Universidade do Minho
Escola de Engenharia
Departamento de Informática

Gabriel Dias Fernandes

Space Colonisation Based Procedural Road Generation

December 2018

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Gabriel Dias Fernandes

Space Colonisation Based Procedural Road Generation

Master dissertation
Master Degree in Computer Science

Dissertation supervised by
António José Borba Ramires Fernandes

December 2018

ACKNOWLEDGEMENTS

There was some fundamental help throughout the development of this dissertation which I would like to express my unconditional gratitude. To my supervisor Professor António Ramires, I would like to thank for the guidance, patience, availability and valuable suggestions during the development of this dissertation. I would like to thank Érica Valente who supported me and accompanied me closer during these difficult months. I would also like to thank to all my friends for their patience and availability when things did not go as planned. Finally, I would like to express my deep gratitude to my family for their invaluable support, help and advice during my studies.

ABSTRACT

Procedural generation of content has been studied for quite some time and it is increasingly relevant in scientific areas and in video-game and film industries. Procedural road layout generation has been traditionally approached using L-Systems, with some works exploring alternative avenues. Although originally conceived for biological systems modelling, the adequacy of L-Systems as a base for road generation has been demonstrated in several works.

In this context, this work presents an alternative approach for procedural road layout generation that is also inspired by plant generation algorithms: space colonisation.

In particular, this work uses the concept of attraction points introduced in space colonisation as its base to produce road layouts, both in urban and inter-city environments. As will be shown, the usage of attraction points provides an intuitive way to parameterise a road layout. The original Space Colonization Algorithm (SCA) generates a tree like structure, but in this work, the extensions made aim to fully generate a inter-connected road network.

As most previous methods the method has two phases. A first phase generates what is mostly a tree structure growing from user defined road segments. The second phase performs the inter connectivity among the roads created in the first phase.

The original SCA parameters such as the *killradius* help to control the capillarity of the road layout, the number of attraction points used by each segment will dictate its relevance establishing a road hierarchy naturally dependent on the distribution of the attraction points on the terrain. An angle control allows the creation of grid like or more organic road layouts.

The distribution of the attraction points in the terrain can be conditioned by boundary maps, containing parks, sea, rivers, and other forbidden areas. Population density maps can be used to supply an explicit probabilistic distribution to the attraction points. Flow-fields can be used to dictate the flow of the road layout. Elevation maps provide an additional restriction regarding the steepness of the roads.

The tests were executed within a graphic toolbox developed simultaneously. The results are exported to a geographical information file format, GeoJSON, and then maps are rendered using a geospatial visualisation and processing framework called Mapnik.

For the most part, parameter settings were intuitively reflected on the road layout and this method can be seen as a first step towards fully exploring the usage of attraction points in the context of road layout.

RESUMO

Gradualmente a geração procedimental de conteúdo tem-se tornado cada vez mais relevante, sendo maioritariamente aplicada em industrias como a dos vídeo-jogos e cinema. No que toca à geração procedimental de redes de estradas, grande parte das abordagens em torno deste tema são baseadas em L-Systems. Embora a área de aplicação dos L-Systems tenha sido originalmente para produzir modelos de sistemas biológicos, mostrou também ser um algoritmo adequado para a geração procedimental de redes de estradas.

Este trabalho apresenta uma abordagem alternativa à geração procedimental de redes de estradas que também é inspirada num algoritmo procedimental de geração de plantas, colonização espacial, utilizando o conceito de pontos de atração como base para gerar padrões de estradas. Como será demonstrado, a utilização de pontos de atração fornece uma maneira intuitiva de parametrizar um padrão de estradas desejado.

Como a maioria dos trabalhos feitos nesta área, este método tem duas fases. A primeira fase gera uma rede semelhante a uma árvore criada a partir de um ou mais segmentos iniciais da rede determinados pelo utilizador. A segunda fase trata de interligar as estradas geradas na primeira fase.

Os parâmetros iniciais do algoritmo de colonização espacial, como o *kill radius*, ajudam a controlar a capilaridade da rede, os pontos de atração que influenciam cada segmento irão ditar a sua relevância na rede geral, estabelecendo a noção de hierarquia de estradas, dependendo da distribuição de pontos de atração no terreno. O controlo do ângulo entre segmentos permite a criação de padrões de estradas tanto em forma de grelha como padrões mais orgânicos.

A distribuição dos pontos de atração no terreno pode ser influenciada por mapas de fronteira, que contêm as áreas válidas e/ou inválidas, como parques, mar, rios, e outras áreas proibidas. Mapas de densidade populacional podem ser usados para fornecer uma distribuição probabilística dos pontos de atração. Campos de forças, podem ser usados para ditar o fluxo da rede de estradas. Mapas de elevação oferecem uma restrição adicional tendo em conta a inclinação das estradas.

De um modo geral, as definições de parâmetros refletiram-se de um modo intuitivo nos padrões de redes de estradas gerados, e este trabalho pode ser considerado como um primeiro passo na exploração do conceito de pontos de atração na área da geração de redes de estradas.

CONTENTS

LIST OF FIGURES

1

INTRODUCTION

1.1 CONTEXT

The procedural generation of content comes from the human curiosity and need to describe natural phenomena with maths. Fibonacci sequences and the golden ratio are well known nowadays by being present in grow patterns in nature.

In 1968, Aristid Lindenmayer, a biologist, published a work (Lindenmayer, 1968) which presented a formal language capable of describing cellular algae growth. Later on, Lindenmayer and Prusinkiewicz used this system to generate fractals, other geometrical patterns, and 3D bushes and trees.

Given the flexibility of L-Systems, they started to be widely used in several topics among procedural generation. When the goals in the community started to be more ambitious and demanding, the procedural methods started to have performance as an important factor of discussion of study. From that need of performance, and visually realistic results, Weber and Penn (1995) was published.

In procedural generation of roads, L-Systems were also the starting point of the generation algorithms, with many works adapting and extending the original L-Systems to achieve plausible road layouts.

L-Systems allow a road layout to grow, starting from a set of user defined road segments.

Returning to tree modelling, the Space Colonization Algorithm (SCA) was proposed in Runions et al. (2007). In this algorithm the tree also starts from a set of user defined branches or segments. The growth process is based on the concept of attraction points. New segments are created based on the distribution of nearby attraction points. The process is applied iteratively, where in each iteration attraction points close by to the new segments are removed.

1.2 OBJECTIVES

The objective of this dissertation is to explore the concept of attraction points to procedurally generate road network layouts. By design SCA produces a tree layout of nodes. In order to produce plausible road layouts the original algorithm must be extended so that it creates a graph of roads as opposed to a tree like structure. This implies defining a process for the otherwise non-connected branches, to generate meaningful interconnections.

The extension of the algorithm will include the original SCA parameters and add new parameterisation to support domain specific features such as road hierarchy and angles between segments. Furthermore, road layout domain specific information such as border maps, population density, and elevation maps should also be supported. Finally, the inclusion of flow-fields may provide an intuitive way to condition road growth towards plausible road layouts.

1.3 DOCUMENT STRUCTURE

The current chapter, Chapter 1, is about exposing the theme of the dissertation and its objectives. The most relevant previous works regarding procedural road generation are presented and detailed in Chapter 2. As mentioned before, this thesis proposal is based on the SCA, namely on the concept of attraction points. This method is described in some detail in Chapter 3. Furthermore, in this chapter it will also be discussed a previous work by the same authors regarding a leaf venation generation algorithm Runions et al. (2005), and why it was discarded in this thesis.

Chapter 4 will present this works contribution detailing the extensions and modification to the original SCA, providing a graphical intuition for the major key points of the proposal. In Chapter 5 results in different scenarios will be presented to show how the concepts described in the previous chapter can be put to practice.

Finally, Chapter 6, concludes the dissertation by summarising the work done, exposing some final assessments and by proposing some future work avenues.

2

STATE OF THE ART

Procedural content generation is a subject that always fascinated the Computer Graphics community. The need to find a formula, an algorithm, to create realistic world features, started a long time ago. More than ever, a compromise between resources and realism is pursued, and from that necessity, procedural methods emerged.

This chapter aims to provide an overview on the subject of procedural road generation covering its most relevant work. L-Systems are the basis on many of the early works, therefore this chapter starts with an introduction to this algorithm. Then, it describes common characteristics of road construction and planning mostly in an city/urban environment. Finally the most influential works in the subject of procedural road layout generation are presented.

2.1 L-SYSTEMS

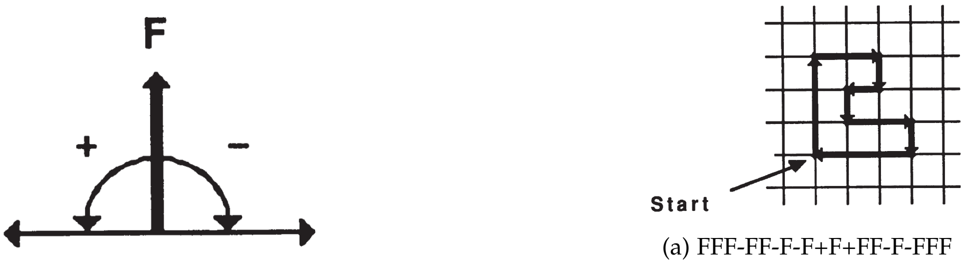
In 1968, the biologist Aristid Lindenmayer, developed a formal language capable of describing the growth of plant-like structures, called *L-systems*.

His work "*Mathematical Models for Cellular Interactions in Development*" (Lindenmayer, 1968), presenting L-Systems, is considered the most relevant and pioneer in the procedural content generation field. An L-System consists in a parallel string rewriting mechanism defined by an *axiom*, which is the initial string, and set of rules called *productions*. The axiom will be iteratively changed by the production rules and it will grow. In each iteration, the current string is scanned from left to right and when a pattern appears corresponding to the left side of a rule, that pattern will be replaced by the right side of the rule.

For instance, a L-System with a production rule shown in Equation 1, given an axiom Y, will produce XYX after the first iteration. When applying the L-System again, it will replace again the Y with XYX resulting in the string $XXYXX$.

$$Y \rightarrow XYX \tag{1}$$

Algae growth was an example of what could be described by L-Systems.



(a) FFF-FF-F-F+F+FF-F-FFF

Figure 1: A turtle interpretation applied to a possible and simple L-System. Source: Prusinkiewicz and Lindenmayer (1990)

Later in ”*Graphical modeling using L-systems*” (Prusinkiewicz and Lindenmayer, 1990), L-Systems were used in more complex ways to generate plant structures. The cellular growth proved that L-Systems could potentially describe other type of patterns. To see if L-Systems could be applied in other fields, for example, to describe visual patterns, it was tested a turtle interpretation of strings as result from the L-Systems.

A simple turtle interpretation will interpret strings as commands to move, rotate and draw in a 2D space, just like a turtle with a pen in its mouth. In each step, the turtle will have its position given by Cartesian coordinates, and an angle which describes the direction the turtle is facing. A simple example is an L-System with 3 symbols: F to move one step, $-$ to rotate 90° clockwise and $+$ to rotate 90° counter-clockwise. We can see a practical example in Figure 1.

Another example could be the *quadratic Koch island* given by the following L-System (Equation 2) with the axiom ω and production rule ρ:

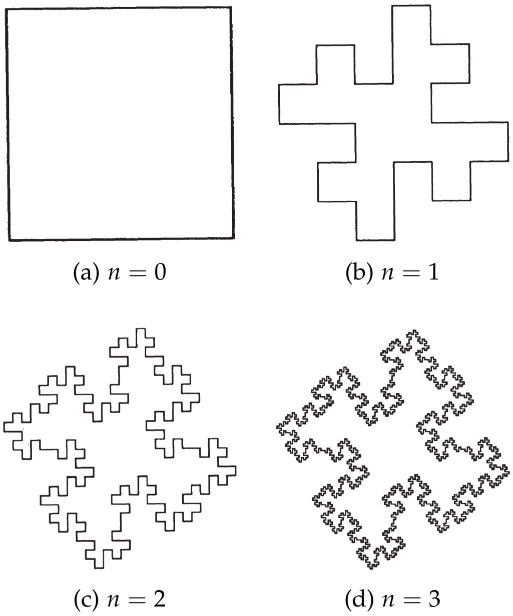
$$\begin{aligned} \omega &: F - F - F - F \\ \rho &: F \rightarrow F - F + F + FF - F - F + F \end{aligned} \tag{2}$$

This L-System, after a turtle interpretation, produces the result shown in Figure 2.

The turtle interpretation was then extended to the 3D space. Now, it is needed to have information of the turtle orientation in the three axis. So, it is needed a set of normalised vectors to represent the *up*, *left* and *heading* directions of the turtle as seen in Figure 3. Rotations are done using three rotation matrices for each of the previous presented three vectors, as seen in Figure 4.

We can see the result of this extension in Figure 5.

From that, Prusinkiewicz and Lindenmayer started examining branching structures. An axial tree is a tree with a root and terminal nodes. There is a distinction between nodes, where the axis is composed by straight segments starting from the root tree, and all the remaining segments are called lateral. The axis and all descendents constitute a branch, and a branch is itself an axial tree. In Figure 6 we have a detailed axial tree diagram, in Figure 7



(a) $n = 0$ (b) $n = 1$

(c) $n = 2$ (d) $n = 3$

Figure 2: Koch island generation (n is the iteration number). Source: Prusinkiewicz and Lindenmayer (1990)

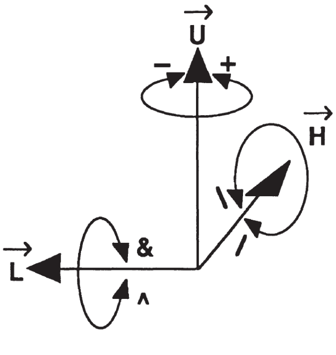


Figure 3: Turtle orientation direction control. Source: Prusinkiewicz and Lindenmayer (1990)

$$\mathbf{R_U}(\alpha) = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R_L}(\alpha) = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{bmatrix}$$

$$\mathbf{R_H}(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$

Figure 4: Rotation matrices. Source: Prusinkiewicz and Lindenmayer (1990)

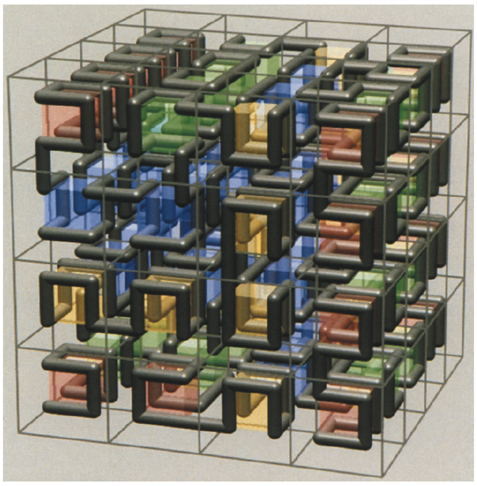


Figure 5: A three-dimensional extension of the Hilbert curve. Source: Prusinkiewicz and Lindenmayer (1990)

we have a generated tree using branching patterns and in Figure 8 we have a diagram explaining how the production rules act to generate axial trees.

A further exploration of branching structures and L-Systems allowed to generate a realistic 3D bush as seen in Figure 9.

In ”Creation and Rendering of Realistic Trees” (Weber and Penn, 1995), the generation of trees took a step further, and now, *child* branches are generated from different random seeds and parameters. This allows to have some differences among the child branches belonging to the tree.

Besides this stochastic feature in the tree generation, there is a major contribution performance-wise from this work.

Similarly as LOD [1], if the model is seen from far away, just the first level of the tree (trunk) is rendered, the closer the camera, the more levels are drawn.

This is an efficient method to render the tree without having a drop on the performance.

Although the authors stress that the parameter designation may not have the same meaning as in the botanical field of study, but instead they have a geometrical intuitive nomenclature, the results are realistic and accurate.

Results may be seen in Figure 10.

2.2 ROAD PLANNING

Well connected road networks are not a modern conception. From the Roman Empire to nowadays, an efficient and simple road network sustains and maintains civilisations, so,

[1] *Level of detail*: by decreasing the complexity/visual quality of the model, the workload is also decreased, to gain rendering performance

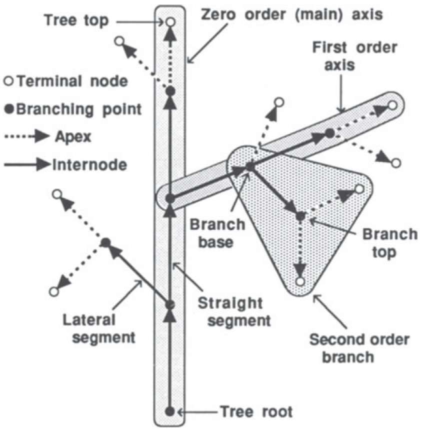


Figure 6: Axial tree diagram. Source: Prusinkiewicz and Lindenmayer (1990)

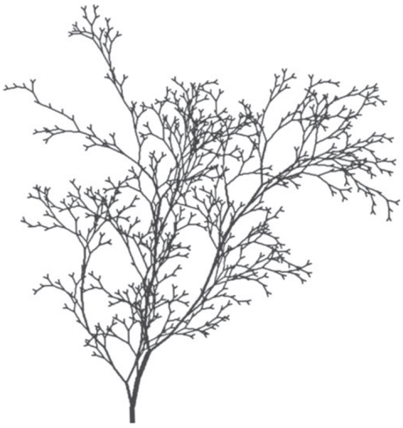


Figure 7: Sample tree generated using a method based on Horton-Strahler analysis of branching patterns. Source: Prusinkiewicz and Lindenmayer (1990)

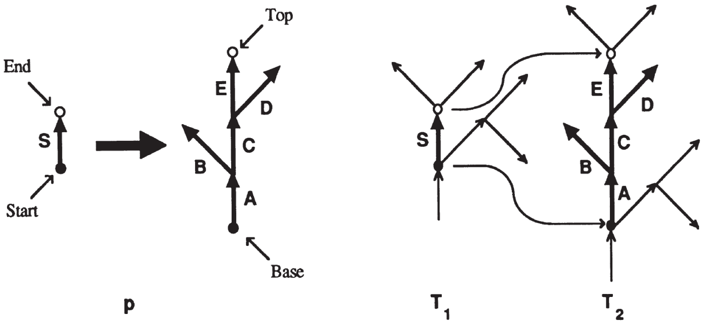
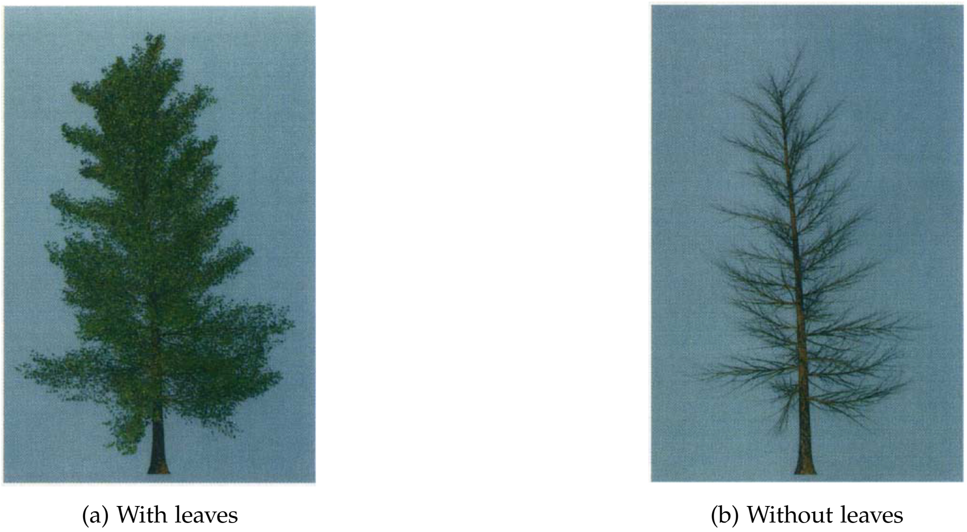


Figure 8: A tree production p and its application to the edge S in a tree T_1. Source: Prusinkiewicz and Lindenmayer (1990)

Figure 9: A three-dimensional bush-like structure. Source: Prusinkiewicz and Lindenmayer (1990)



(a) With leaves

(b) Without leaves

Figure 10: Black Tupelo tree. Source: Weber and Penn (1995)

we must expect that road planning and design is an old subject. By observing the current heavily dense world road network, we may point out similarities with neuron maps, as seen in Figure 11, blood transportation in animals and tree growth.

Three important types of city layouts are recognised: grid, radial and hexagonal.

2.2.1 *Grid layout*

This layout became known by the time of the foundation of the United States of America, and it is thought as fresh and modern. However, this design appeared *circa* 500 BC by Hippodamus of Miletus (Figure 12), as stated in ”*The Origin and Spread of the Grid-Pattern Town*” (Stanislawski, 1946), considered to be ”the Father of European Urban Planning”, known as the ”*Hippodamian Plan*”.

His city, Miletus in ancient Greece, may be one of the oldest cities characterised by this design.

Nowadays, besides the typical example of Manhattan, we have cities like Barcelona that share this design, as seen in Figure 13.

2.2.2 *Radial layout*

In the Renaissance era, in mid-fifteenth century Italy, radial layouts became popular because of the geometrical purity and the Church as the center of the state.

A famous example is the city of Palmanova (Figure 14), which 5 centuries ago, was already built as a radial city.

