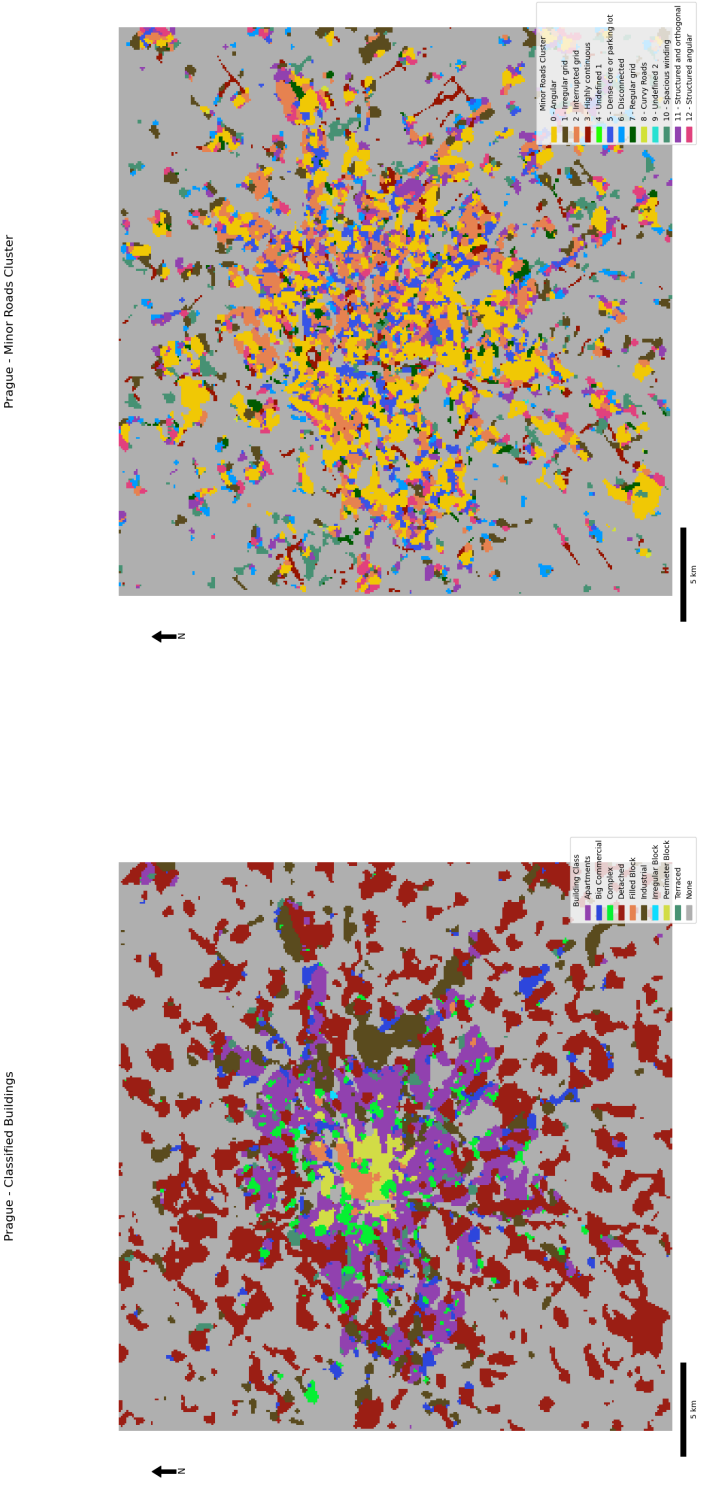
Figure 126 - Final typology grids for Breda