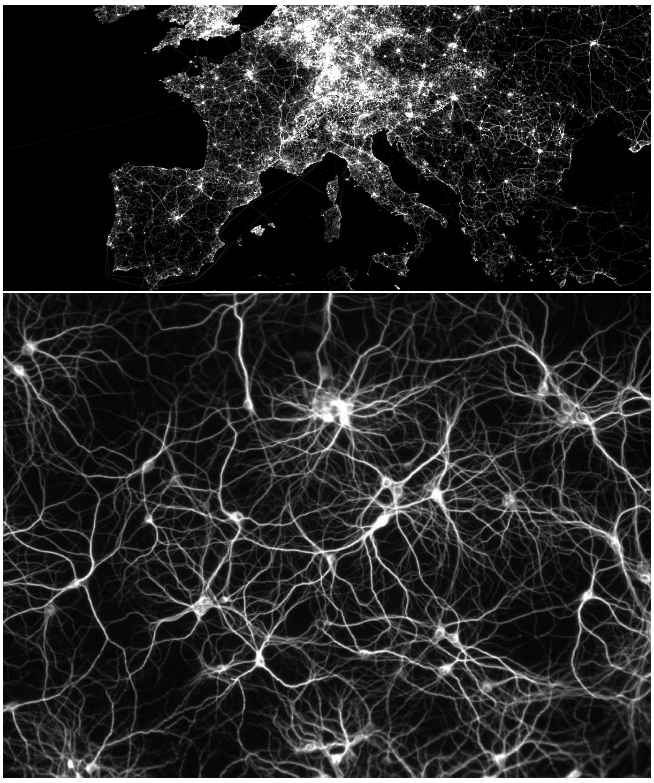


Figure 11: An Europe road network flow (top) and a neuron map (bottom). Sources: (GISgeography, 2016) and (Free Association Design, 2010)

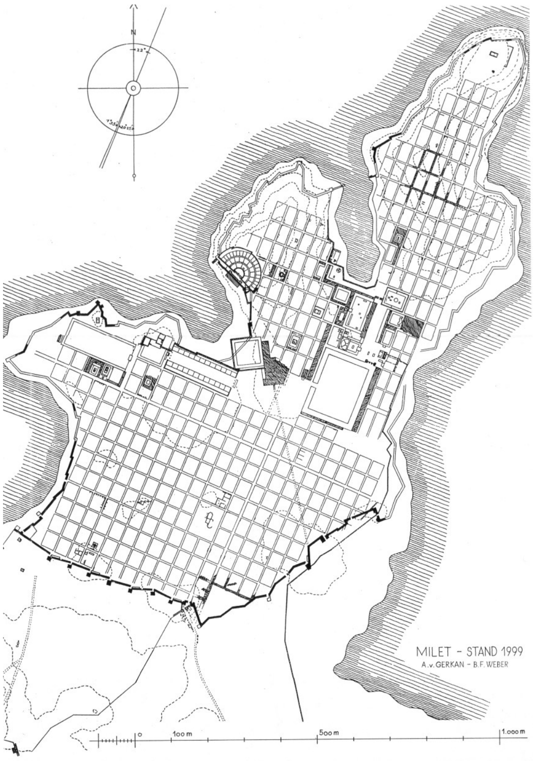


Figure 12: Miletus city plan. Source: Architekten von Gerkan and B.F. Weber GmbH

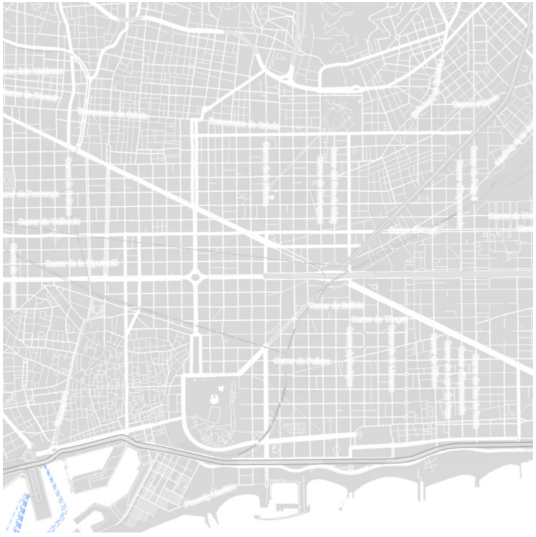


Figure 13: Barcelona city plan. Source: Google LLC (a) customised with Snazzy Maps

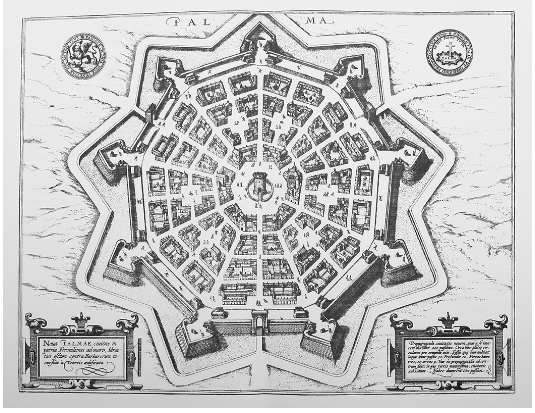


Figure 14: Palmanova city plan. Source: Hogenberg and Braun (1593)

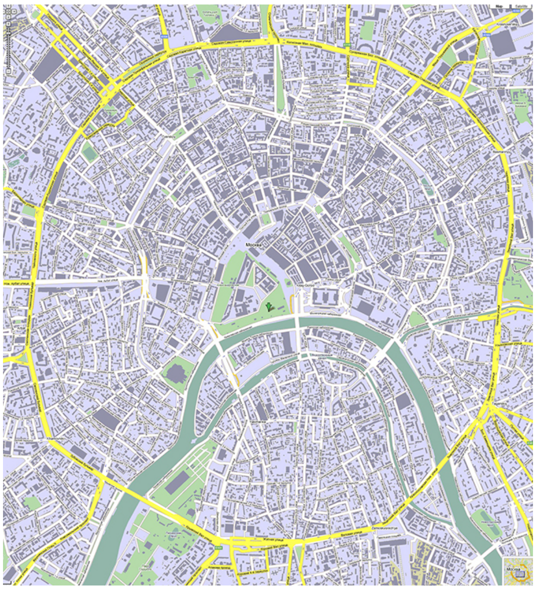
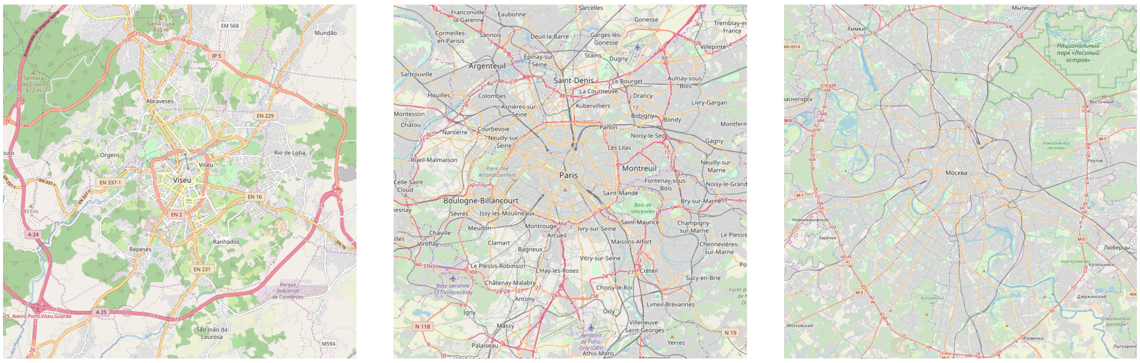


Figure 15: Moscow city plan. Source: Orange Smile (2017)



(a) Viseu, Portugal

(b) Paris, France

(c) Moscow, Russia

Figure 16: Radial road layout examples

In modern times, this pattern is still used in many of the most famous and developed cities, like Paris, France and Moscow, Russia, as seen in Figure 16 b) and c). In Portugal, Viseu is an example of this pattern as well (Figure 16 a)).

Combining both radial disposition and nesting properties, the hexagonal layout allows the optimisation of proximity centers allowing a *honeycomb* resemblance.

We can also attribute its origin to Palmanova, however, a couple centuries later, the city of Hamina (Figure 17) in Finland clearly implemented this layout.

Currently, the city of Al-Kufrah (Figure 18) in Libya, show us multiple hexagonal centers placed next to each other.

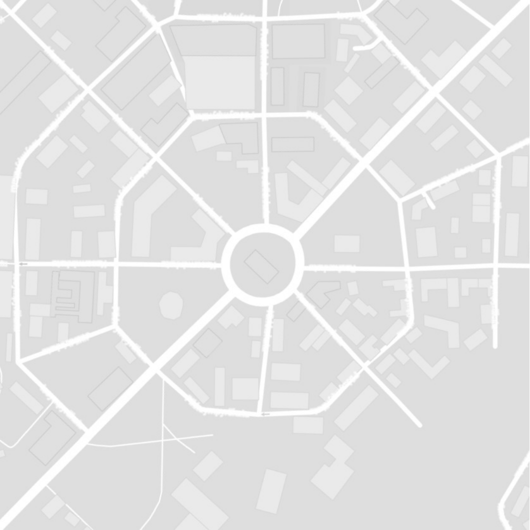


Figure 17: Hamina city plan. Source: Google LLC (b) customised with Snazzy Maps

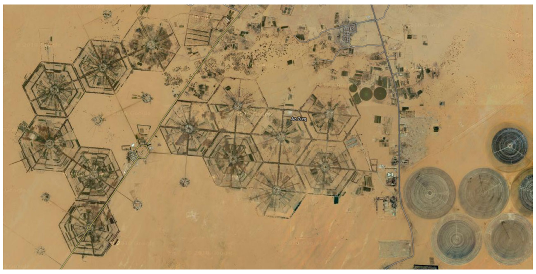


Figure 18: Al-Kufrah city plan. Source: Hexnet (2017)

2.3 ROAD AND CITY GENERATION

By noticing some similarities between roads and trees, former works in this subject start by using L-Systems to define road-like structures.

In 2001, in "*Procedural Modeling of Cities*" (Parish and Müller, 2001), is presented a system, called CityEngine, capable of generating urban environments from scratch including both buildings and road network, using an extended version of L-Systems. The input fed to CityEngine, consists in geographical and sociostatistical maps such as elevation maps, population density and land/water maps as seen in Figure 19. From this input and the given parameters, CityEngine builds both the road map and buildings. The road map is generated by connecting all the important areas, such as highly populated areas, by highways and afterwards connecting every smaller area by streets. CityEngine pipeline is shown in Figure 20.

Results can be seen in Figure 21.

One year later, "*Template-Based Generation of Road Networks for Virtual City Modeling*" (Sun et al., 2002), uses also the geographical and statistical maps as inputs, and complements them with templates to generate the roads and illegal area crossing detection.

Those templates represent the most used patterns in city road construction. A previous work, "*A pattern language*" (Alexander et al., 1977), describes over 250 patterns using a pattern language, and Sun et al. (2002) uses the most relevant ones as seen in Figure 22.

For the population based templates, it is used Voronoi diagrams. From the population density maps, the densest points are used as input points for the Voronoi diagram. Then

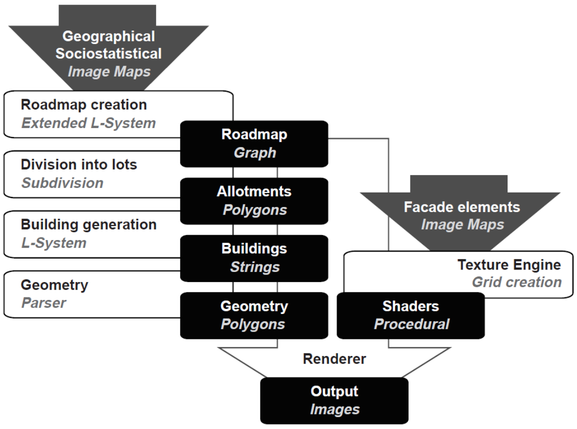


Figure 19: CityEngine pipeline. Source: Parish and Müller (2001)

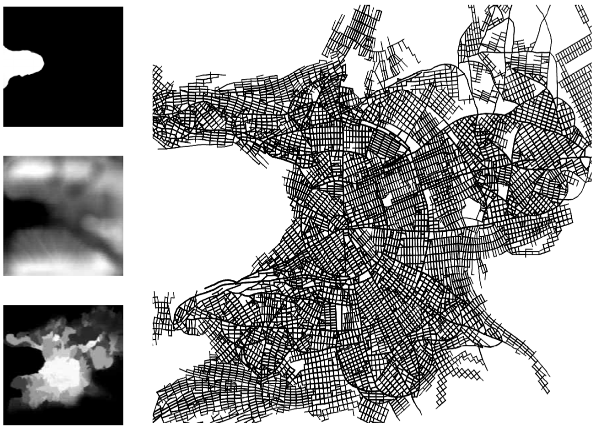
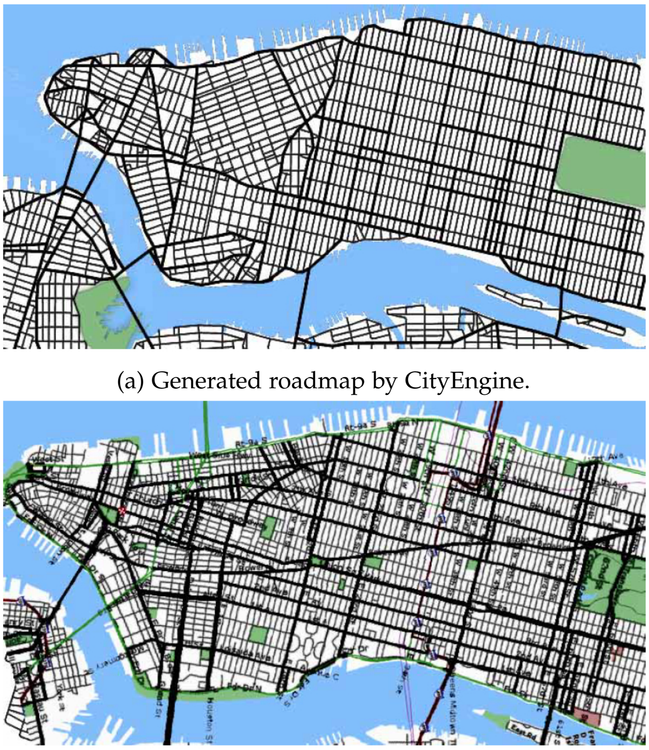


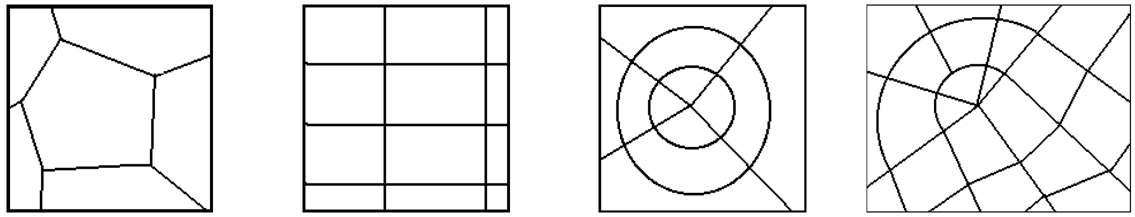
Figure 20: CityEngine input maps and generated roadmap. Source: Parish and Müller (2001)



(a) Generated roadmap by CityEngine.

(b) Manhattan, New York city map.

Figure 21: Comparison between generated roadmap and Manhattan. Source: Parish and Müller (2001)



(a) Population-based (b) Raster (c) Radial (d) Mixed

Figure 22: Templates used in *"Template-Based Generation of Road Networks for Virtual City Modeling"* (Sun et al., 2002). Source: Sun et al. (2002)

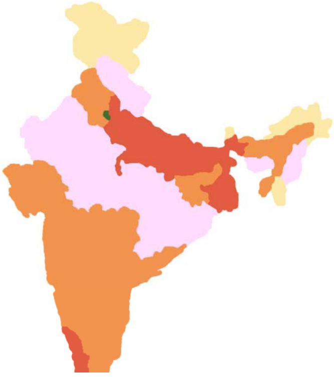


Figure 23: Population density map. Source: Sun et al. (2002)

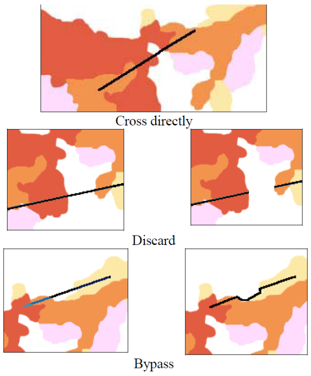


Figure 24: Illegal area crossing alternatives. Source: Sun et al. (2002)

the diagram will be consequently subdivided into more cells until a certain threshold is reached, and the edges of the cells are the resulting roads. The threshold takes into account the population and area of each cell. Through that threshold the granularity of the diagram is controlled, if there are more cells, the road network will be more dense and vice versa.

For the raster and radial templates, the algorithm starts with a single point and generates vertices until the city boundaries are reached.

The pipeline of this work consists in taking user input data, generate highways based on the templates, validate them, connect the resulting breakpoints, shape modification, extract regions, generate the streets and finally visualise and adjust the results.

The input data will be the population density maps, as seen in Figure 23, a water/land map to know where are the illegal areas, and statistical maps.

The validation of the highways consists in checking the segments that cross illegal areas. If so, three different approaches can be taken. If the distance of illegal area crossed, d_i, is less than a certain threshold, D_{max}, then the road segment is allowed and it will cross that area. May b_i be the bypass distance which is the shortest distance along the legal area that goes around the illegal area. If d_i is higher than D_{max}, then the road is discarded if the ratio between b_i and d_i is higher than the threshold ratio R_{max}, otherwise, the road will bypass the illegal area. An example can be seen in Figure 24.

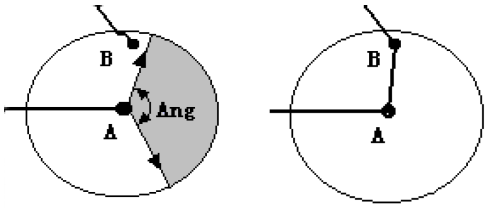


Figure 25: Connection of explicit interior breakpoints. Source: Sun et al. (2002)

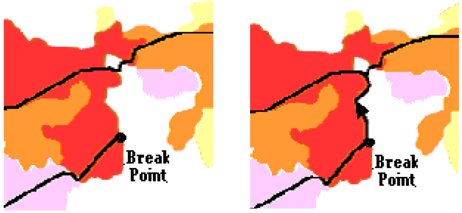


Figure 26: Connection of water boundary breakpoints. Source: Sun et al. (2002)

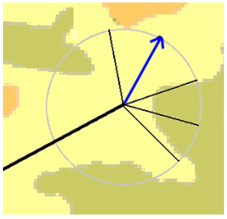


Figure 27: Search for the best elevation direction. Source: Sun et al. (2002)

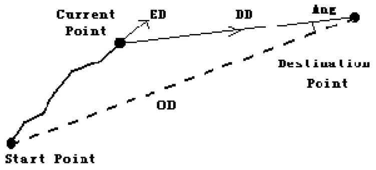
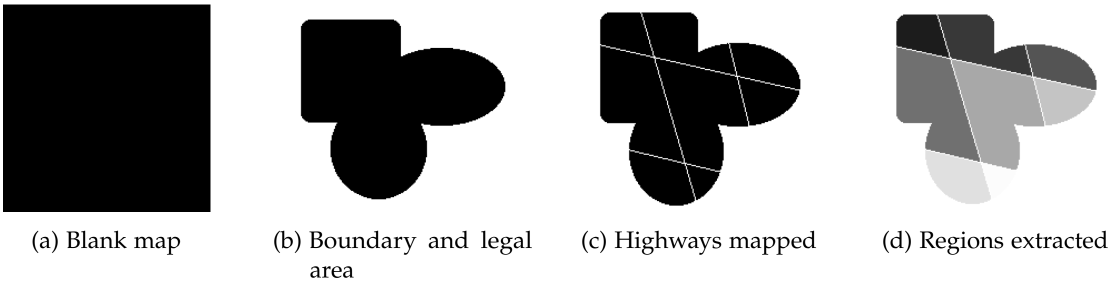


Figure 28: Computation of a new direction sub-segment. Source: Sun et al. (2002)

To connect the resulting breaking points, there are two major types: interior points and boundary points. The implicit interior breakpoints, resulting from the existence of illegal areas in a city, are connected by finding the nearest valid road. The explicit interior breakpoints, resulting from the user input, are connected by searching for a valid connection within a certain radius limited by a given angle as seen in Figure 25. The land boundary breakpoints are those that end in the land, those stay as breakpoints. The water boundary breakpoints, those that end next to a water area, start to grow another road along the coastline until it meets an existing road, as seen in Figure 26.

Another step of the pipeline, is the shape modification, which consists in changing the existing road to follow, if desired, a new path with less elevation. May OD be the original direction of the road, DD be the direction from the current starting point and ED the best elevation direction. Basically, from the current starting point of the road, many fixed-length radials are emitted, the radial which has less elevation variation is then chosen as ED, if there is more than one, it will be chosen the radial with the smallest angle between it and OD, as seen in Figure 27. Then, the new direction will be chosen taking into account the ED, DD, OD and F which is a *freedom* threshold which regulates the importance between the ED and the DD. If F value is 1, ED is the new direction of the next sub-segment, if it is 0, DD will be the new direction. An example is shown in Figure 28.

For the region extraction step, first the boundaries maps are applied to get the overall region, then highways are overlapped in the map with the breakpoints already fully connected, finally each region is extracted separately as seen in Figure 29.



(a) Blank map

(b) Boundary and legal area

(c) Highways mapped

(d) Regions extracted

Figure 29: Region extraction steps. Source: Sun et al. (2002)

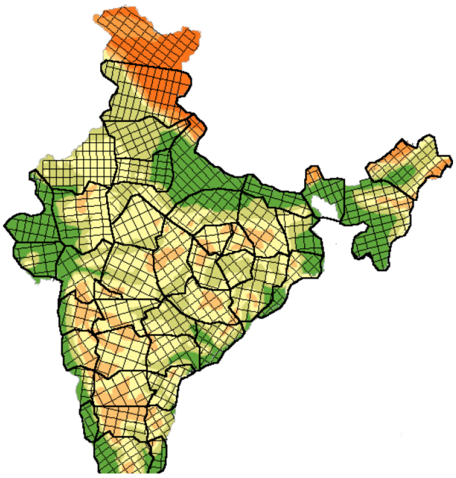


Figure 30: Result example map with a mixed road pattern. Source: Sun et al. (2002)

We can see a result of this work in Figure 30.

In 2007, *"Autopolis: Allowing User Influence in the Automatic Creation of Realistic Cities"* (Teoh, 2007) it is presented an approach that allows user interaction when generating cities. The main focus of this work, is the generation of a realistic city with a relevant infrastructure such as commercial, residential and industrial areas, airports and seaports. Concerning the generation of roads, this work also divides the generation in three road types: highways to connect the most relevant areas, main streets to connect important areas and minor roads to connect inner regions. Each tile on the map is a node with a cost associated. Elevation and water influences the cost of a tile which can be set by the user. From industrial areas, city centers, seaports and airports, a shortest path algorithm is used to create the highways between them. The shortest path algorithm takes advantage of existing roads to try to connect to them, as seen in Figure 31.

By setting a user-defined grid, main roads will be created by connecting the most developed areas in each point of the grid. The degree of development is defined by a user set threshold. There are two modes of picking the road points: a regular mode that considers the centre of the grid cell as a road point, and the irregular mode which picks a random

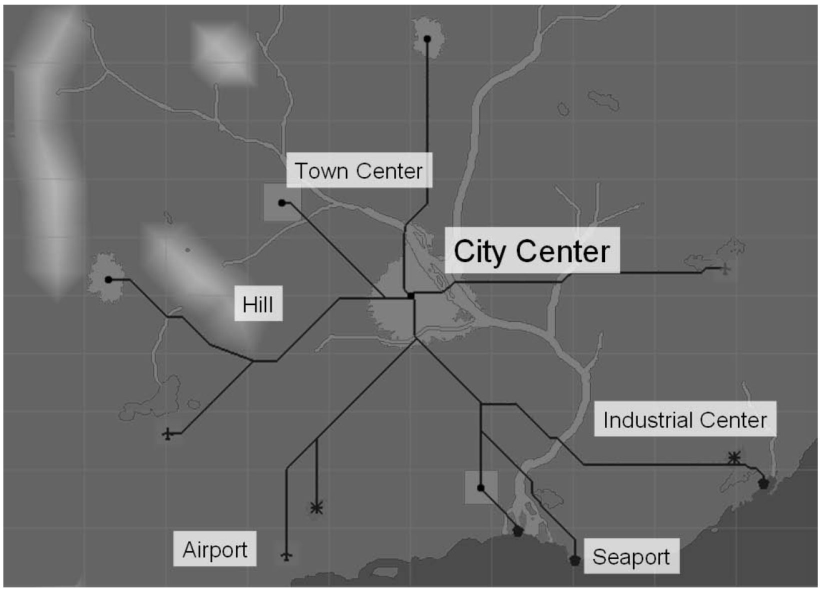


Figure 31: Most relevant areas connected by highways. The path-finding algorithm takes water and hills in consideration so highways cannot cross them. Source: Teoh (2007)

point in the grid cell. Each pair of adjacent grid points will be connected and a shortest path algorithm will create the roads. The result is shown in Figure 32.

Commercial areas tend to have a grid road network layout, also called *regular*, whereas residential areas have a *irregular* layout.

An interesting feature in this work is the use of a vector field to dictate the orientation of minor roads. This vector field may be affected by the coastline, the relief and the existing roads. The coastline automatically forces the neighbour cells to have a parallel vector which produces more realistic results that resemble coastal boulevards. The relief forces the vector to be perpendicular. Those minor roads fill the regions encircled by main streets and have a grid pattern. Each cell containing a main street has an associated vector parallel to it which will also be associated to its neighbour cells. When all the cells inside the region have a vector associated, minor roads are created in a parallel and perpendicular directions, varying the distance between them. We may see an example in Figure 33.

In the same year, another work focusing the city generation using an interactive system was published called *"Citygen: An Interactive System for Procedural City Generation"* (Kelly and McCabe, 2007). Citygen is a system that generates both buildings and road network and it is highly configurable by the user. It is completely interactive and produces results in real-time.

Again, we have two steps: a primary road generation connecting relevant points of the city, and secondary roads connecting within the regions bordered by main roads. The primary road network is represented by two adjacency graphs, as seen in Figure 34, corresponding to two distinct levels. The high level graph is the topological structure, i.e. contains the primary

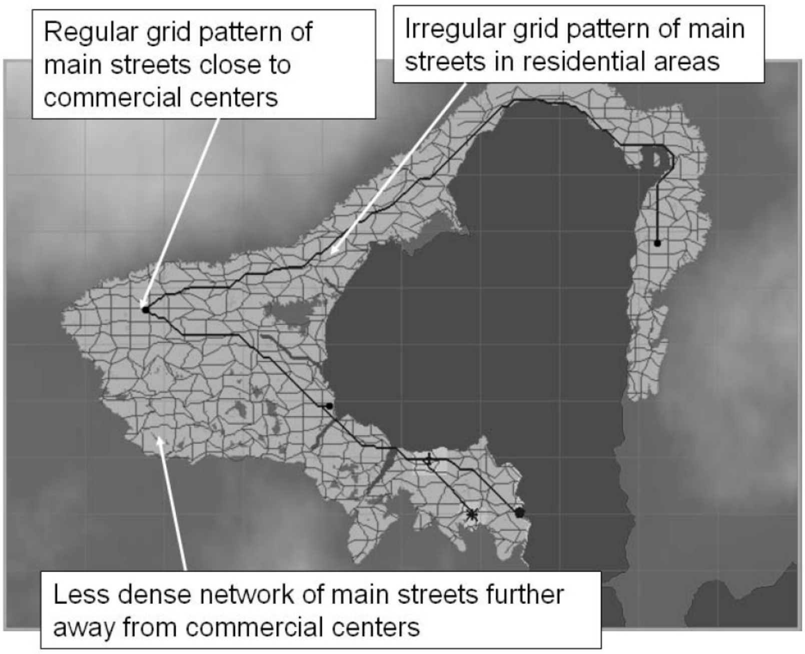

Regular grid pattern of main streets close to commercial centers
Irregular grid pattern of main streets in residential areas
Less dense network of main streets further away from commercial centers

Figure 32: Main streets connecting the most developed areas. Hybrid city with regular and irregular street patterns. Source: Teoh (2007)

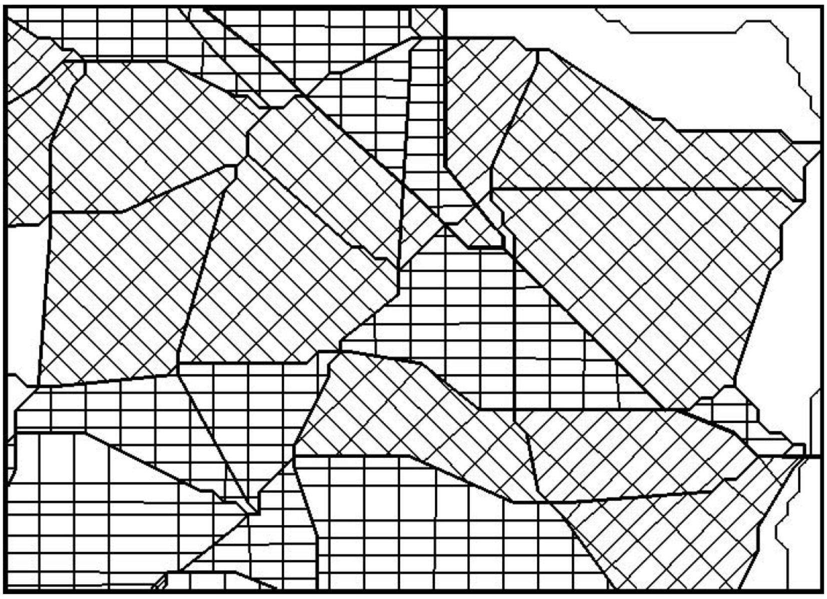


Figure 33: Minor streets within regions. Source: Teoh (2007)

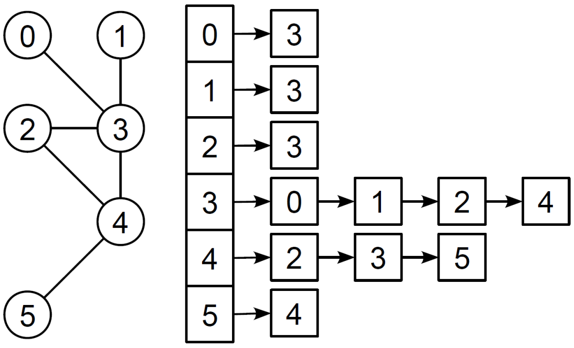


Figure 34: Adjacency graph data structure. Source: Kelly and McCabe (2007)

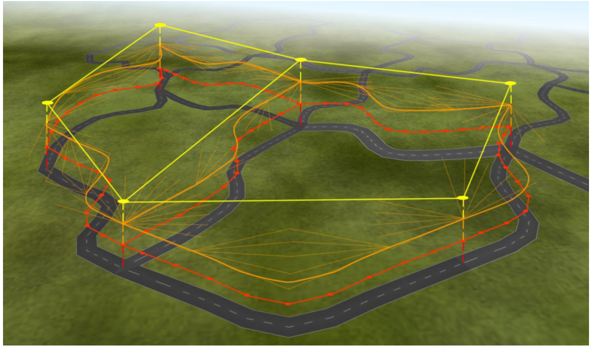


Figure 35: Primary road graph levels. Yellow: high level, intersection nodes only. Red: Low-level. Orange: interpolation spline. Source: Kelly and McCabe (2007)

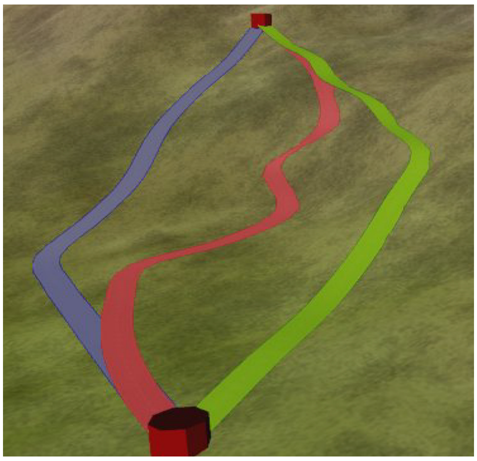
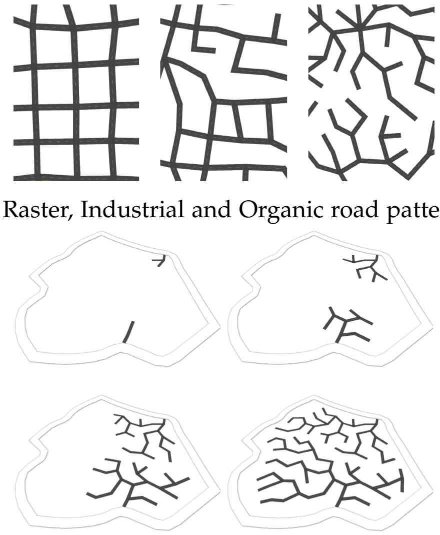


Figure 36: Adaptive roads in Citygen. Blue - Minimum Elevation, Red - Least Elevation Difference, Green - Even Elevation Difference. Source: Kelly and McCabe (2007)

road intersection nodes. The low level graph contains the nodes which define the road path. This separation optimises the system since the connectivity information is retrieved with less data. We can see an example in Figure 35.

Adaptive roads are created by sampling the connection between two control points, then the selected samples are inserted into a spline, interpolated, and inserted into the low level graph. The sampling is flexible where the user can choose between several terrain approaches: minimum elevation, least elevation difference and even elevation difference, as seen in Figure 36.

Secondary roads connect the area enclosed by primary roads, also called, city cells. The first step is the extraction of the city cells by applying a Minimum Cycle Basis algorithm to the high level graph of the primary road structure. Then the secondary roads are generated using an optimised version of L-systems that can be run in parallel, leading to a real-time



(a) Raster, Industrial and Organic road patterns.

(b) Secondary roads growth with 10, 100, 300 & 1000 steps.

Figure 37: Secondary roads characterisation. Source: Kelly and McCabe (2007)

generation. Secondary roads grow from the boundaries towards the centre of each city cell. Citygen allows to change the road segment size, the distance threshold used to connect to existing infrastructure, how many times should create a branch and if two segments should connect. This produces many road network patterns from raster to organic. An example of secondary road patterns and growth can be seen in Figure 37.

Soon Tee Teoh, author of Autopolis (Teoh, 2007), published in the next year "*Algorithms for the Automatic Generation of Urban Streets and Buildings*" (Teoh and Soon Tee, 2008). This work improves the street generation algorithm from Autopolis by allocating space for the freeway road, dividing the space for the buildings and an alternative method for curvy streets in suburban neighbourhoods. The space reservation for the roads is saved into a grid after a scan-converting phase where control points defining a road are converted to a Bezier curve. After the freeways, major streets are generated by disposing control points across a grid. If every control point is at the center of each cell, a regular street pattern will emerge. To get a irregular one, points are disposed randomly within each cell. A region is defined by being encircled by major streets, that region is then connected with minor streets. Those minor streets can have a grid pattern with different frequencies and two different orientations, axis-aligned and diagonal, or have a irregular and curvy pattern.

In "*Interactive Procedural Street Modeling*" (Chen et al., 2008), a more robust system is presented. This system presents a well-defined pipeline, as seen in Figure 38, and we may see each step result in Figure 39.

The first step of the pipeline takes into account input data as presented in previous works such as: a water/land map, a height map, a population density map and a park/forest map. From that input data, it generates a tensor field. This tensor field, which dictates the street

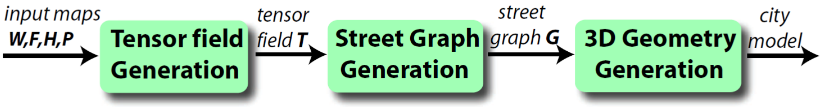


Figure 38: The modeling pipeline. Source: Chen et al. (2008)

directions and topology, is the main novelty of this work, which adds a strong mathematical base to the procedural street generation theme. The user is allowed to intervene in every step, both in the tensor field and in the street network, shaping each one to meet a desired result. Even though it is an interactive system, it produces high quality results without user interaction, by taking into account city details such as water boundaries, where roads should follow it and adding Perlin noise (Perlin, 1985) to make it more organic.

Weber et. al. in Weber et al. (2009) propose a 4D simulation, including time to simulate the growth of cities. Regarding street generation their concern is that the street layout must make sense at any time t. Therefore, their strategy is based on expansion of the previous street layout in a two stage process: expanding major streets and filling quarters with minor streets. To determine which streets to grow at any particular time a probabilistic approach is used based on the distance to the nearest growth centre. Direction, length, and other street parameters are chosen based on the street pattern which can be radial, organic or grid based. Figure 40 shows how the different areas are located and how do they change over time. Figure 41 shows a generated city at three different times.

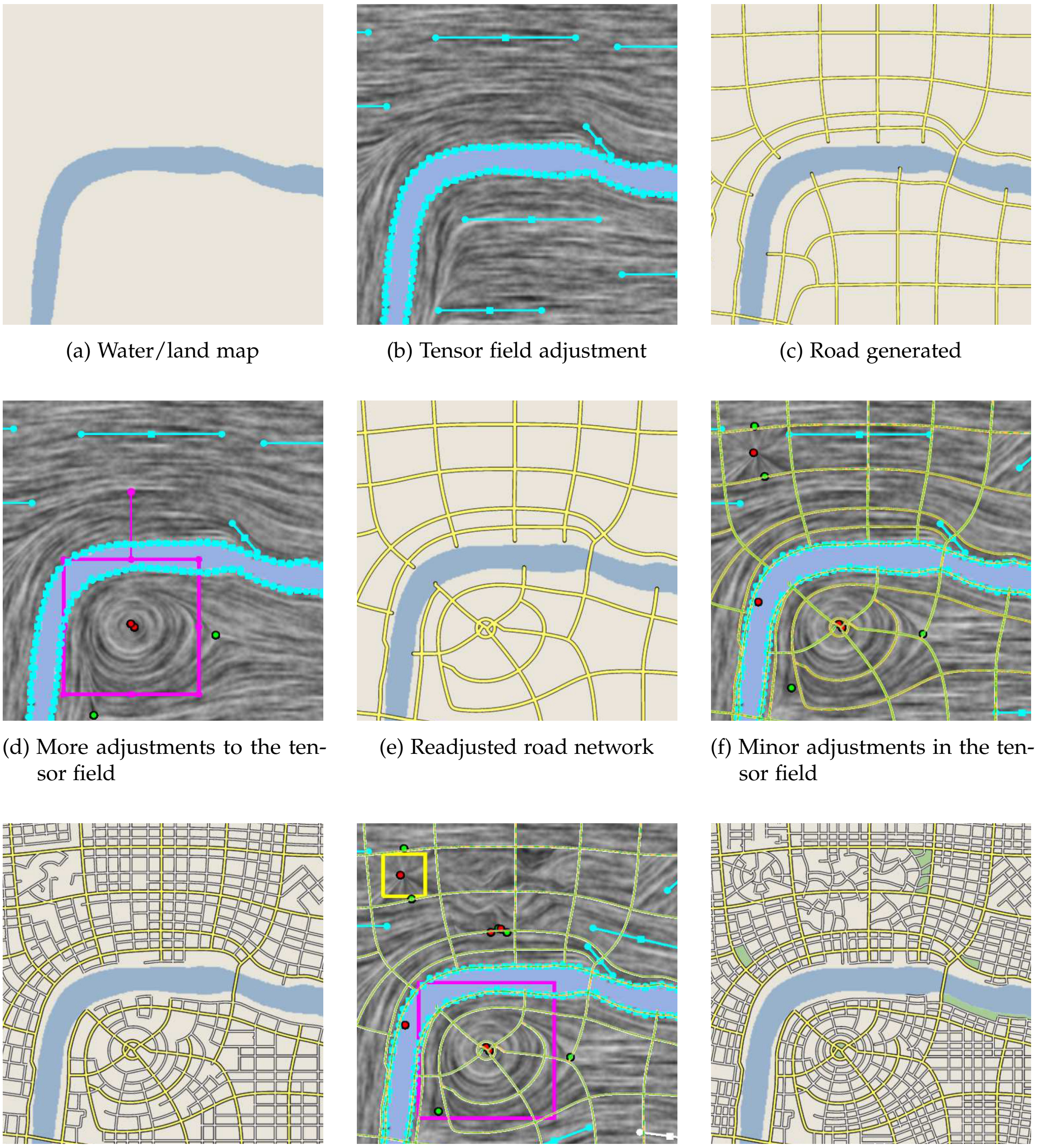
In "*Automatic Generation of Cities in Real-time*" (Egges and Hausmann, 2010), is presented another system capable of generate cities in real-time. Concerning the road generation, in this work, roads are generated after the generation of buildings. Buildings are placed in given vectors or by filling polygons, and then roads are generated following the margins of those areas and inside them to connect the buildings within.

In "*Procedural Generation of Roads*" Galin et al. (2010), is presented an algorithm capable of generating a single road between a starting and an ending point. Unlike the previous works, this article focus in generating a single procedural road in a complex environment, such as the countryside. The countryside is a perfect case study to test this approach due to the existence of rivers, forests, slopes which are not trivial to solve.

This approach uses a shortest path algorithm to minimise the distance between the start and end points, and takes into account the previously stated natural obstacles. The shortest path algorithm is executed after turning the terrain into a discrete set of points in a 2D grid.

Each point has a set of points that can connect to. This set is called *neighbourhood*. The point connectivity is defined by the size of the neighbourhood. Another term to call to the neighbourhood is *path segment masks*.

Each mask has a number associated, k, meaning that neighbourhood has a range of $[-k, k]$ in both 2-D axis. The higher the k, the more neighbours it has, and the angle between all



(a) Water/land map

(b) Tensor field adjustment

(c) Road generated

(d) More adjustments to the tensor field

(e) Readjusted road network

(f) Minor adjustments in the tensor field

(g) Minor roads generated

(h) Adjustments in the tensor field

(i) Regenerated road network with changed minor roads

Figure 39: Modeling steps sequence. Source: Chen et al. (2008)

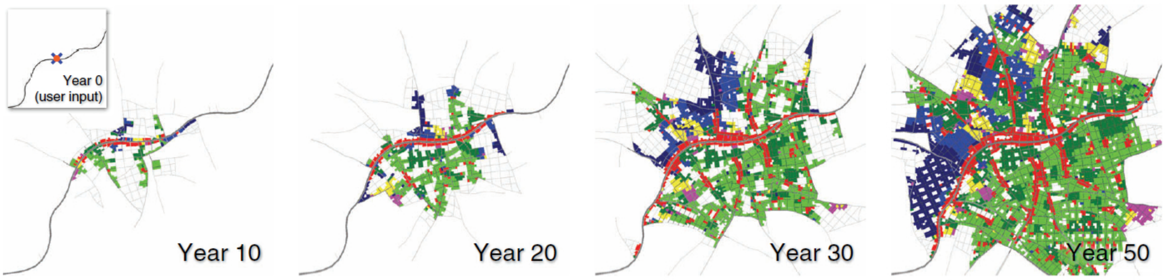


Figure 40: City evolution over time in terms of different urban areas (green: residential; blue: industrial). Source: Weber et al. (2009)

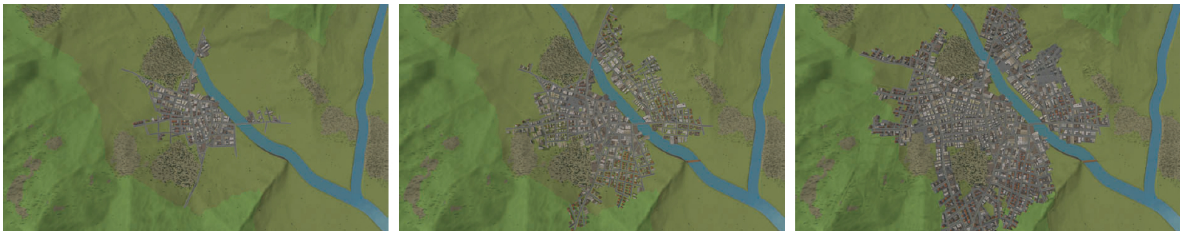
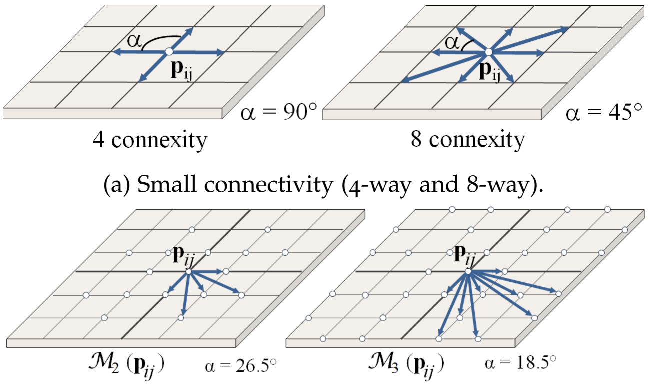


Figure 41: City example (three different times). Source: Weber et al. (2009)

possible directions is lower. Path segment masks are visually described in Figure 42. By increasing the connectivity, the shortest path algorithm converges to a solution with lower cost but with higher length, due to having more possible directions to go, thus being more resource demanding and taking more time to compute it. We can see the results in Figure 43.

To take into account the curvature of roads to make it more realistic when approaching slopes, the algorithm now operates in a three-dimensional grid. The result, in an uphill scenario, is a shorter road with less shape turns and with a lower cost as seen in Figure 44.

Tunnels and bridges are then identified across the trajectory by using tunnel/bridge path segment masks. If the a tunnel/bridge eligible point has a lower cost then a surface point,



(a) Small connectivity (4-way and 8-way).

(b) High connectivity (16-way and 32-way).