(a) Building Classes (b) Minor Roads Clusters

Figure 127 - Final typology grids for Rome

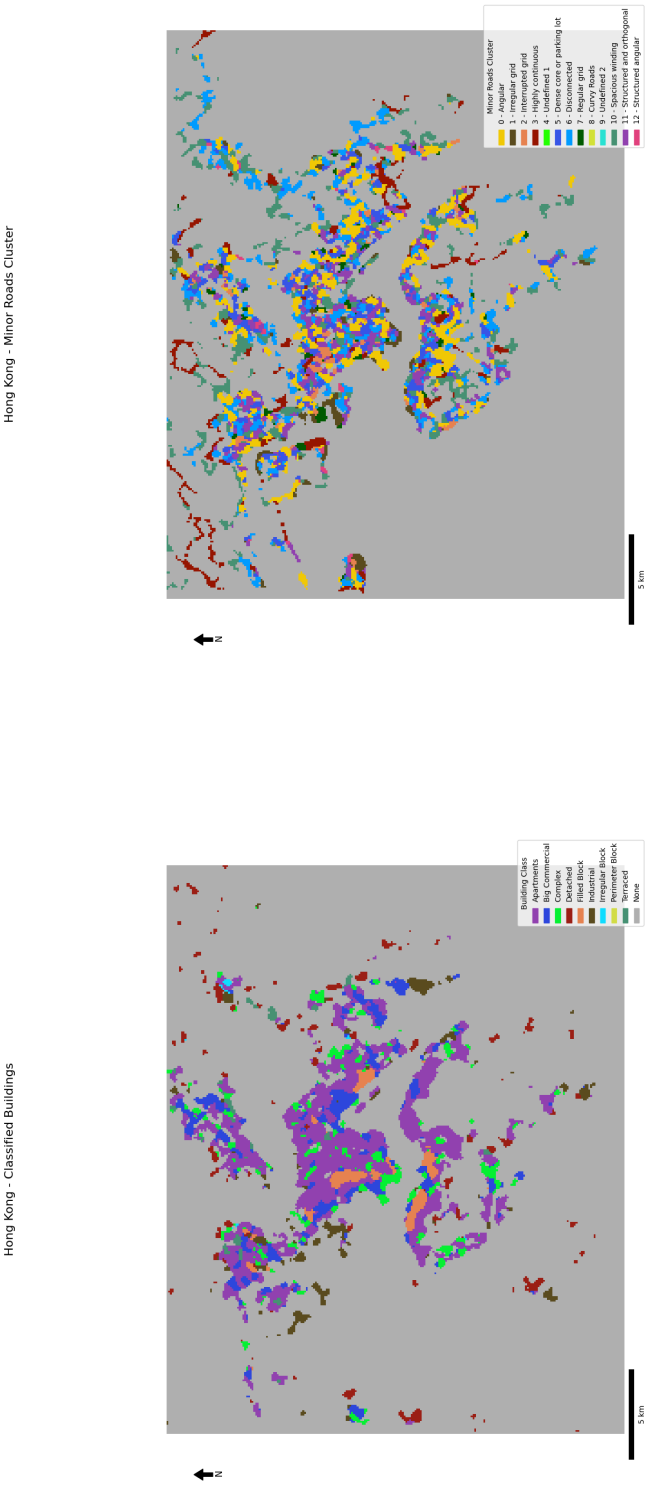
(a) Building Classes (b) Minor Roads Clusters

Figure 128 - Final typology grids for Dubai



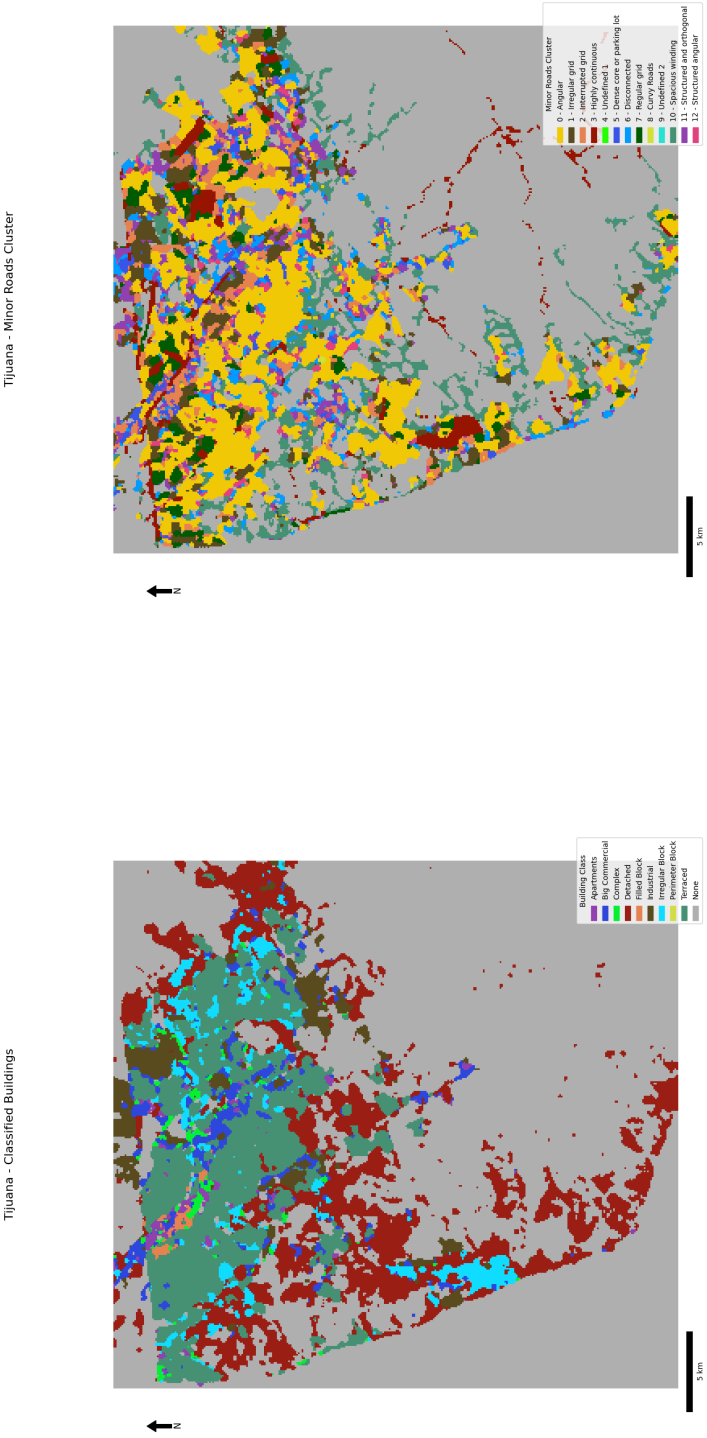
(a) Building Classes (b) Minor Roads Clusters