Figure 42: Path segment masks comparison. Source: Galin et al. (2010)

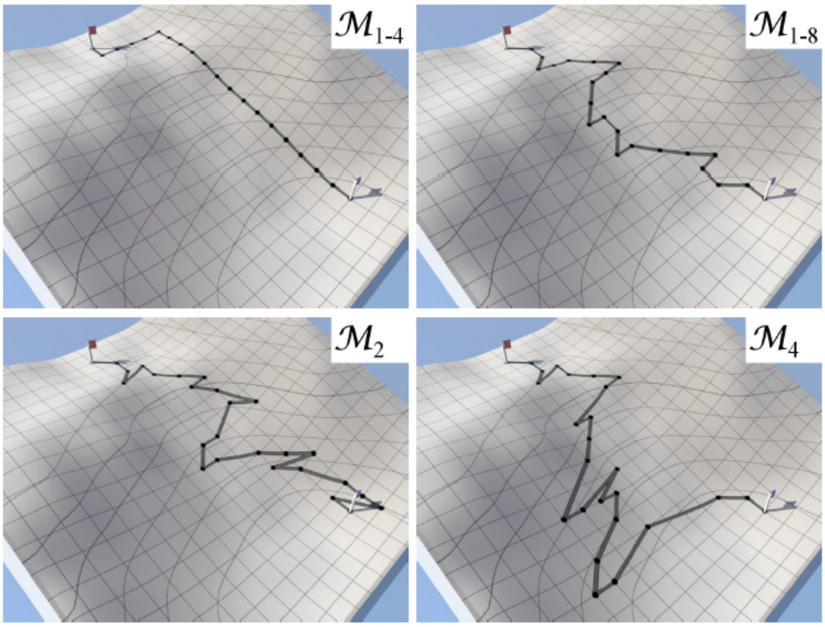
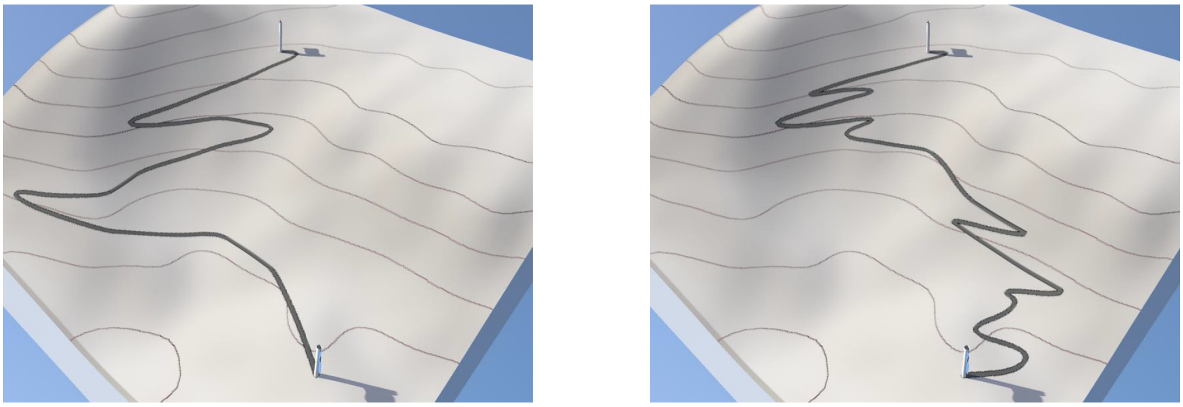


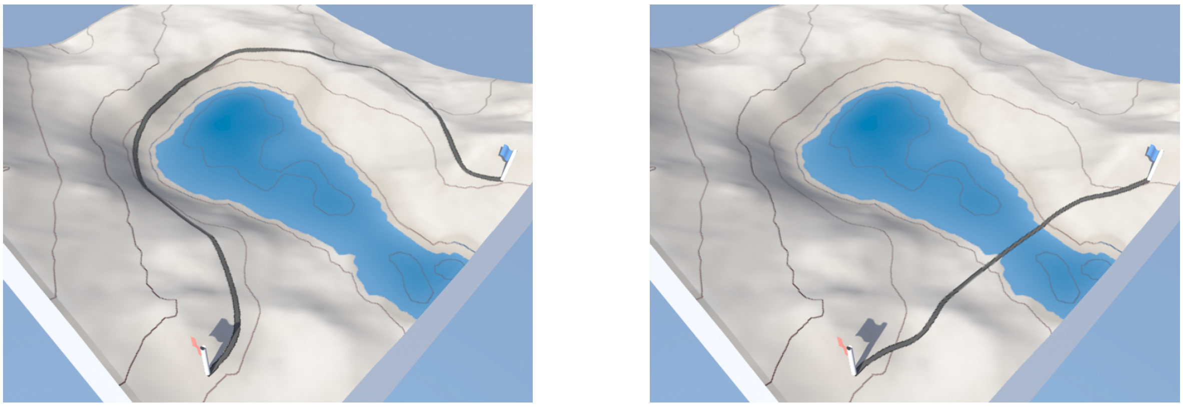
Figure 43: Comparison between different results varying path segment masks. Source: Galin et al. (2010)



(a) With curvature constrains.

(b) Without curvature constrains.

Figure 44: Curvature constrain influence in uphill road scenarios. Source: Galin et al. (2010)



(a) Without bridge.

(b) With bridge.

Figure 45: Bridge influence in road generation. Source: Galin et al. (2010)

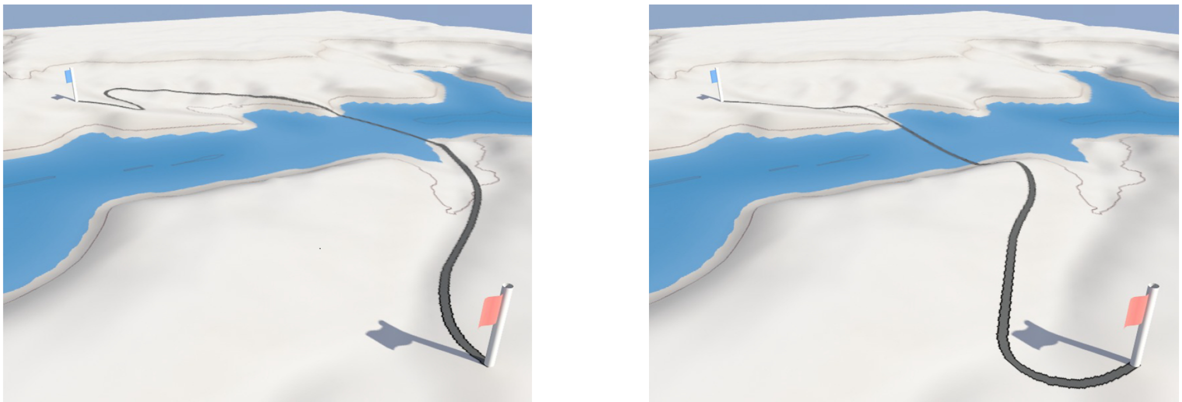


Figure 46: Different generated roads due to the use of a stochastic method. Source: Galin et al. (2010)

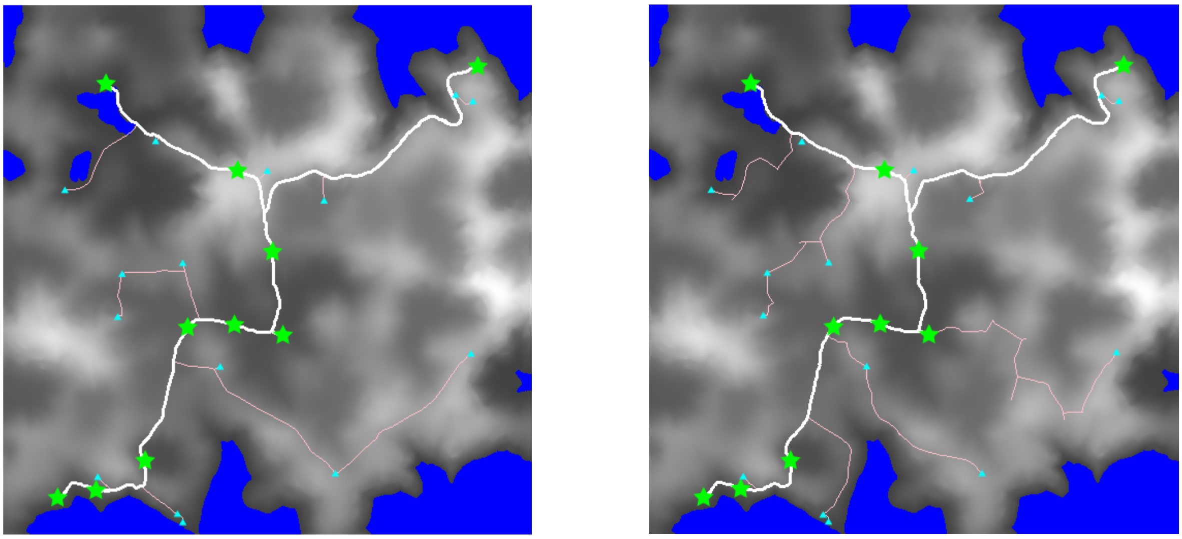
a tunnel/bridge will be constructed. By introducing these structures, the road generation takes more time to compute, giving better results with shorter roads and lower cost, as seen in Figure 45.

To lower the time needed to compute them, the points where can be constructed a tunnel/bridge are now sampled, this stochastic method is a major improvement although it results in slightly different trajectories for the same problem, where some may be more expensive than when taking into account all the points. We may see the result in Figure 46.

The found shortest path is converted to a trajectory using splines and in the end, the road, bridge and tunnel models are generated using procedural methods.

This approach also offers a set of parameters that affect the overall cost of the path by given different weights to the natural obstacles and formation of bridges and tunnels. All these features achieved a realistic procedural road generation method.

As a follow up to this work, *"Procedural Generation of Road Networks for Large Virtual Environments"* (Martek, 2012) adapts the previous work and compared its results with real-world city maps. Given a set of cities, it uses the shortest path algorithm to connect each



(a) Naive approach.

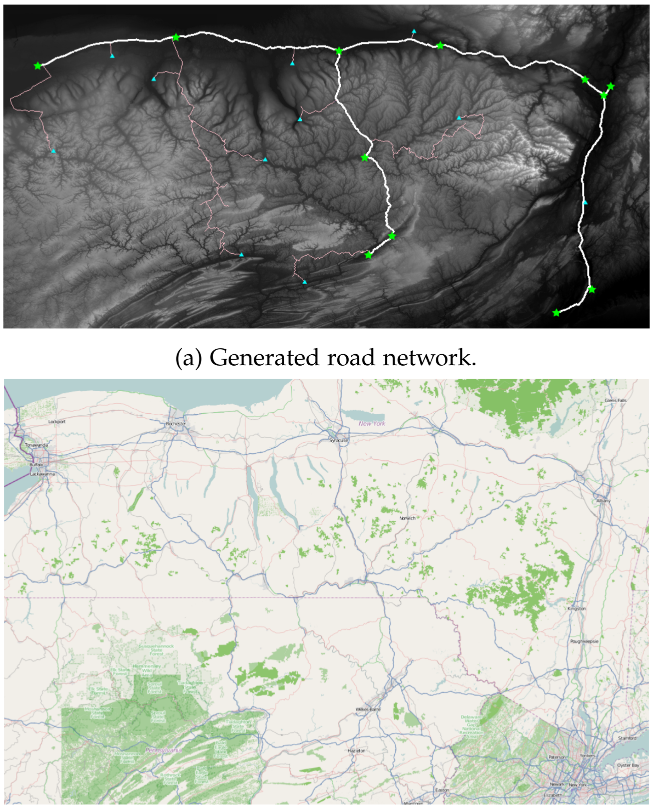
(b) RERT approach.

Figure 47: Comparison between minor roads generation methods. Source: Martek (2012)

major city until there are no disconnected cities, adding the resultant roads to a minimum spanning tree, forming the major roads. In this work is also added a new cost function related to population density. To connect the minor cities to the road network there are two approaches: the naive approach consists in also executing the shortest path algorithm between a minor city and the closest point in the major road network, the other approach, Rapidly Exploring Random Tree, consists in randomly selecting points across the grid where the shortest path has a cost equal to or lower than a given maximum cost threshold. It then connects to the closest point in the network. It also has a pruning mechanism to discard unnecessary network segments, minimising the network. This work achieved interesting results where some major roads from New York and North Carolina were closely replicated (Figure 48), and it was also noticed that the RERT approach to generate minor roads was more pleasing (Figure 47), interesting and realistic than the naive approach.

The reason to explore road network generation methods may vary from including the results in a video-game, or in a movie with Computer-Generated Imagery (CGI), but also, it can be used to simulate how the traffic behaves in that generated network.

To contribute to the latter, *"Environmental-Sensitive Generation of Street Networks for Traffic Simulations"* (Lindorfer et al., 2013) explores a road network generation method based on the renowned L-Systems and by input data such as geographical maps, that is completely user-controlled to execute traffic simulations. This approach differs from the previous ones, Parish and Müller (2001), Chen et al. (2008) and Sun et al. (2002), because it strictly separates the highway network, the interconnection of cities and minor roads. It also allows to interfere in the street generation anytime, and the system will automatically re-adjust itself when



(a) Generated road network.

(b) New York map.

Figure 48: Comparison between generated road network and New York. Source: Martek (2012)

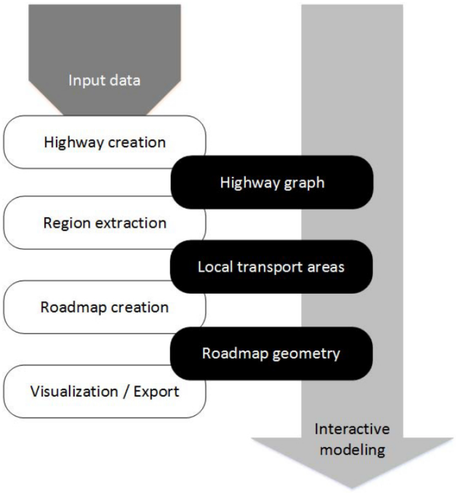


Figure 49: Network generation pipeline. Source: Lindorfer et al. (2013)

changed. This will save time by avoiding post-processing adjustments. The system pipeline is shown in Figure 49.

As mentioned before, the inputs may be geographical maps to set the boundaries of the region, statistical data to have information of the population density and parameter files to control road segments length and curvature. As seen in other works, the statistical population density maps are read to identify highly concentrated regions to be considered as cities, which are defined by a user given threshold. Those cities are therefore interconnected by highways using a minimum spanning tree algorithm (Kruskal, 1956), leaving the subdivided minor areas to be interconnected by streets. The highway graph can be seen in Figure 50. From the minor areas encircled by highways, they will be subdivided into local transport areas, as seen in Figure 51.

To connect those local transport regions, it is used a environmental-sensitive L-System. The L-Systems used in road generation are more complex than the cellular growth examples. The street modules are generic and they do not care about the parameters. The user-defined parameters are set as default and the setting and modification of parameters are defined by external functions, like roadConstrains, that deal with environmental constrains. If the parameter adjustment is successful, the road segment is introduced in the street graph, otherwise it is discarded. This accelerates the process because it allows to keep the street modules intact being the result only changed by changing the module parameters by the roadConstrains function, whilst regular L-Systems would have to rewrite the entire module to modify the existent production rules.

This process pipeline is shown in Figure 52.

Based in Alexander et al. (1977) work also used in Sun et al. (2002), the street pattern chosen in this work are raster and radial, as seen in Figure 53a and Figure 53b.

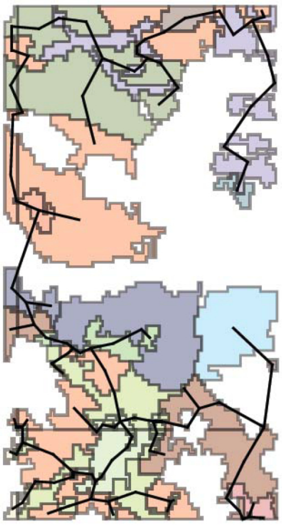


Figure 50: Highway generation. Source: Lindorfer et al. (2013)

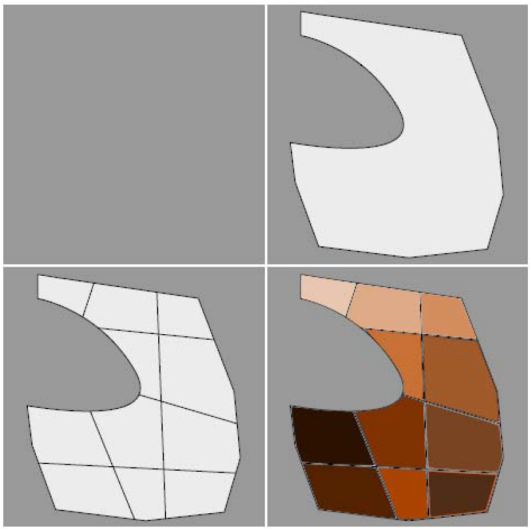


Figure 51: Region extraction. Source: Lindorfer et al. (2013)

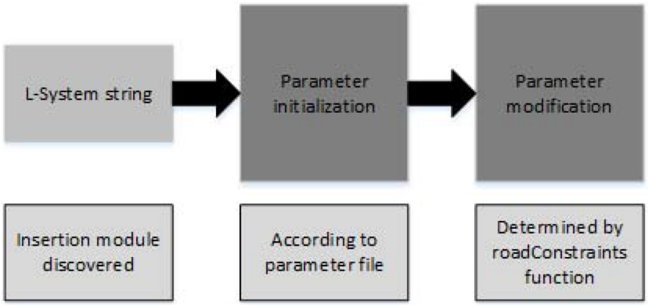
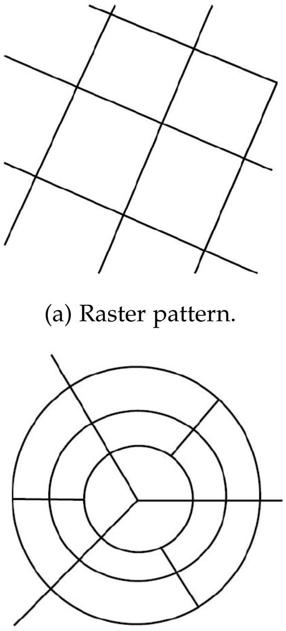

L-System string
Parameter initialization
Parameter modification
Insertion module discovered
According to parameter file
Determined by roadConstraints function

Figure 52: Parameter modification process. Source: Lindorfer et al. (2013)



(a) Raster pattern.

(b) Radial pattern.

Figure 53: Road patterns. Source: Lindorfer et al. (2013)

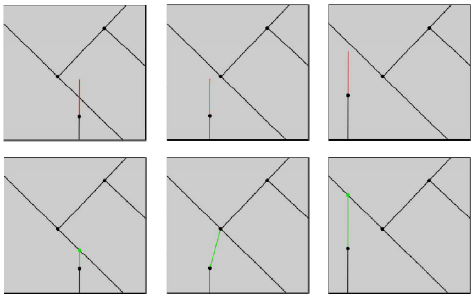


Figure 54: Parameter adjustment by roadConstrains function. Source: Lindorfer et al. (2013)

To clarify, the external function roadConstrains checks if the proposed road segment lays over an illegal area, if so, it tries to prune and/or rotate the segment, and if no valid arrangement is found, that segment is discarded. Even if the segment is not above an illegal area, it is also pruned and/or rotated to connect to an existing intersection in the network or by creating a new one, thus avoiding a dead-end road segment, as seen in Figure 54. Besides this mechanism, there is a breakpoint connection method to avoid dead-ends that consists in finding if every dead-end node can reach the network in a given radius, if so, the road segment is rotated to connect to the network.

Yang et. al. Yang et al. (2013) propose an interactive system where the user starts by specifying main roads, lakes, parks and other types of map constrains. The map is then divided in cells limited by these constrains. The division is made using two different algorithms: template-based splitting and streamline-based splitting, as seen in Figure 55. The template-based splitting aims to distort and fit existing templates into a sub-region. The template to be used in a sub-region is chosen by taking into account how common that pattern is, the *energy* required to use it and how many times it was already used in the layout. For each cell the user can then specify the thresholds for a template to be chosen to fill the cell. The energy is a metric that considers the deformation of the template to be used. The streamline-based splitting computes a *cross field* within the sub-region, then streamlines are extracted and the best one is picked to split the sub-region.

Figure 56 shows the initial division and categorisation of the sub-regions, a set of available templates, and a result computed by using the algorithm from the initial division.

In Figure 57 we may see different layouts generated for the same region based on placing different constrains. Each layout uses different templates and also streamline-based splitting.

Campos Campos (2015) based his method in the procedures used in road engineering, producing road paths definitions according to design standards, and current practices in road design, thus producing roads similar to those found in the real world. Figure 58 shows several results.

Nishida et. al. Nishida et al. (2016) approach is based on the extraction of patches from real road maps, where a patch is defined as being a meaningful road structure. An

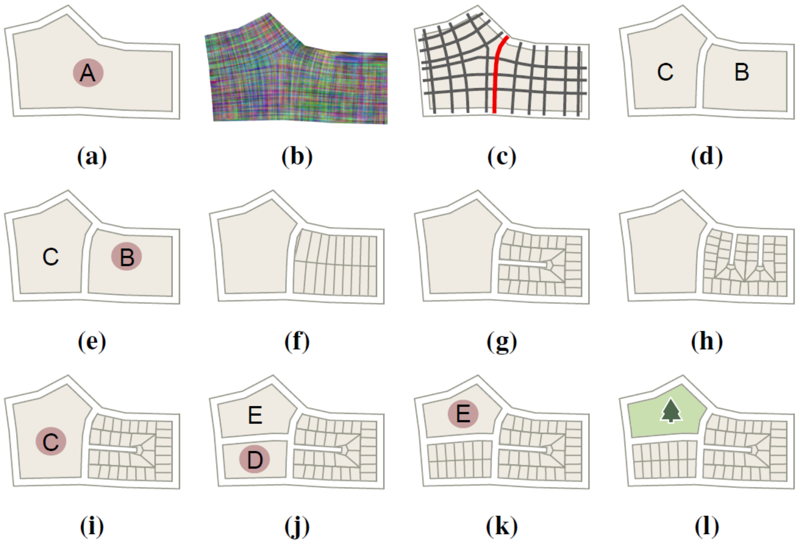


Figure 55: Illustration of the splitting operations. The top row illustrates streamline-based splitting: (a) a region, A, is selected for splitting; (b) a cross field is computed; (c) streamlines are extracted and ranked (the best one is highlighted in red); (d) the region is split along the best streamline generating two sub-regions, B and C. The middle row illustrates template-based splitting: (e) the sub-region B is selected for splitting and available templates are deformed to evaluate how well they fit. Three candidate templates are shown in (f), (g), and (h). Option (g) is selected as the best match. In the bottom row, the example is finished with streamline-based splitting (i,j) and template-based splitting (k,l). Source: Yang et al. (2013)

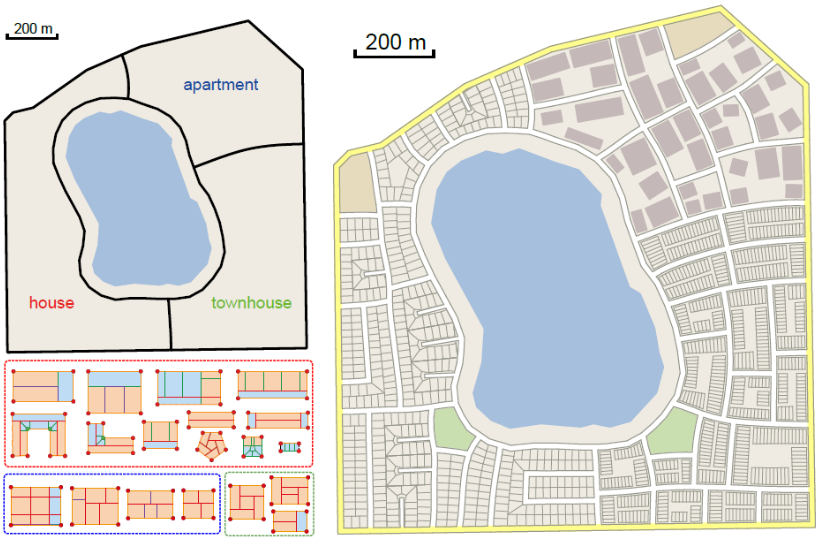


Figure 56: Top left: initial layout with different types of sub-regions; Bottom left: selected templates; Right: final layout. Source: Yang et al. (2013)

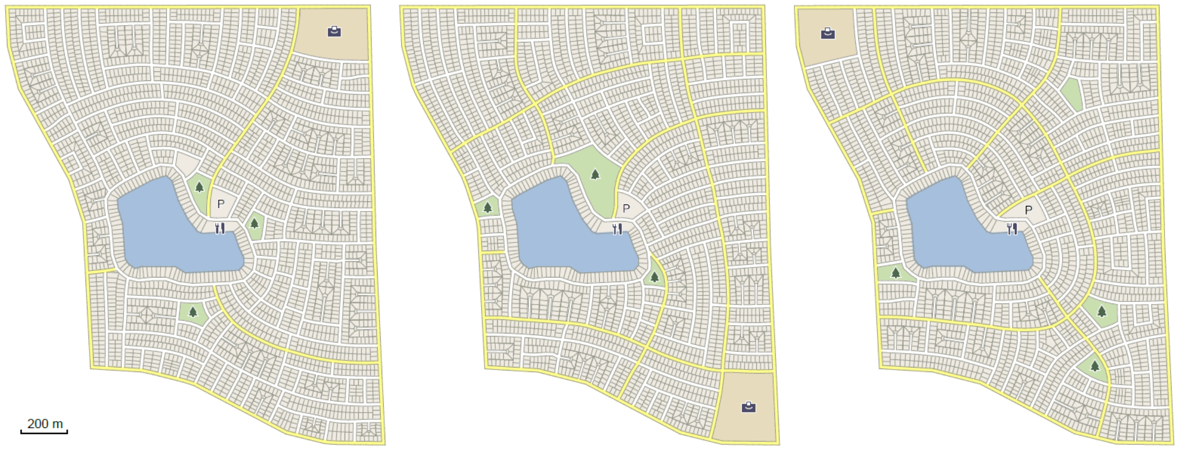


Figure 57: Three design variations for the same region starting from different user constraints (highlighted by the yellow roads). Source: Yang et al. (2013)

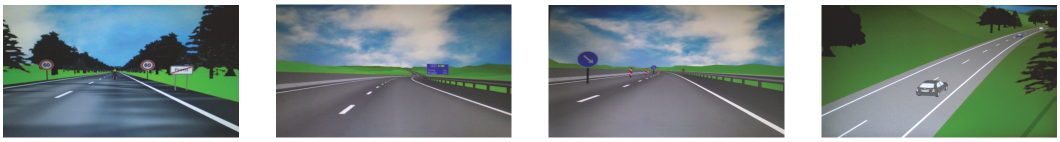


Figure 58: 3D Results showing the generation of both vertical and horizontal signalisation in both urban and rural environments. Source: Campos (2015)

interactive system was built in which the user selects areas from real-road maps from which patches are extracted and used in the new road layout. The new road layout will use the patches to generate the roads, preserving the style of the road layout in the selected patches. Furthermore, the production of new roads using examples may be combined with the use of procedural operations. This work also generates a 3D model of the city with its elements (buildings, parks, vegetation and street lamps). Figure 59 shows each step of the process.

Figure 60 shows how the patches are extracted from the network, how they are used and what is done when there is no usable patch to grow a road network.

In Figure 61 it is shown an example of redeveloping an existing city. From an existing real road map, the user was allowed to select an area and apply different generation methods, as desired.

Later, Teng and Bidarra Teng and Bidarra (2017), extended the core patch idea to include semantics, and allow a higher level editing based on these semantics. It separates the road generation in two phases: the first one to generate main roads and the second one to generate the local streets. Unlike Nishida et al. (2016), the first phase uses a parametric graph growing algorithm, inspired in Kelly and McCabe (2007), to generate the main streets. First phase roads will form main cells, later, the second phase will grow local streets inwards within these main cells using patches and parametric-based roads if needed. Figure 62 shows several patches with its semantic tag associated. Figure 63 shows the second phase cell

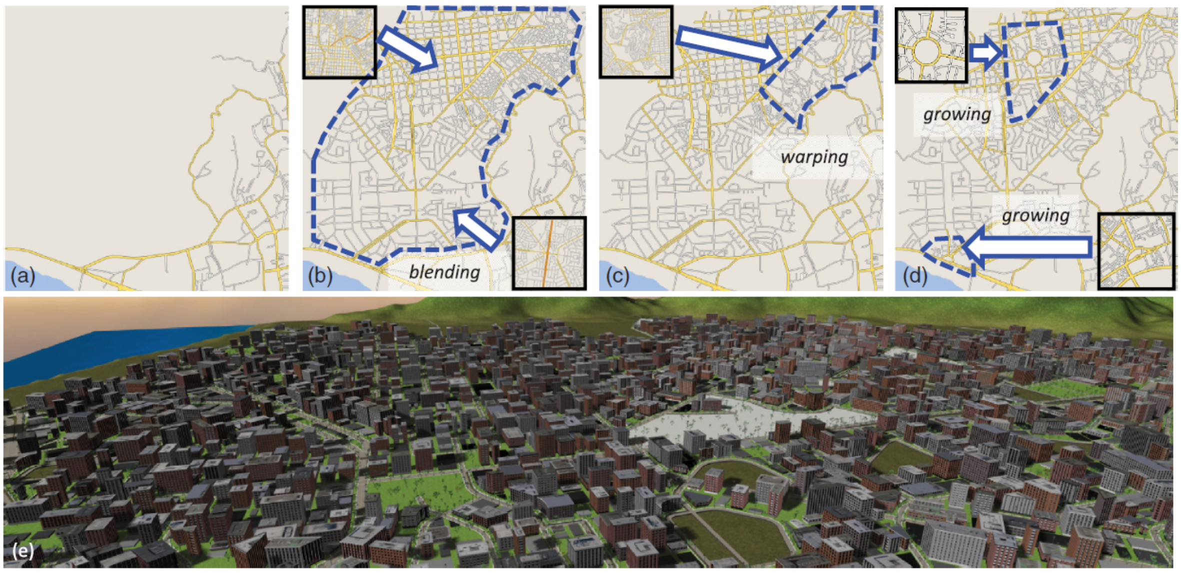


Figure 59: a) User-selected real road map; b) User-selected target area to generate roads; c) - d) User-selected areas with an alternative road generation; e) 3D city model. Source: Nishida et al. (2016)

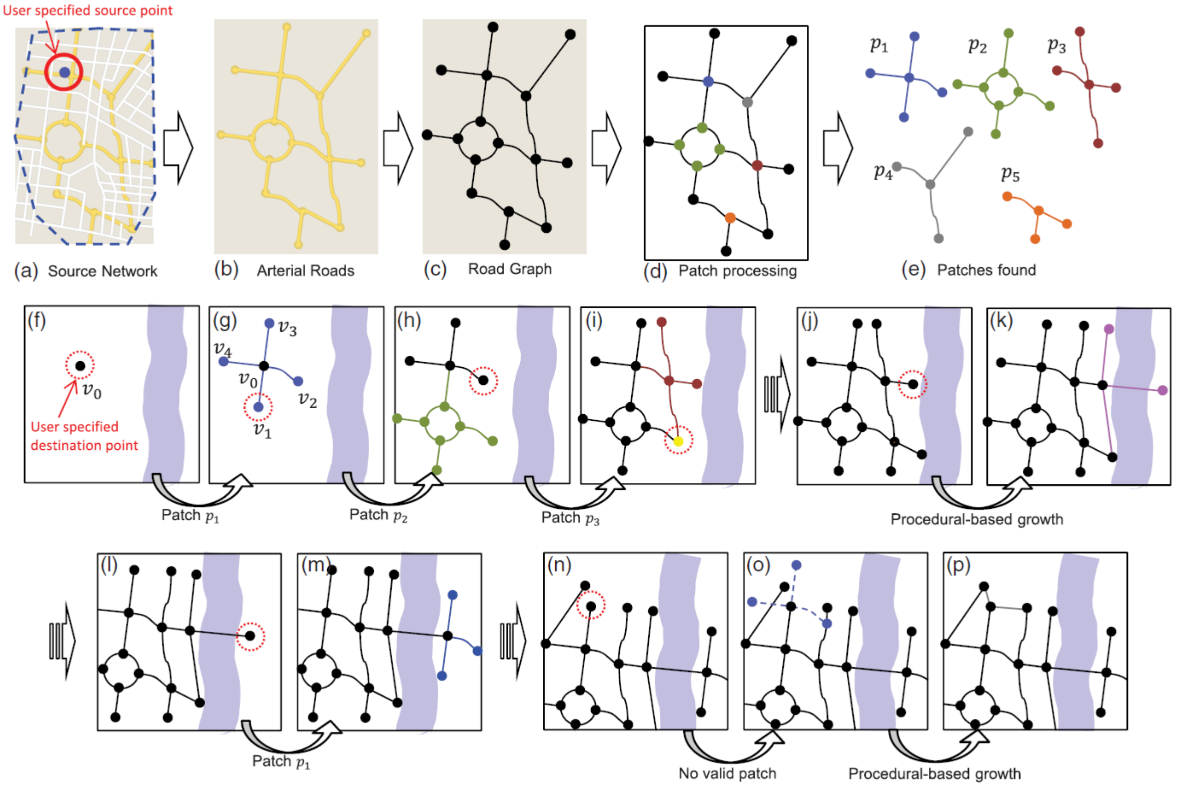


Figure 60: a) - e) Patch extraction; f) - i) Patches applied to the new road network; j) - l) Procedural-based growth due to lack of usable patches. m) - p) Repetition of the previous steps. Source: Nishida et al. (2016)

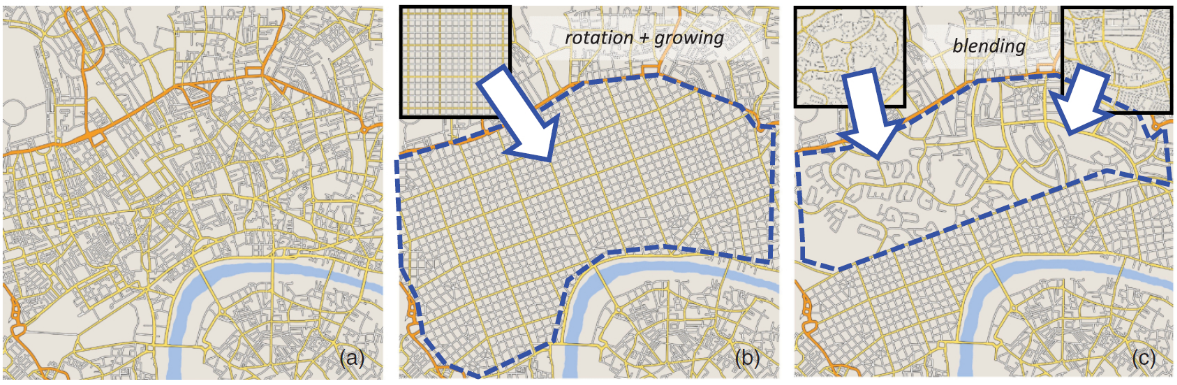


Figure 61: a) User-selected target area; b) Area replaced with a regular grid pattern; c) A more organic type of network was generated in the centre. Source: Nishida et al. (2016)

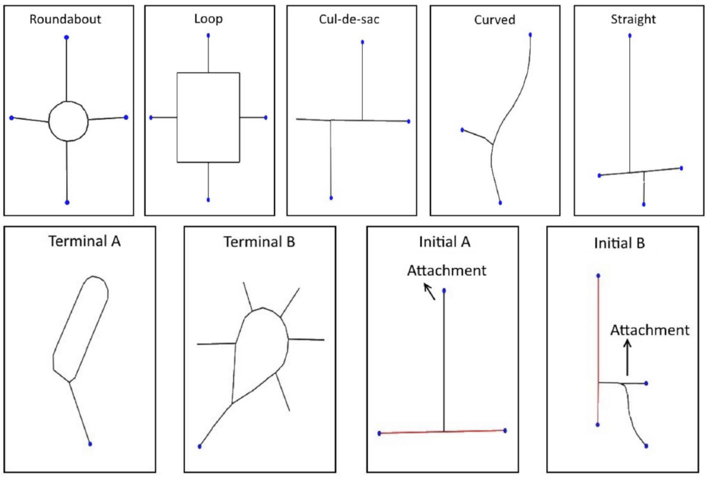


Figure 62: Set of patches with its semantic tag. Source: Teng and Bidarra (2017)

initialisation process using different patches. And in Figure 64 we have an example of a fully generated network.

2.4 SUMMARY

In conclusion, there is some diversity on the approaches taken, although a very significant number uses L-systems and/or extensions. The earliest works explore L-Systems, or variants, as the main procedural algorithm to generate both the roads and buildings in cities. Other algorithms include the shortest path algorithm, minimum spanning trees, templates, and even tensor fields. More recent works are based on pattern extraction from real world road layouts.

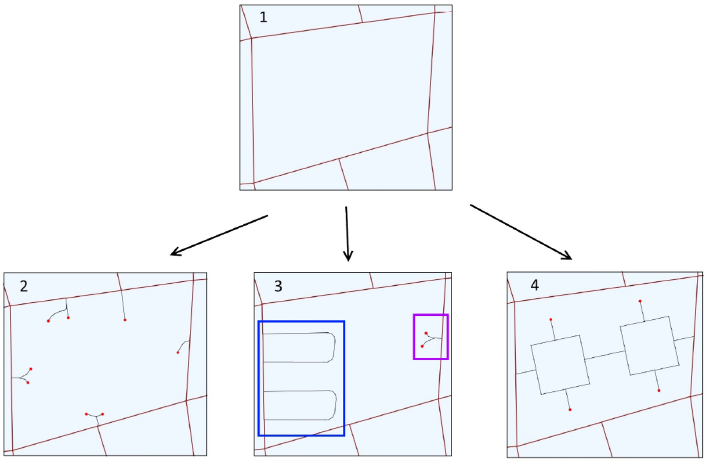


Figure 63: 1) Main cell formed by main roads; 2) Patches added to each main road intersection vertex; 3) Cul-de-sac pattern applied; 4) A more regular patch applied. Source: Teng and Bidarra (2017)

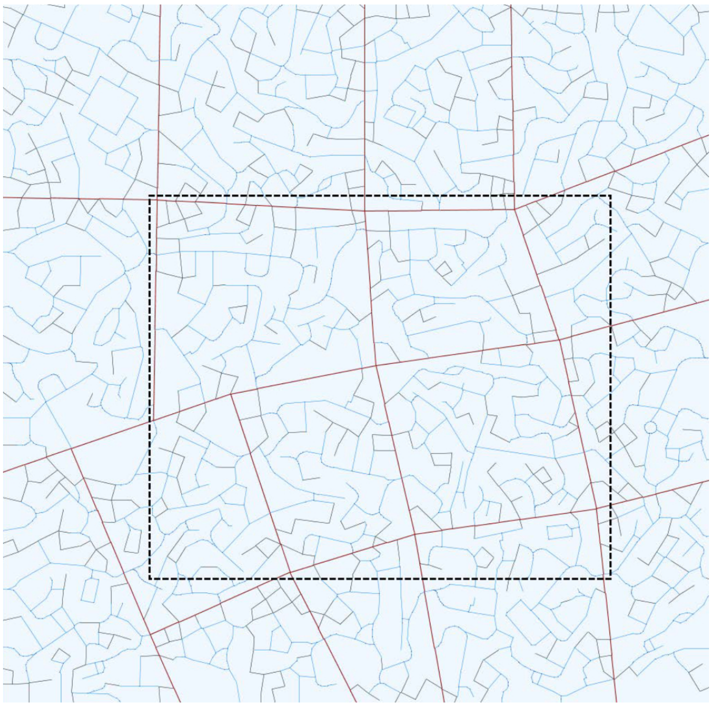


Figure 64: Purple lines: main roads; Blue lines: local streets generated by patches; Black lines: parametric-based generated roads. Source: Teng and Bidarra (2017)

3

SPACE COLONISATION FOR TREE GROWTH

SCA is an algorithm originally design for the procedural generation of leaves and trees. The essence of the SCA is the notion of attraction points, which promote branch growth and bifurcation. As mentioned before the proposal in this thesis for procedural raod generation is based on ths concept of attraction points. This chapter provides a short introduction to the algorithm in which the proposal is based as well as an explanation of the leaf venation pattern generation algorithm and why it was discarded in this work.

3.1 SPACE COLONISATION ALGORITHM

SCA is presented by Runions et. al. in Runions et al. (2007) and in summary consists in a tree generation algorithm that promotes space colonisation.

A tree is composed by *tree nodes* connected between them. The tree growth will be influenced by a set of *attraction points* that will dictate how the tree will grow and develop.

Initially, attraction points are scattered on the desired volume, together with an initial tree node or nodes. The tree will grow starting from the existing tree nodes, towards attraction points that influence them. An attraction point can only influence the closest tree node, being required that the distance between the two is less than a given distance called *radius of influence*.

Although an attraction point can only influence one tree node, a tree node can be influenced by many attraction points.

For any tree node under the influence of attraction points, a new node is created along the direction given by the average of all normalised directions between the current tree node and all attraction points that influence it. The new tree node will be placed at a distance D, the segment length, from the predecessor tree node.

All attraction points which were used to influence a new node creation that are at a distance from the node which is less than a pre-defined threshold, called the *kill distance*, are eliminated.

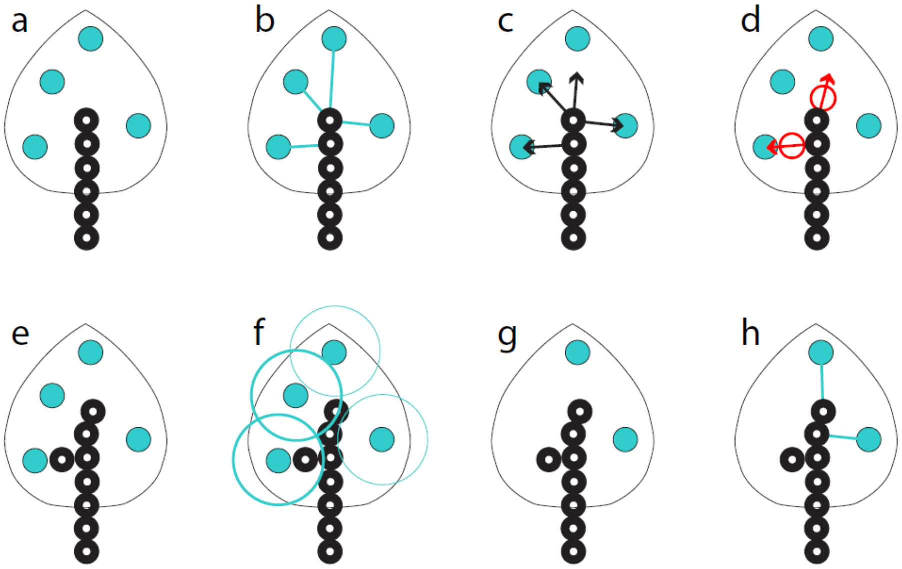


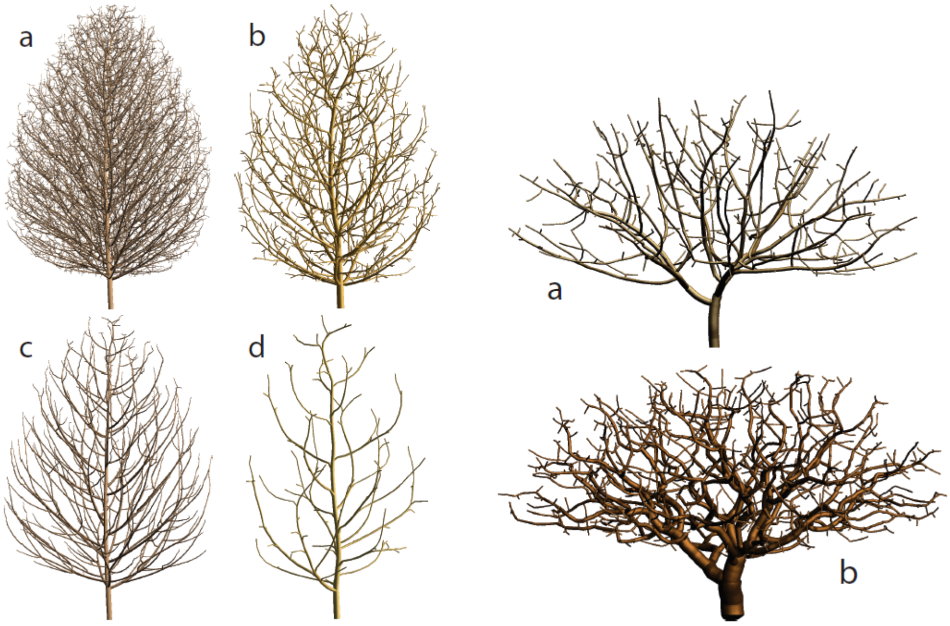
Figure 65: Space colonisation steps. Black dots - Tree nodes; Blue dots - Attraction points; Blue lines - Influence link; Black arrows - Normalised directions; Red dots - New tree nodes; Blue radius - Kill distance. Source: Runions et al. (2007)

All these steps are repeated until all attraction points are consumed, the tree is no longer attracted by any points, or the user decides to stop the algorithm. Figure 65 presents the algorithm graphically.