Figure 129 - Final typology grids for Prague



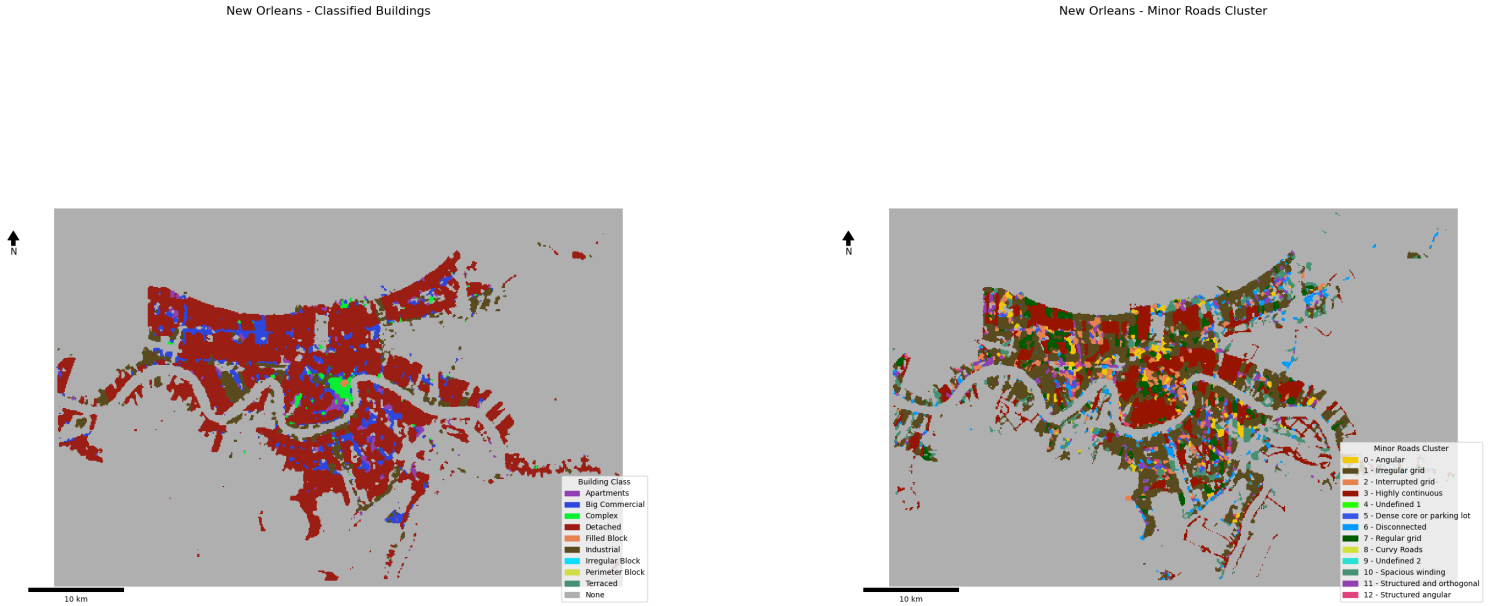
(a) Building Classes (b) Minor Roads Clusters