The results from Figure 66a show that the kill radius and the number and disposition of the attraction points have a crucial impact in the outcome. The more attraction points used, the more uniform is the growth of the structure, because with more attraction points, their individual impact decreases. The larger the kill radius, the more sparse the structure is, because the attraction points have a shorter lifespan. In Figure 66b, we may see the impact of the radius of influence. A larger radius of influence, means that each node may be influenced by more attraction points, making the growth more uniform whereas a smaller radius of influence, may produce a growth with lots of twists and turns. Later we will conclude that these parameters and how they, individually and/or in combination, impact the outcome, will be important to obtain different results and layouts, when applying SCA to road generation.

3.2 LEAF VENATION PATTERNS

Prior to the space colonisation algorithm, Runions et. al. published *"Modeling and visualisation of leaf venation patterns"* (Runions et al. (2005)). The leaf venation patterns can be distinguished by two types: open and closed patterns, as seen in Figure 67. The open venation pattern consists on a vein network tree like structure, where existing vein nodes do not connect



(a) a) $N = 12000, dk = 2D$; b) $N = 1500, dk = 2D$; c) $N = 12000, dk = 20D$; d) $N = 375, dk = 20D$.

(b) a) $di = \infty$; b) $di = 8D$

Figure 66: Examples of the impact of SCA parameters

to each other, except when a new node is generated. It will result in a more or less sparse network that it is not intra-connected. The algorithm that generates the open venation pattern, is essentially the space colonisation algorithm and is the method that gave rise to the latter work Runions et al. (2007). As for the closed venation pattern, this is a more usual pattern found in nature. A dense vein graph network may have a ladder-like pattern (*percurrent*) or a cracked/netlike pattern (*reticulate*).

Getting into further detail about the closed venation pattern, it is based on the open venation pattern but it also allows that more than one vein may grow towards the same source. This feature is to promote the formation of anastomoses, the connection of two different veins. To be similar to nature, anastomoses should be done when the two or more veins are close enough to each other, yet they are relatively far from each other. To do so, the algorithm will use the notion of relative neighbourhood. Two points, s and v, are relative neighbours of each other if for every any other point u from the set of all points that are closer to s than v, v is closer to s than to u.

In Figure 68, we may see that v, a and b are relative neighbours of s, and we can confirm it by checking their relative distances shown by lines. The green area admits the points that are both closer to s than v, and that v is closer to them than to s, in other words, that area must be empty otherwise v would be promptly discarded as neighbour of s as said

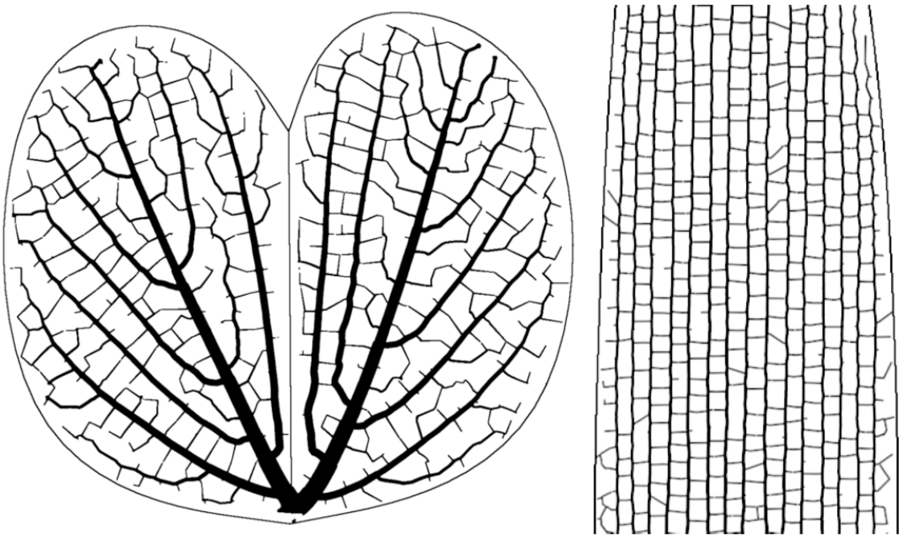


Figure 67: Left: Reticulate pattern (Orchid leaf); Right: Percurrent pattern (Grass leaf). Source: Runions et al. (2005)

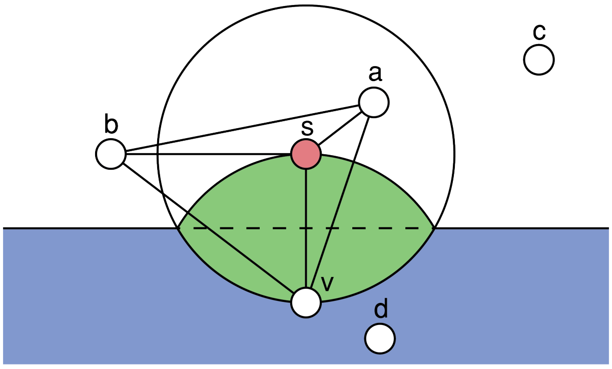


Figure 68: Relative neighbourhood example. Source: Runions et al. (2005)

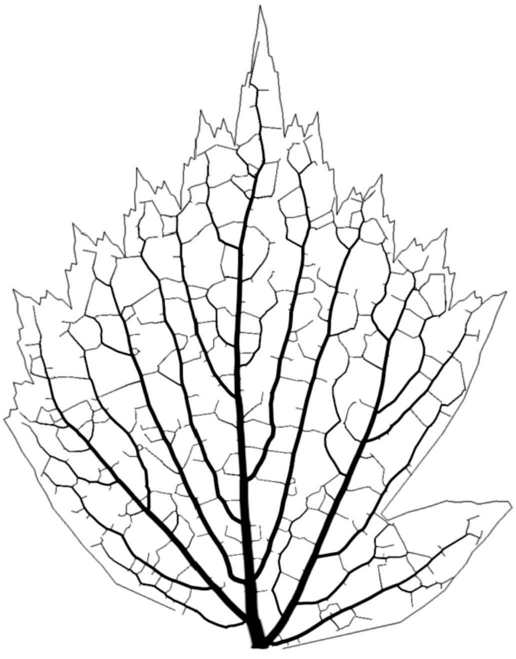


Figure 69: Left: A toothed leaf; Right: An entire margin leaf.
Source: Runions et al. (2005)

before. The blue area consists in every point that is closer to v than to s, which are promptly discarded as well.

As already seen in Figure 67, this algorithm produces extremely realistic results in terms of generating a venation network that not only tries to mimic botanical elements visually but also how do they work biologically (auxin sources, anastomoses).

As previously shown in Figure 66, the leaf venation pattern generation (Runions et al. (2007)) will also be influenced by the number of attracting sources and by the kill radius. If Figure 70 we may see the impact of these parameters in the leaf venation generation process.

This could have been the perfect algorithm to solve the problem of inter-connecting the main roads of a network, however, the relative neighbourhood method which is the main reason why the closed venation pattern is so realistic and optimal, is not flexible enough in terms of controlling its output in a road layout context.

Hence, SCA was chosen as the base for this approach, and to further parameterise and customise to suit the needs of generating a road topology.

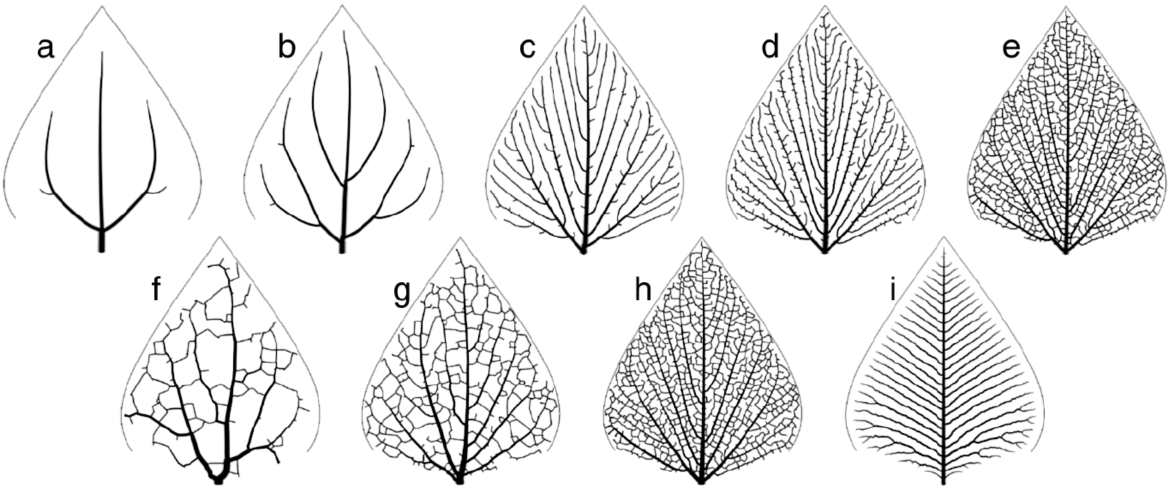


Figure 70: (a)–(e) The impact of the kill distance on venation patterns. From left to right, the kill distance decreases. (f)–(h) The impact of the number of sources inserted per step. It increases from left to right. (i) A venation pattern generated to test the slow marginal growth of the leaf. Source: Runions et al. (2005)

4

SPACE COLONISATION FOR ROAD LAYOUT GENERATION

This chapter covers the proposed extension to SCA so that it fulfils the purpose of generating plausible road networks. As in SCA Runions et al. (2007), the extension uses attraction points to shape the final network. The start of the process is to create a cloud of attraction points (in 2D). This cloud can be influenced by information maps (4.3.1). Attraction points will be used for road segment growth, much like in the original algorithm. It is also required to specify a set of initial nodes and/or segments.

Similarly to many of the previous road generation methods, this approach is a two phase algorithm. The first phase creates a 2D network of roads in what is mostly a tree like structure (4.1). The second phase produces the connections between nodes created in the first phase (4.2), effectively creating the required graph structure. Additional configuration features, namely a flow field and information maps are discussed in section 4.3.

4.1 FIRST PHASE

The first phase of the algorithm aims to generate what is mostly a tree shaped road network containing mostly main roads. When the road network cannot grow further, due to consuming all the attraction points or by not being influenced by them anymore, there are two options: to add more attraction points and continue growing the initial road network, or to proceed to the second phase which will connect the roads previously created in the first phase and add roads for neighbourhoods.

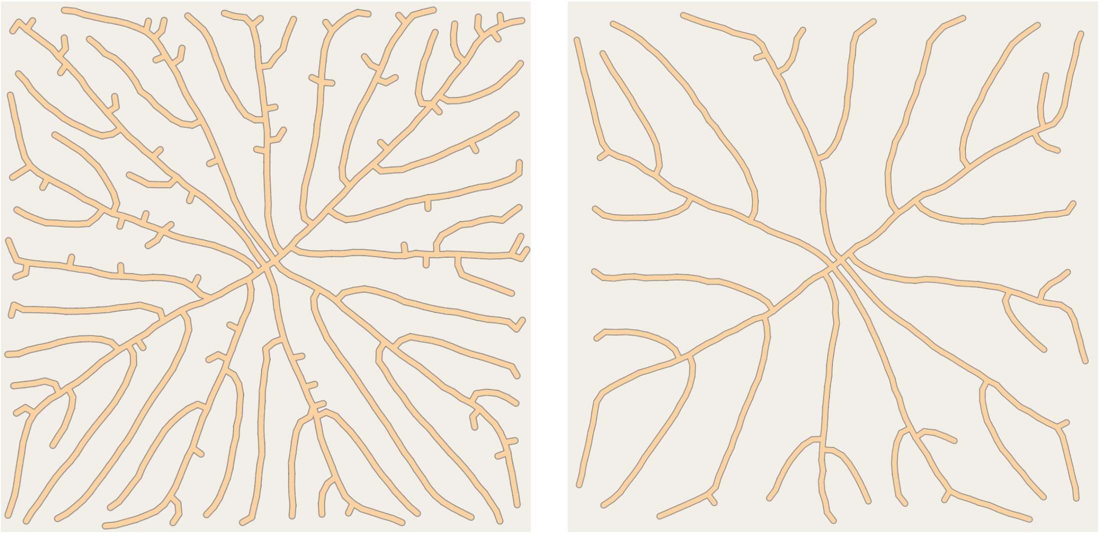
In here it will be explored some of the possible parameterisation for the first phase, starting with the original SCA's parameters, followed by the proposed extensions.

4.1.1 *Capillarity*

Different topologies require different settings. For instance, the main roads of a country or a network that represents a city map are different in terms of density, or *capillarity*. Obtaining these layouts can be achieved with SCA original parameters: *radius of influence* and *kill* *distance*. A network can have more or less capillaries, or roads in this context, depending on the ratio between the *radius of influence* and *kill distance* parameters, as stated in Chapter 3. As discussed in Runions et al. (2007) the *radius of influence* should be larger than the *kill distance* otherwise roads will not grow due to the lack of attraction points influencing the existing nodes.

The control of this feature is very intuitive, for instance, a dense network is generated by having a large ratio between the *radius of influence* and the *kill distance*, otherwise, it will produce a sparse network. In other words, with a low *kill distance* an attraction point will tend to live longer than with a higher radius, causing the growth of more roads.

Full grown road network results seen in Figure 71. These were generated using the same uniform distribution of attraction points, *radius of influence* set to 5 and the segment length set to 1, with the first node placed in the middle of the grid and with different *kill distance*. The difference between the two road layouts is due to different settings of the *kill distance* parameter.



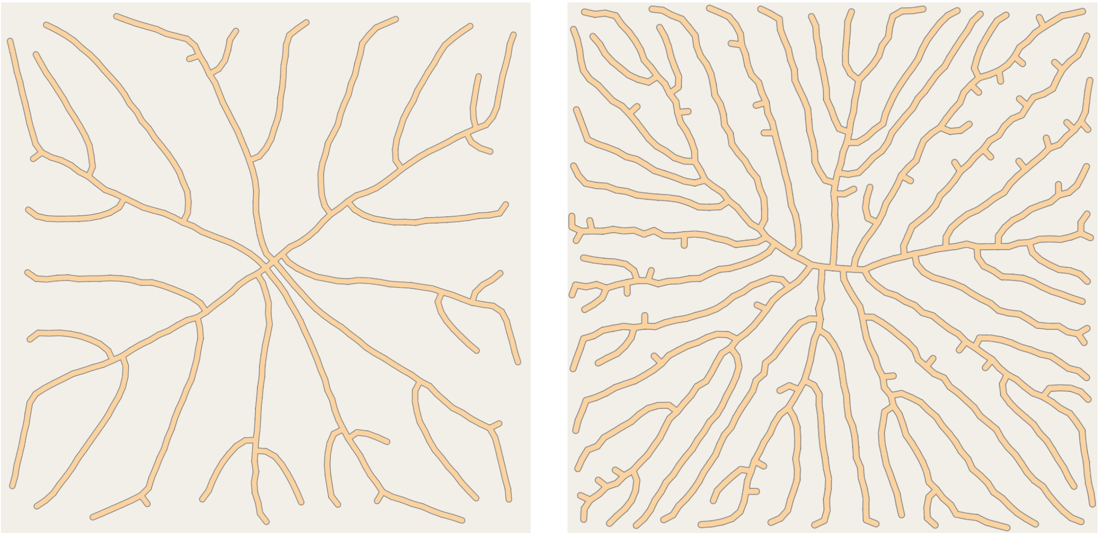
(a) Kill distance = 2

(b) Kill distance = 4

Figure 71: Capillarity influence result.

4.1.2 *Angle constraints*

To control the appearance of the initial road layout, SCA was extended to include parameters to control the angle between segments of the road. These parameters impose a maximum and minimum angle for newly created nodes and segments. Proposed directions outside the defined range are snapped to the valid range as shown in Figure 73. With this feature a rigid grid-like city map, or a more organic road layout look can be created as shown in Figure 74.



(a) Radius of influence = 5; Kill distance = 4

(b) Radius of influence = 2.5; Kill distance = 2

Figure 72: Capillarity of two maps with a kill distance and radius of influence ratio of 0.8.

Optionally, it is also allowed for the road to grow in a straight direction, even when setting angle constraints, considering a small angle range around the forward direction (0°). This angle constraint relaxation is called *tolerance range*, and it is, by default, set to half the amplitude of the given angle restriction. As seen in Figure 75, this will add a more natural look to the road network, because besides having roads within a [30°, 45°] angle range, it will also produce roads going forward or with a small deviation from the forward direction due to the use of the tolerance range, which is, in that example, [0°, 7.5°]. On the other hand, by being too restrictive, for example, when the angle is fixed to 90°, the tolerance will only consist in moving forward. This effect can be seen most pronounced in Figure 74 c).

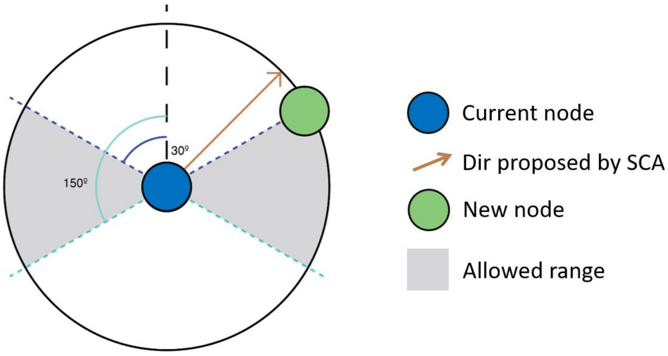
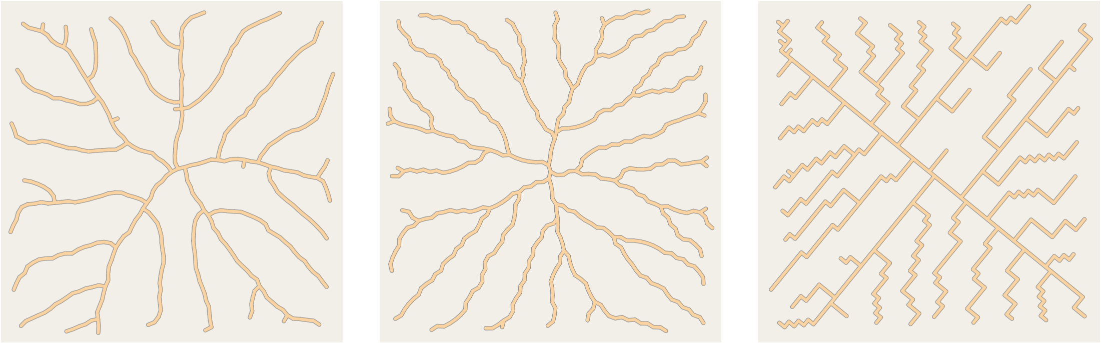


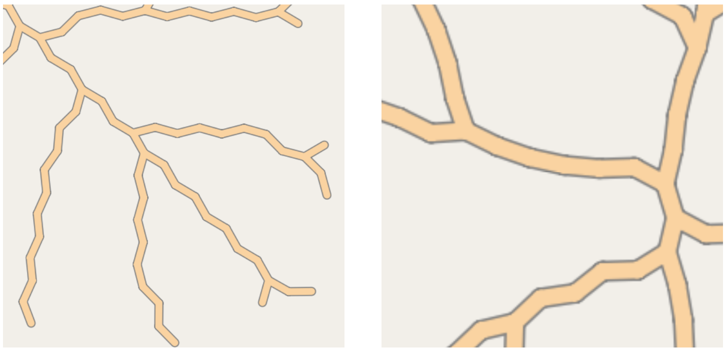
Figure 73: Angle control example.

An issue with setting angle constraints occurs when considering a node which already has bifurcations. In this case when a segment is added it may fall too close to a previously



(a) No restrictions (b) Angle range: $[30°, 45°]$ (c) Angle set to $90°$

Figure 74: Angle restriction effect on the road network.



(a) Roads following the angle restriction ($[30°, 45°]$) (b) Roads using the tolerance range ($[0°, 7.5°]$)

Figure 75: The impact of tolerance in angle restriction found in Figure 74 b).

existing segment. To avoid this it is possible to set a *minimum angle* between segments. This parameter avoids having cluttered roads.

4.1.3 *Snapping and Merging*

Mainly due to the angle limitation feature it is possible that newly created nodes are too close, or superimpose other nodes, or even crossing other segments. An example of nodes being created to previously segments and nodes is presented in Figure 84 c).

If the proposed node is too close to any other road segment or node, by a given distance, called *snapping distance*, one of two things will happen: if the nearest part of the road network to the proposed node is a previously existing node, the proposed node will be merged with this node, if the nearest part is in the midst of a road segment, it will snap to it creating a new node in that position.

Node merging takes in to account the road hierarchy. When merging two nodes belonging to roads with the same relevance, the new location is the average location of both nodes. If

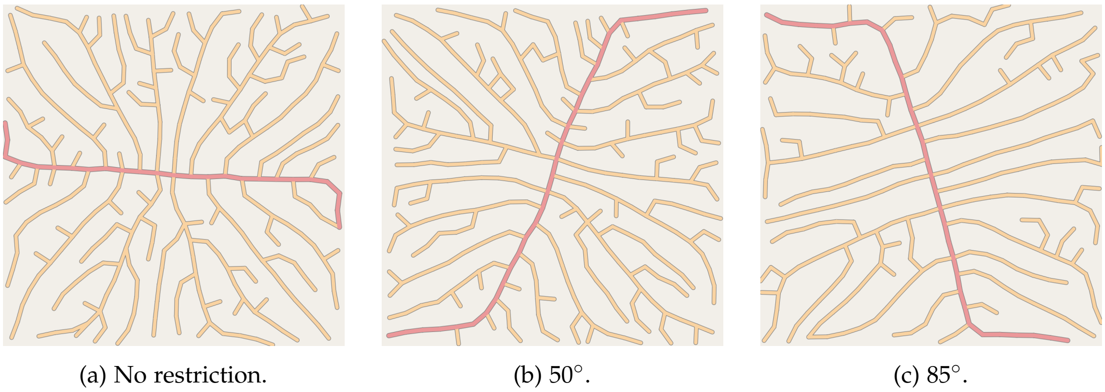

(a) No restriction.
(b) 50°.
(c) 85°.

Figure 76: Road maps generated with different minimum angle restriction values.

the new road has a different road hierarchy compared to the segment that is closest to, the node with lower hierarchy is moved to the position of the higher hierarchy node, where a node's hierarchy is the maximum hierarchy of the roads it connects

This is done to preserve the layout of the most relevant roads at the expense of modifying the topology of the least relevant ones.

Figure 77 illustrates this procedure, where road thickness relates to road relevance. This is similar to the local constraints defined in Parish and Müller (2001).