Figure 130 - Final typology grids for Hong kong



(a) Building Classes (b) Minor Roads Clusters

Figure 131 - Final typology grids for Tijuana

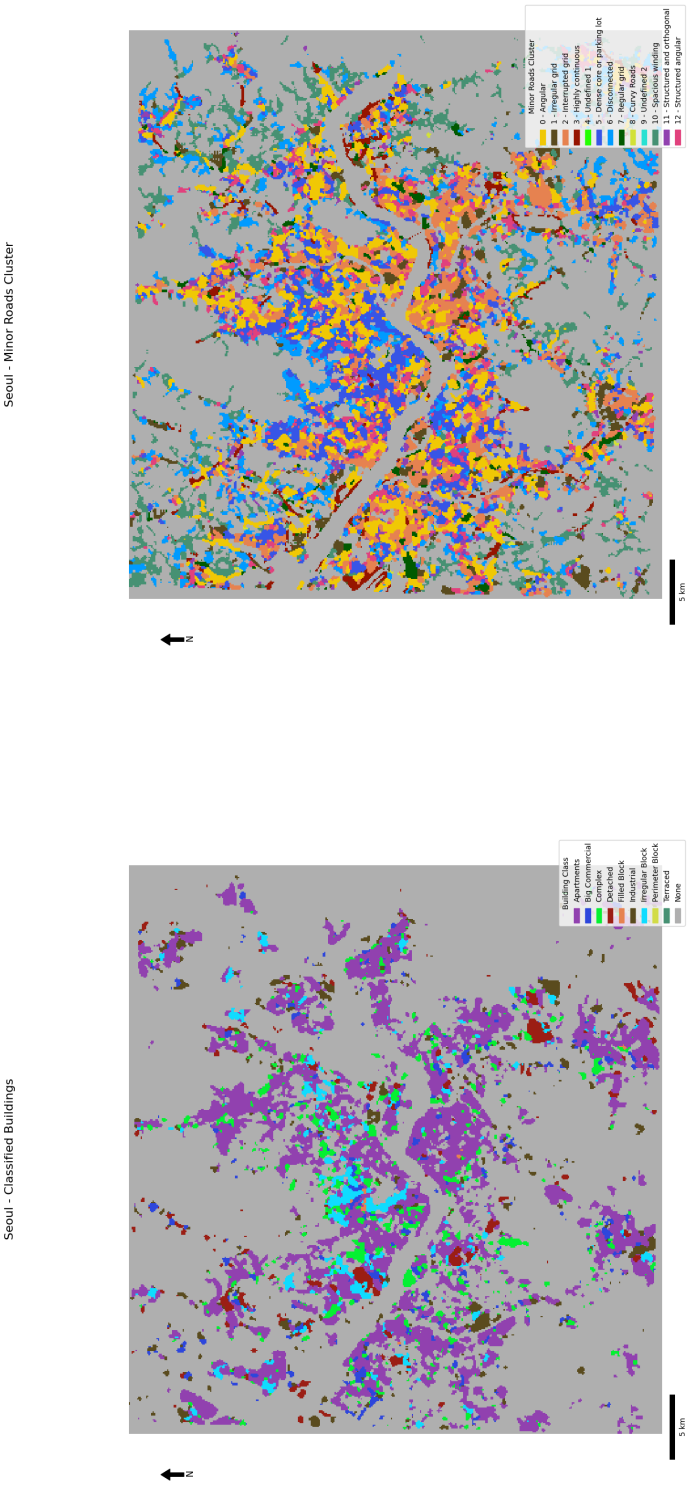


(a) Building Classes (b) Minor Roads Clusters

Figure 132 - Final typology grids for New orleans

(a) Building Classes (b) Minor Roads Clusters

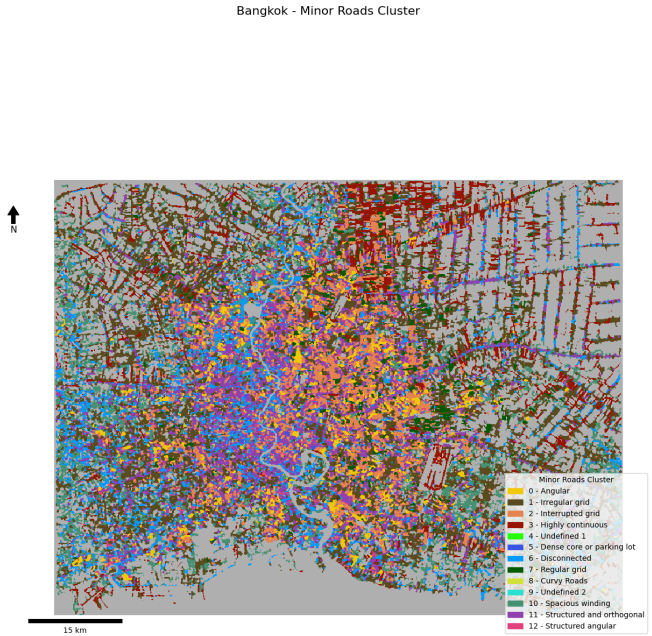
Figure 133 - Final typology grids for Lima



(a) Building Classes (b) Minor Roads Clusters

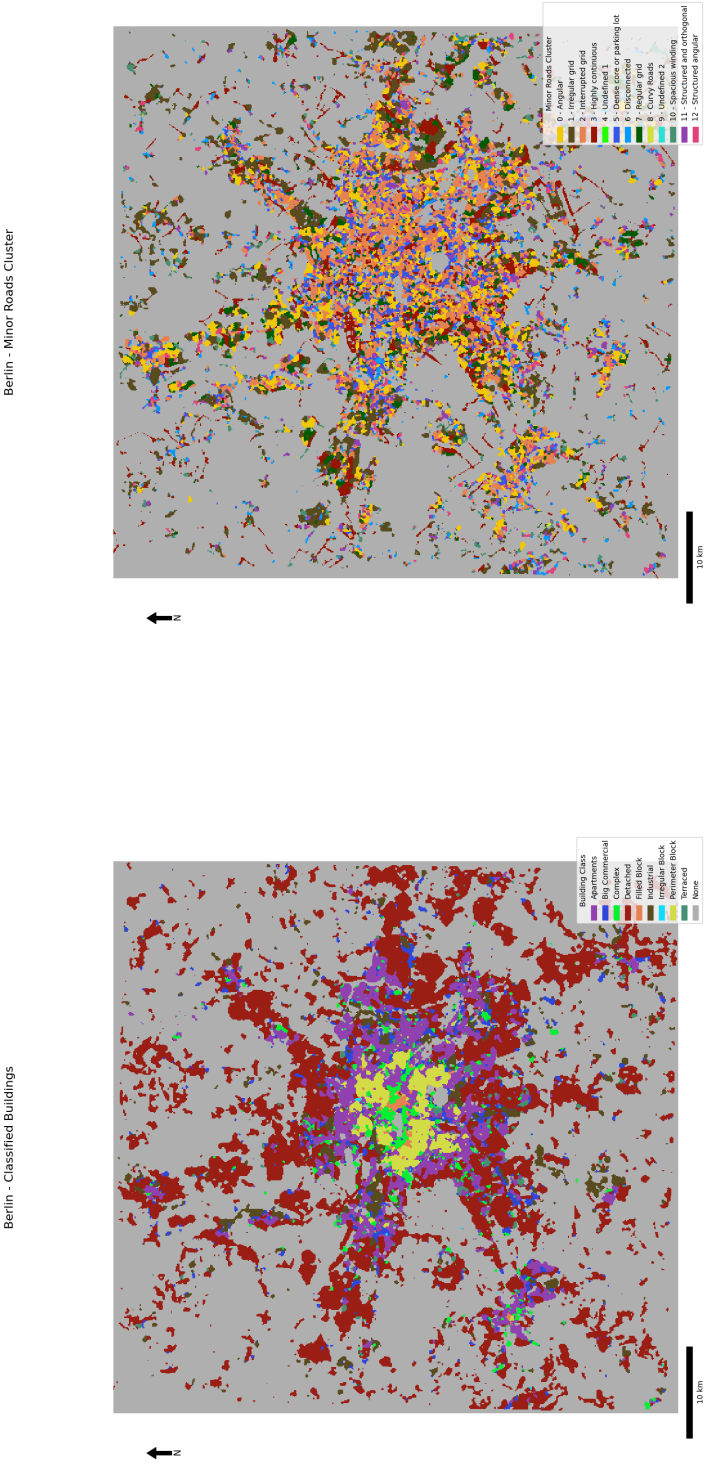
Figure 134 - Final typology grids for Seoul

(a) Building Classes



(b) Minor Roads Clusters

Figure 135 - Final typology grids for Bangkok



(a) Building Classes (b) Minor Roads Clusters

Figure 136 - Final typology grids for Berlin