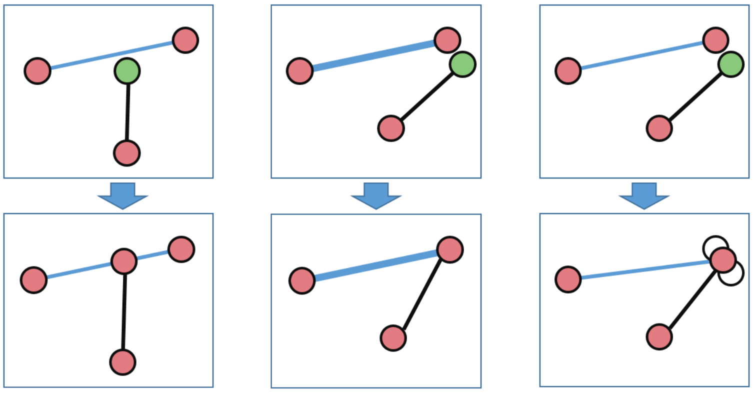


Figure 77: Snapping modes. Left: snap to segment; Middle: snap to higher relevance node; Right: snap both nodes to average position.

4.2 SECOND PHASE

In general, the first phase tends to produce a road network shaped mostly like a tree. To get an inter-connected road graph network, a second phase must be run to connect the previously created roads, and generated neighbourhood roads. In here, it will be detailed the approach to achieve this.

4.2.1 *Road inter-connection*

The first step in the second phase is to directly connect nodes created in the first phase. These connections will take into account two different aspects: the nodes to be connected should be close enough to each other, and the suggested connection should make an angle with the previous node connections that should be as close as possible to the *ideal angle*. The *ideal angle*, is the angle that allows the new segment to be the most distant to any of the other existing segments of the node, to promote a distributed network.

Figure 78 presents two examples on how the connecting mechanism works. Nodes A and B are both inside the valid area, in which case the connection will be from the current node to B because it is closer to the ideal angles.

The algorithm executes as follows. For each node, it will take the closest nodes that are not connected to it, which is measured by a distance threshold set as default to the segment length. Then, based on the current node the ideal angles of those that are close to it are measured. A node can have multiple ideal angles based on different scenarios depending on the cardinality of the node connections. When there is none, the ideal angles are perpendicular or parallel to the previous connection with the node. When there is at least one descendent from the node, the ideal angles are the average between the subsequent directions of the connections from and to this node. When we have calculated the angle between each close node and its ideal angle, we measure their *force*, based on their distance and the measured angle, and we pick the strongest to connect to the current one. May a node n, a close node c with an angle α between it and the ideal angle, the force of c is measured as detailed in Equation 3. This calculation favours nodes that are close to the ideal angle and close to the other node.

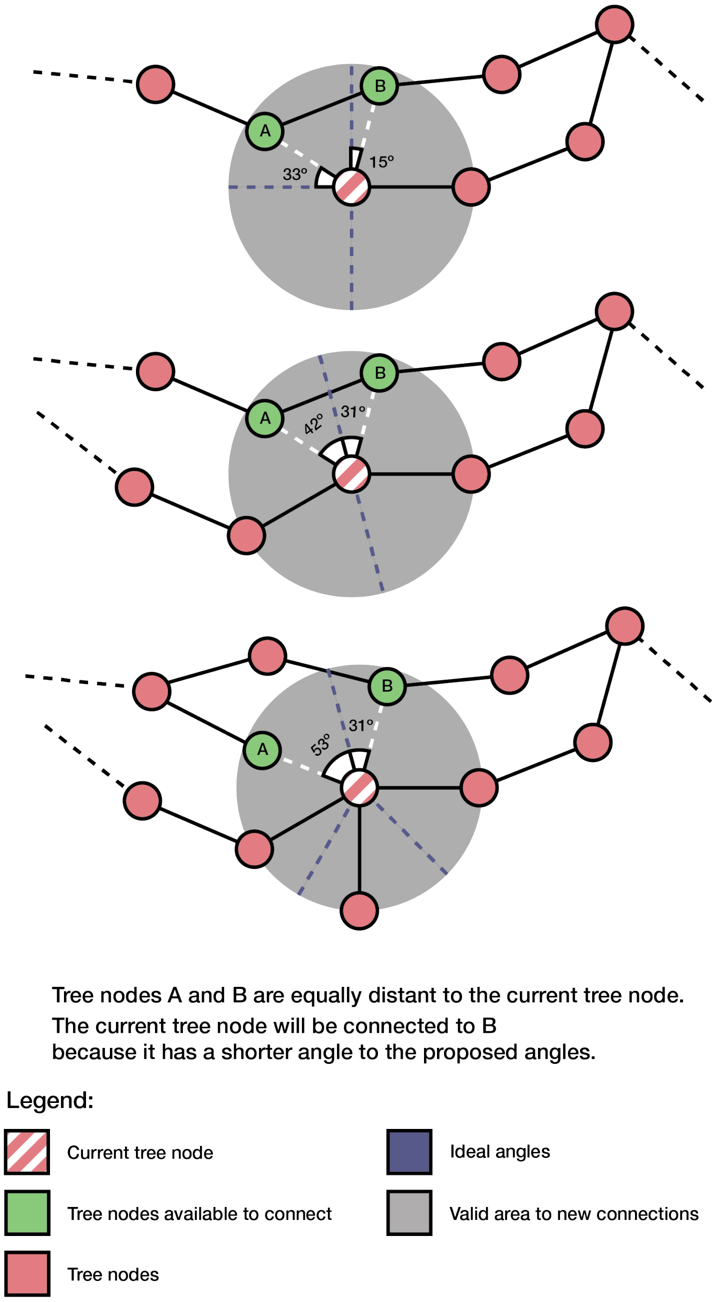
In this approach, both the angle force and the distance force are evenly taken into account.

$$\begin{aligned} angle_f &= \frac{180^\circ - \alpha}{180^\circ} \\ distance_f &= \frac{threshold_D - (|\vec{NC}|)}{threshold_D} \\ force &= angle_f \times distance_f \end{aligned} \tag{3}$$

As Figure 79 shows, when applying this procedure as described, there is a tendency to form consecutive triangular connections in some scenarios.

This is caused, for instance, by having two different roads, close enough to each other, growing from a common node, as seen in Figure 80.

Those connections may not be desirable and can be discarded without negatively affect the road network in terms of layout. Therefore, a restriction was added to prevent the formation of triangular connections in this step.



Tree nodes A and B are equally distant to the current tree node.
The current tree node will be connected to B
because it has a shorter angle to the proposed angles.

Legend:
Current tree node
Ideal angles
Tree nodes available to connect
Valid area to new connections
Tree nodes

Figure 78: Node connection mechanism.

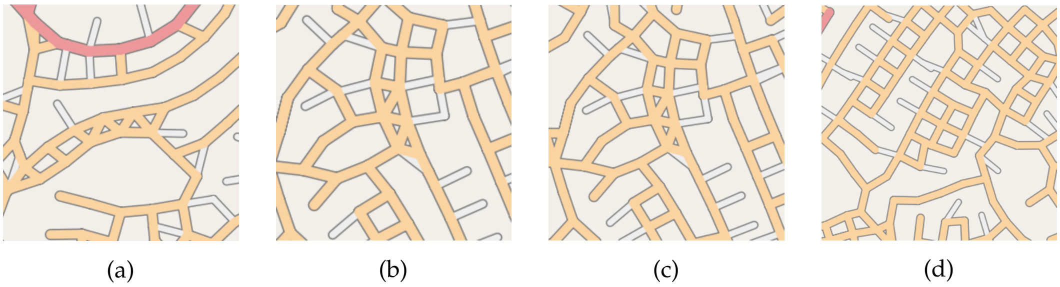


Figure 79: Images obtained from a single road map: a-c) Triangular connections; d) Perpendicular connections.

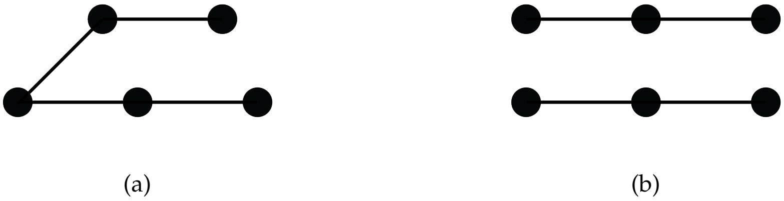
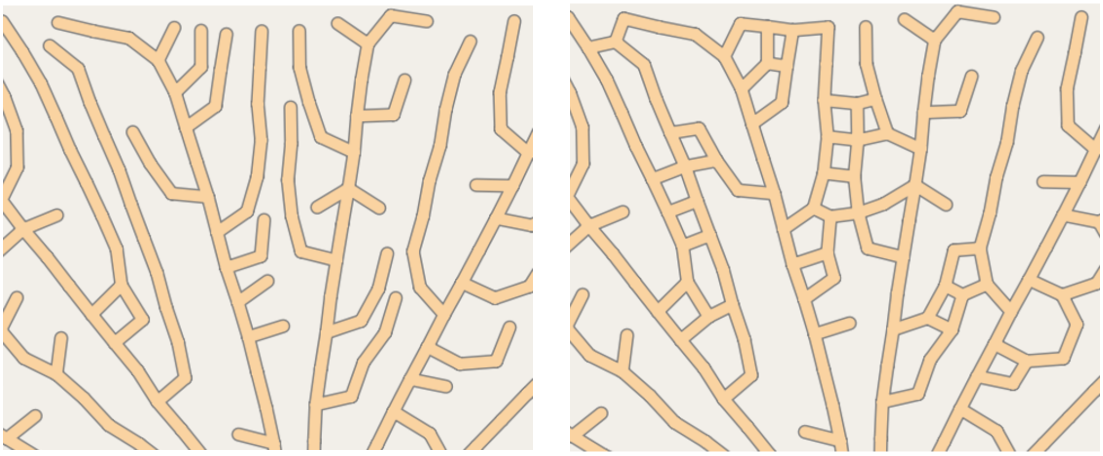


Figure 80: a): layout promoting triangle connections; b): layout promoting perpendicular connections.

An example of the application of this step is presented in Figure 81, showing both the network at the end of the first phase and the end result of this step.



(a) Before

(b) After

Figure 81: Road inter-connection impact on the network.

4.2.2 *Attraction point validation*

In the general case, to continue to grow the road network in the second phase it is required to add more attraction points. Since there is already a formed road network, the attraction point distribution cannot be completely random, otherwise new attraction points may cause the network to grow roads that can cross the original network, or create road segments too close to existing ones. Of course this can somehow be avoided using with the previous explained angle limitations, but it should be also possible to prevent those unwanted results a priori just by carefully validating the new attraction points.

To prevent this it a threshold parameter was added to control the minimum required distance from the attraction point to every existing road segment to accept it. By default this is set to half the segment's length. The distance between the attraction point and the existing segments should be higher than the given threshold. Attraction points inside this safety margin will be removed, see Figure 82.

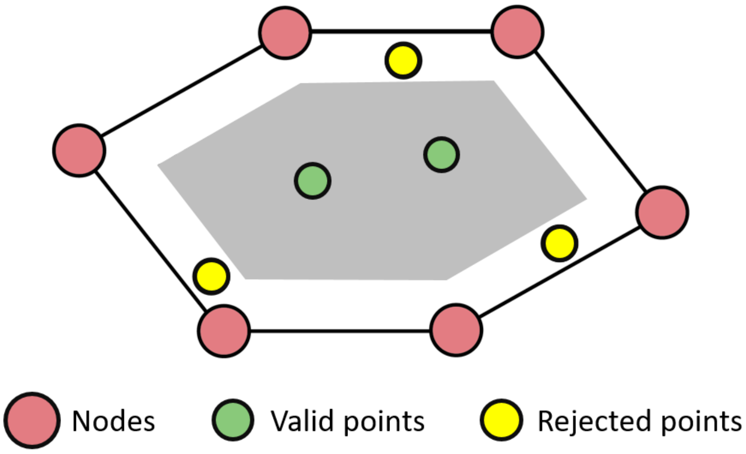


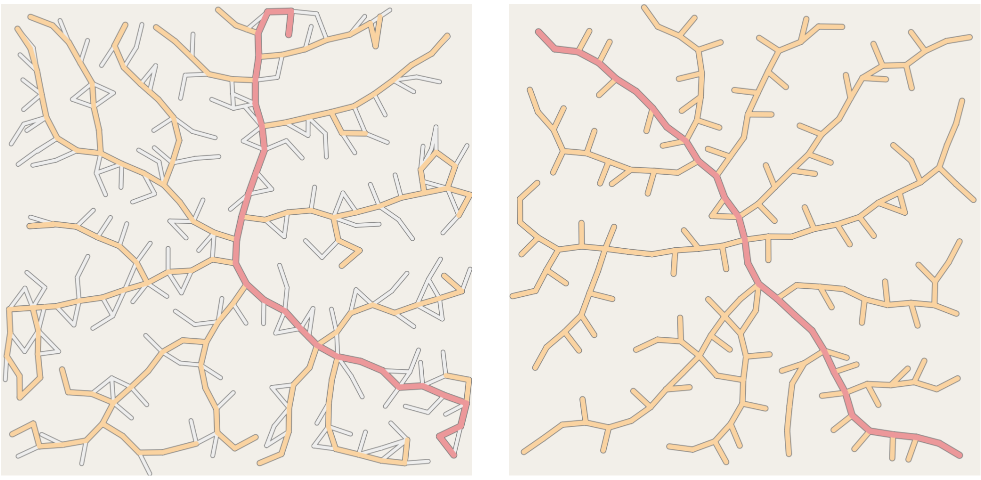
Figure 82: Attraction point validation example.

In Figure 83, it is shown the impact of the validation of attraction points in the generation of the road network. In both road networks were used the same parameters. Figure 84 shows the unwanted road segments by not using the validation method.

To visually present how many attraction points are removed using this method, another road map was generated as seen in Figure 85. For a segment length of 1, the minimum distance of an attraction point to be valid was set to 0.3 from every segment. The number of attraction point scattered was 2500, but when using the validation method, only 1432 were considered valid.

4.2.3 *Redefining attractors in the second phase*

To achieve the desired inter-connection between the first phase nodes, these nodes will also be considered as attractors. Furthermore, attraction points will attract the N closer nodes instead of just the closest node, unlike the original SCA (it was found that $N = 2$ provides more pleasing results).



(a) (b)

Figure 83: a) No validation; b) Validation used, safety distance as half of segment length.

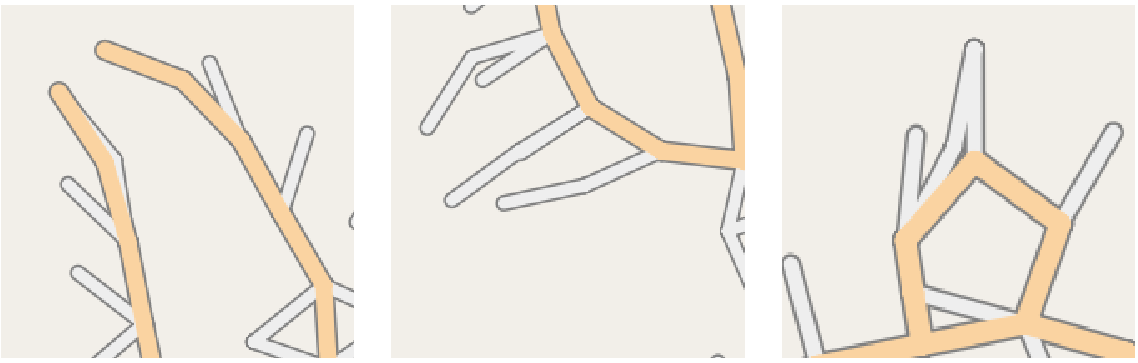


Figure 84: Examples of too close road segments

To guarantee that the road network will successfully connect itself, the *lifespan* of the attraction points relative to the first phase must be extended. This can be achieved setting a lower kill radius in this phase. By default it is set at half the value from the first phase.

Every attraction point, in the second phase, will measure its distance to every node. If the distance is under the radius of influence, that node will be added to a set of possible nodes to attract. Given the N nodes to attract, the N closest nodes to the attraction point will be attracted by it.

In Figure 86 we may see the importance of the value of N. When the scene is densely populated with attraction points, the resulting direction will be more uniform and it will probably follow a forward direction. But with few attraction points, it is relevant how many nodes they will influence. Influencing too many nodes, may cause some nodes to be influenced by attraction points that may create road layouts that are not appealing. For instance, when having close to parallel roads, and when scattering a few points, some may follow an oblique direction, towards connecting the main parallel roads. To test this, it was

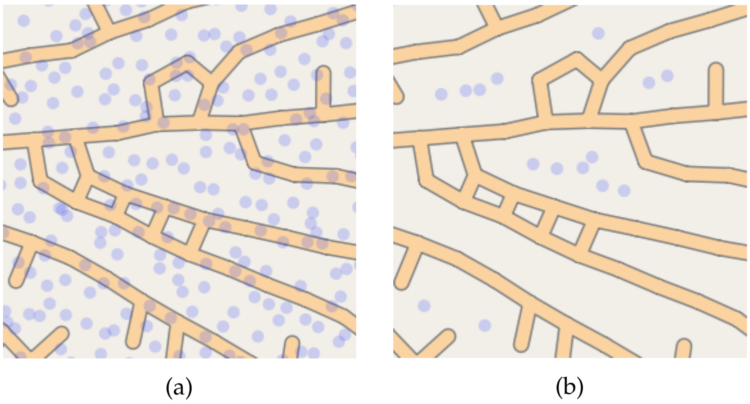


Figure 85: Blue dots - Attraction points. a) No validation; b) Validation used, safety distance as a third of segment length.

used a scenario where the attraction points had a large radius of influence that influences both roads, the minimum angle was kept to 20° to still allow a large range of directions, the kill distance in the second phase was small to extend their life span, the attraction points scattered were just a few.

In Figure 86 a) and b) we may confirm that when only influencing a node per attraction point, as in the first phase, the generated roads between the secondary ones, were just four even though much more were within the radius of influence. In c) and d), influencing two nodes, both secondary roads generated new roads which allowed to fully connect them and their directions were closer to perpendicular. We could use both in different environments, by influencing one node, because it generated a sparser network in the second phase, we may use it in rural areas, were we may find dead-ends that lead to a couple houses, in a urban area, we may want the primary/secondary roads to be connected by some roads at least. In e) and f), because there are too many nodes being influenced, the new directions are worse in terms of fluidity of the network, for instance, two roads connecting to the same node coming from two connected nodes. This can be avoided by deleting these triangle formations but it still does not aid to generate a fluid network.

4.2.4 *Extra-pruning*

When the road network is finally fully generated, in some situations it can be spotted that pairs of roads were created forming a *triangle*, as seen in Figure 87. Unlike the ones described in Subsection 4.2.1, these arise from the snapping and merging occurred in both the first and second phases of the algorithm. Those triangular patterns may not be wanted in the final road network, so, a method to eliminate them is provided. The unwanted triangle patterns found in the network are described as follows: if three nodes *a*, *b* and *c* are all connected to each other, and *a* and *b* have three or more connections each and *c* has only two connections,

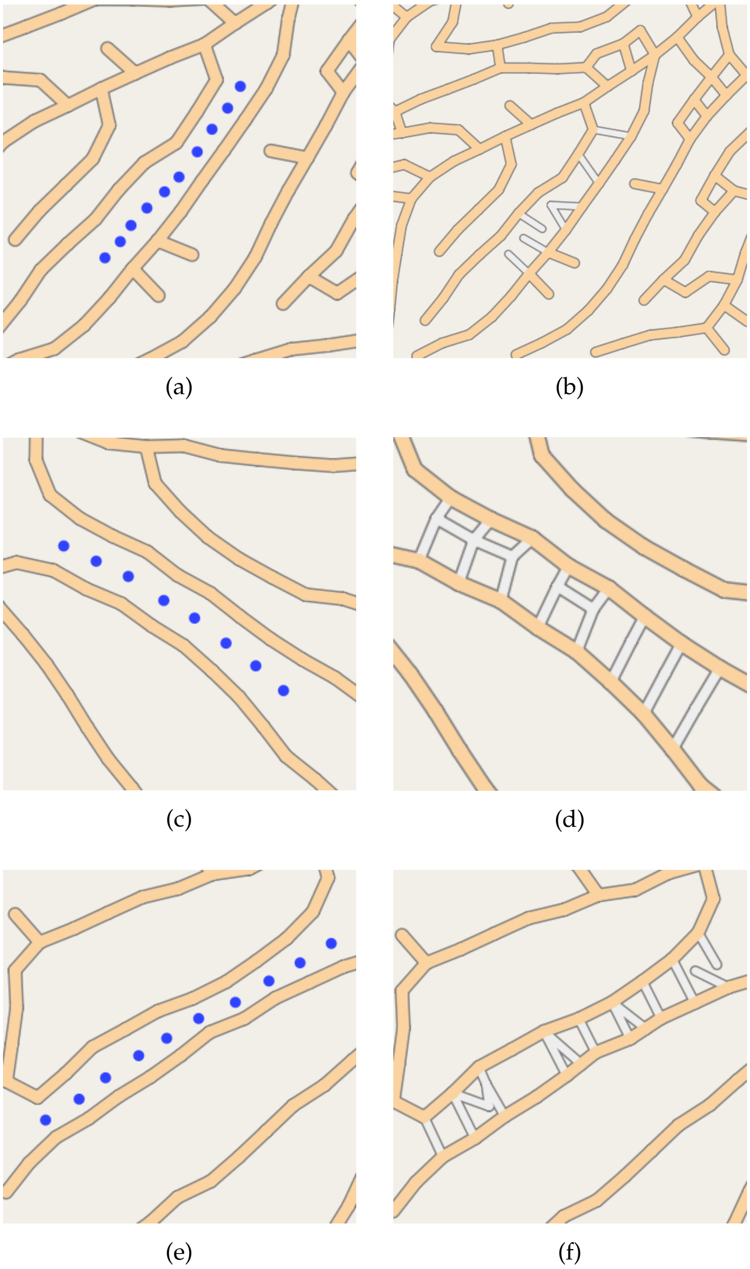
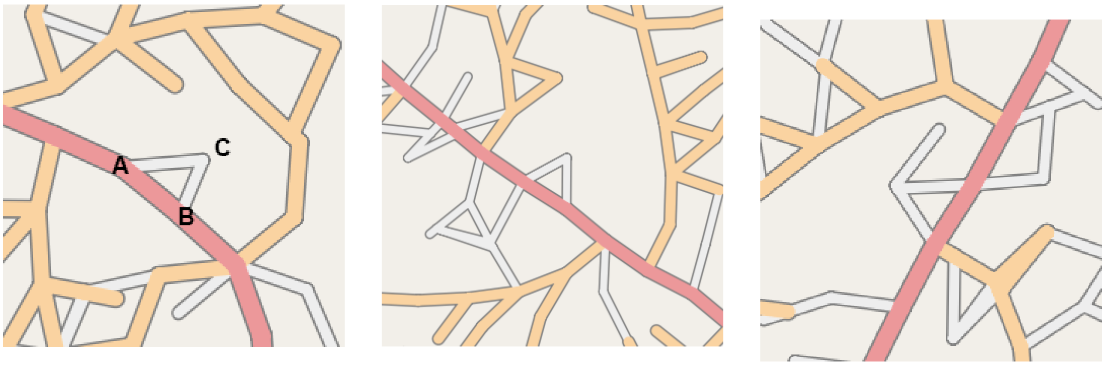


Figure 86: Number of nodes influenced per attraction point (blue dots): a) - b): 1; c) - d): 2; e) - f) 4;

this will be considered a triangle, and one of the connections of c has to be deleted. In Figure 87 c), the triangle in the centre has at least three connections in each node, which is considered valid because each node has at least a road that does not contribute to the triangle. Triangles forming at a road ending, i.e., there are two nodes with two connections only, are considered valid.



(a) Example of a triangle dead-end.

(b) Another example of a triangle dead-end.

(c) Acceptable triangle layout

Figure 87: Examples of triangle layouts in the road network

4.3 ADDITIONAL FEATURES

In this section, additional features that improve the base algorithm will be presented, such as the information maps and flow fields. These influence the placing of attraction points, in the case of population density maps or the border maps, or restrict road directions, considering height maps and flow fields.

4.3.1 *Information maps*

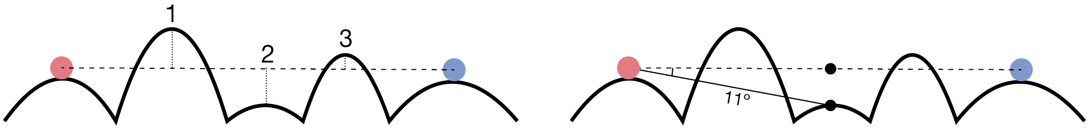
Three distinct types of information maps are considered: population density , border and height maps.

A population density map is a grey level map where lighter areas imply higher population density. These maps will condition the placement of attraction points. When attraction points are placed they are accepted/rejected based on a probability taken from the intensity at the corresponding location in the density map.

Border maps has information about illegal areas to grow roads, such as rivers, lakes, sea, city and natural parks. This will dictate which areas are available to support the road network. It can be extended to differentiate areas so that the algorithm can behave differently when generating the road network according to those areas. Attraction points placed in illegal areas will be removed, and road segments have to be checked to ensure they do not cross these illegal areas.

Height maps contains the height of each point in the grid/terrain. Darker tones are associated with low elevations, whereas lighter tones correspond to higher altitudes. These maps are used to determine the steepness of the terrain in order to decide if a road can grow in a particular direction based on a parameter The unwanted triangle patterns found in the network are described as follows:.

May $\vec{v}$ be the direction from the node to an influencing attraction point, the angle between $\vec{v}$ and the plane, $\vec{p}$, has to be lower than the *maximum steepness* angle. However, due to a large radius of influence or due to low definition elevation maps, some attraction points that may seem valid will cause the new roads to go through terrain, like tunnels, or to go over it, like bridges. To avoid that, we verify a given number of points between the node and the attraction point, and if all of them respect the maximum steepness threshold. The number of steps used will influence the generation of roads when using height maps, as well as the generation performance. Figure 88 describes this process.



(a) 1 - 3) Higher terrain; 2) Lower terrain;

(b) One point verification.

Figure 88: For a maximum steepness of 10°: a) without extra verification, the attraction point is valid; b) a verification along the direction rejects the attraction point.

Another usage for height maps is to decide if an attraction point has influence over a particular node. When the steepness of a segment connecting the two is larger than *maximum steepness*, the attraction point will not influence the node.

An example of a complete set of information maps is seen in Section 5.6.

4.3.2 *Flow field*

Many scientific areas use vector fields to represent some models and phenomena such as force, wind movement, gravitational force, *et cetera*. In this context, they are called *flow fields* since they influence the direction of new segments.

Flow fields are built using flow primitives, based on the work by Wejchert and Haumann Wejchert and Haumann (1991). The primitives will shape the field, influencing the direction of the vectors in the grid, corresponding to the building blocks of the flow field: *uniform* which causes a flow in straight lines, parallel to the direction specified by the primitive;

source, which creates a flow away from the location of the primitive; *sink* which flows towards the primitive; and *vortex* which causes a circular flow around the primitive. Each primitive is described by a linear equation, and the cumulative result at each grid point can be computed as the sum of the influences for all primitives. Under this approach, a flow field is defined by placing these primitives.

According to Wejchert and Haumann (1991), using cylindrical coordinates, the potential and the velocity field, for a source at the origin, with strength a, is described in Equation 4. A sink is described by the same equation but the strength a is a negative constant. A vortex is described by Equation 5.

$$\phi = \frac{a}{2\pi}\ln r; \quad v_r = \frac{a}{2\pi r}; \quad v_\theta = 0; \quad v_z = 0; \tag{4}$$

$$\phi = \frac{b}{2\pi}\theta; \quad v_r = 0; \quad v_\theta = \frac{b}{2\pi r}; \quad v_z = 0; \tag{5}$$

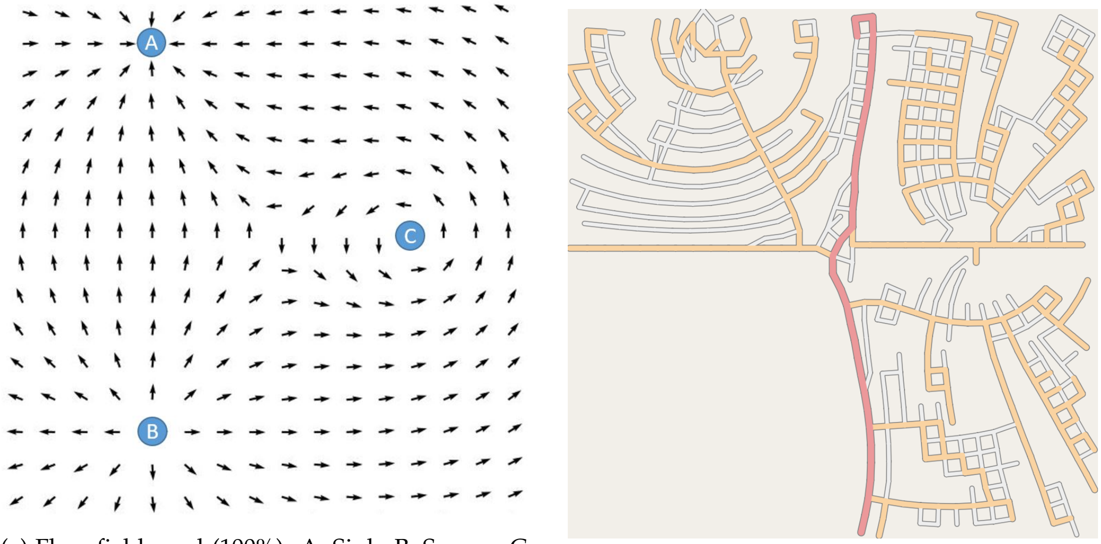
The implementation of the flow field in this approach is in two-dimensions, fitting the terrain of the road map, so it was needed to convert the cylindrical coordinates into Cartesian coordinates to use it. The flow field relies on a discrete grid instead of the continuous terrain used to generate the roads. For each new primitive added, *prim*, given a grid $[n, m]$, for each grid position, $[i, j]$, the flow field is calculated as in Equation 6. A sink will have the the *strength* as a negative constant, and a vortex has a different θ as $\theta_{vortex} = \theta \pm \frac{\pi}{2}$. Plus means that the flow will be counter-clockwise, minus means it will be clockwise.

$$\begin{gathered} r = \sqrt{(i - prim_i)^2 + (j - prim_j)^2}; \quad s = \frac{strength}{2 \times \pi \times r}; \quad \theta = \arctan(i - prim_i, j - prim_j); \\ flowfield_{i,j} \mathrel{+}= (s \times \cos\theta, s \times \sin\theta); \end{gathered} \tag{6}$$

Considering using the flow field to create road layouts, for each new segment, a direction is obtained from the flow field considering the position of the new node, and the segment is snapped to either the flow field direction, or its perpendicular, whichever is closer to the segment's angle.

Flow fields can also be combined with angular restrictions, as described in Section 4.1.2, with a linear combination using a ratio of influence. For instance, if we want to have a flow field influence of 90%, we may want to set the resulting direction vector to the flow field direction vector times 0.9 plus the suggested direction using angular constraints times 0.1.

Figure 89 shows a flow field created using the referred three primitives.



(a) Flow field used (100%): A: Sink; B: Source; C: Vortex.

(b) Road map

Figure 89: Use of flow fields to generate roads

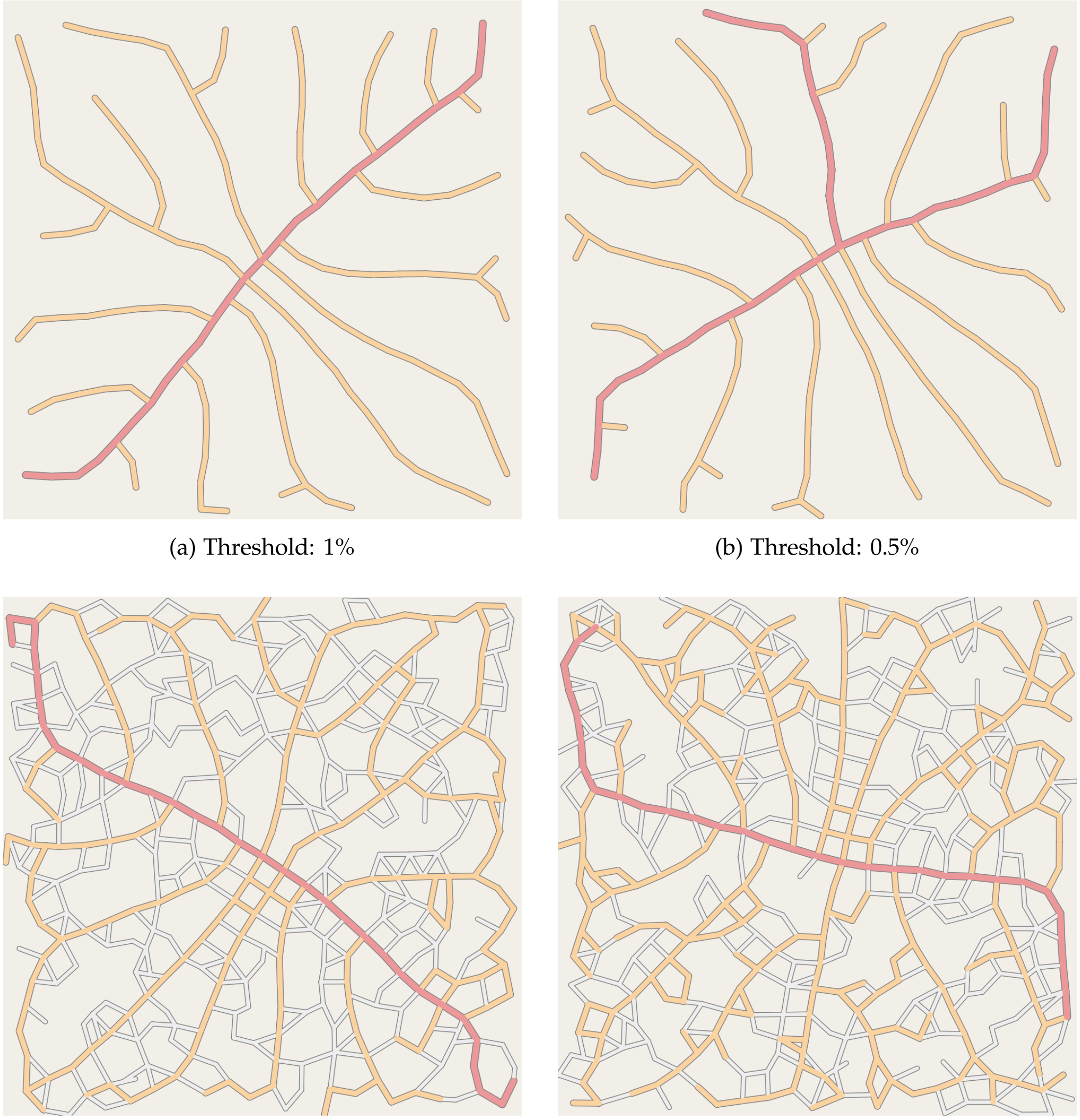
4.3.3 *Road hierarchy*

In real-world road networks, there is a hierarchy that classifies roads by their relevance. An automated procedure creates this hierarchy, classifying roads based on the number of attraction points consumed by their segments. It can cope with any number of hierarchical levels based on user defined thresholds. It is possible to further restrict road hierarchical classification per phase, for instance, considering roads created in the first phase as being either primary or secondary, and relegate all roads created in the second phase to being tertiary roads, reducing the number of threshold parameters to one. Furthermore, it is possible to define the highest hierarchy level for the second phase. For instance, in most of the presented examples the second phase only produces secondary and tertiary roads.

Using attraction point count per segment as the base for defining a hierarchical level is an intuitive approach that is directly related to the density of attraction points in specific areas. As seen in 4.3.1, the distribution of attraction points can be made to match population density providing more important roads in higher population density areas.

The road hierarchy algorithm starts by considering the initial segments defined by the user as primary roads. Road classification is performed on each new segment as follows: when the new segment implies a bifurcation, it evaluates the road relevance for the new segment depending on the number of attraction points that influenced the creation of the segment; otherwise, when the segment is just extending the previous road, it sets its relevance equal to the previous segment. It is also convenient to ensure that the relevance of a new segment can never be set higher than the previous segment. This approach tends to create long primary roads, and shorter roads with less relevance. Let *attppercentage* be the percentage of the attraction points influencing a node, considering all the introduced attraction points (both the active and the deleted ones). This value will be used to distinguish between different hierarchies using the given thresholds. It is given in percentage so that it does not depend from the number of attraction points initially used. Because this takes into account all the attraction points introduced, these threshold values to have any impact in the road generation, are mostly small.

Figure 90 shows four examples of the automatic road classification, a) and b) show the impact in the first phase, and c) and d) show the impact in the second phase. By decreasing the thresholds in both phases, more high hierarchy roads were generated, an additional primary road was generated in Figure 90 b) and multiple secondary roads were generated in Figure 90 d).



(a) Threshold: 1%

(b) Threshold: 0.5%

(c) Threshold: 1%

(d) Threshold: 0.5%

Figure 90: Automatic road hierarchy: red to primary roads; yellow to secondary roads; grey to tertiary roads. a) - b): First phase threshold experiment; c) - d): Second phase threshold experiment.

5

RESULTS

In this chapter, it will be presented how the experimentation was performed, the workflow to produce results, and finally the results obtained and its analysis.

5.1 GRAPHICAL TOOLBOX AND MAP RENDERING

To test the extended SCA, a graphical toolbox was developed. The development of the toolbox and the extensions to the SCA were done simultaneously. This toolbox consists in a desktop software built in C++, using OpenGL (The Khronos Group, Inc) as a graphics library and ImGUI (Omar Cornut) as a GUI. Through this toolbox, the user can experiment with our algorithm, load and see the information maps, scatter the attraction points and tree nodes both automatically and manually, parameterise the algorithm, create a flow field, see the elevation of the height map represented in a 3D mesh, and observe, iteratively, the road generation process, which can be saved in a GeoJSON file at any moment.

The goal of the development of this toolbox is to ease the process of experimentation, however, the visual result of the generated network is rather raw. To produce a proper map corresponding to the generated network, a geospatial visualization and processing tool is used to convert the GeoJSON (Internet Engineering Task Force) file to a raster image map.

The process of obtaining the results is described in Figure 91.

5.1.1 *Toolbox*

As said before, the toolbox has an extensive set of options that helps the user to test the algorithm with a high level of customization. The available panels are shown in Figure 92. Figure 93 shows the toolbox being used in a scene with information maps and elevation. We may see that it allows the user to see both the 2D and 3D road network.

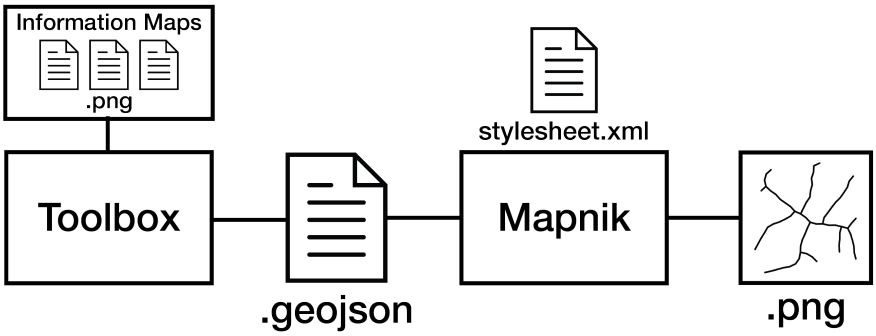


Figure 91: Experimentation workflow

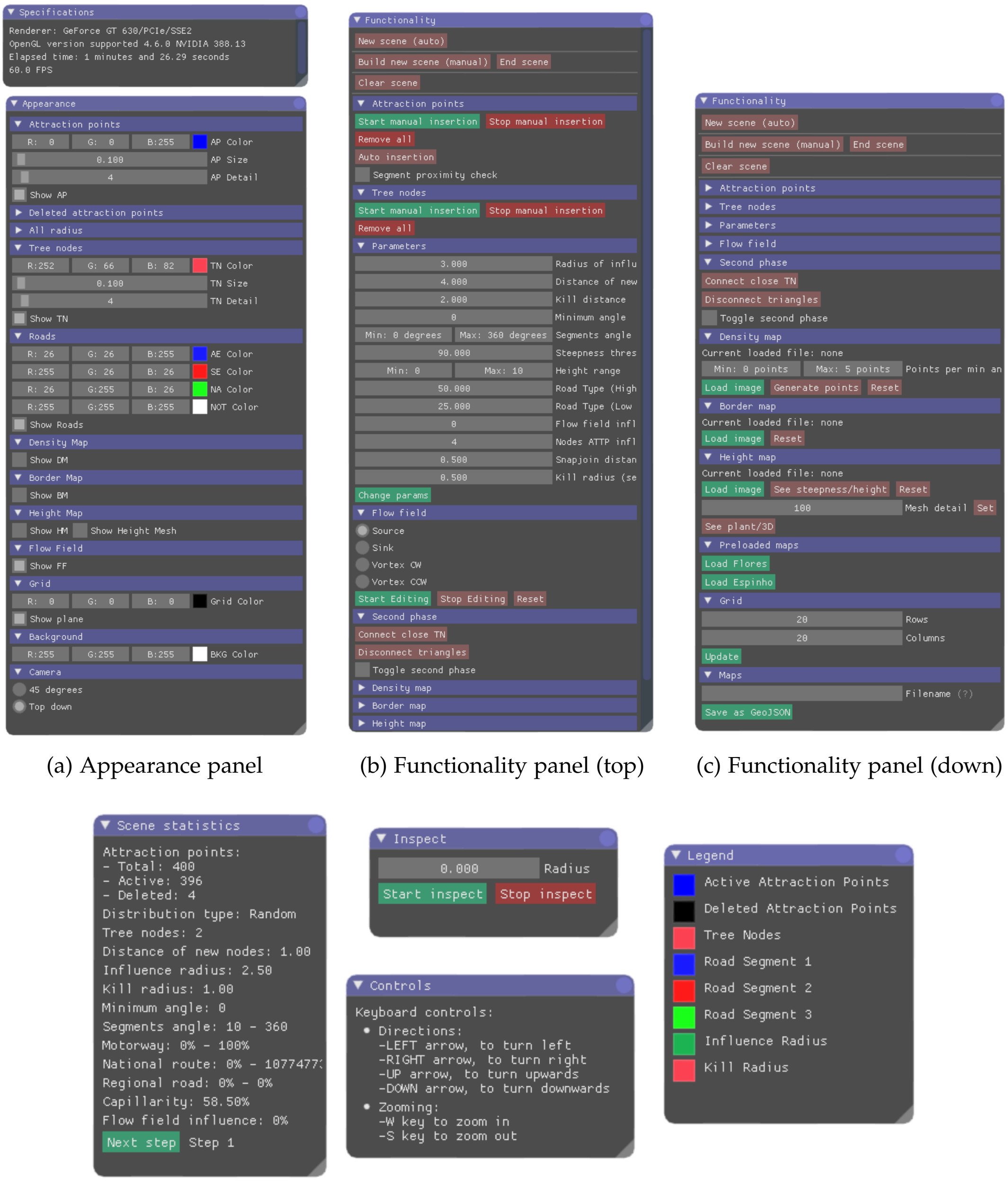
5.1.2 *Map rendering*

To save the previously generated network using the toolbox, the GeoJSON format was chosen. The GeoJSON format is an extension of the JSON files to represent geographical features and its attributes. With GeoJSON, it is possible to represent points, lines, polygons and multi-part primitives of the previously stated ones. These primitives may be used to represent real-life structures such as locations, streets, roads, countries, water and land divisions, and so on.

The generated GeoJSON will save the parameters that created the network, the location of the attraction points, the location of the flow field primitives and the actual flow field and the road network (with its hierarchy categorization). The GeoJSON file will be used by a framework called Mapnik (Artem Pavlenko) which allows to easily create customisable maps. This framework is lightweight, and so, with a simple Python script and a XML *stylesheet*, it was possible to generate the corresponding map from the GeoJSON file. The use of this framework alongside the toolbox, allowed for a more fluid and fast way of obtaining polished and pleasing results. Another advantage of standardizing the results to GeoJSON, is that it can be used and be visualized by other GIS systems.

5.2 FLOW FIELDS

The first result, shown in Figure 94, is the result of using a simple flow field, with few primitives of each type, as seen in Figure 94 a), and by using the parameters described below. The source (B) placed in the right-center of the map will prevent any suggested road to grow towards it. This will cause that zone to being completely unreachable by any road. In Figure 94, a sink (A) is placed in the top-left and top-center side of the map,



```
▼ Specifications
Renderer: GeForce GT 630/PCIe/SSE2
OpenGL version supported 4.6.0 NVIDIA 388.13
Elapsed time: 1 minutes and 26.29 seconds
60.0 FPS
```

```
▼ Appearance
 ▼ Attraction points
  R:  0   G:  0   B:255  AP Color
  0.100                  AP Size
  4                      AP Detail
  □ Show AP
 ▶ Deleted attraction points
 ▶ All radius
 ▼ Tree nodes
  R:252   G: 66   B: 82  TN Color
  0.100                  TN Size
  4                      TN Detail
  □ Show TN
 ▼ Roads
  R: 26   G: 26   B:255  AE Color
  R:255   G: 26   B: 26  SE Color
  R: 26   G:255   B: 26  NA Color
  R:255   G:255   B:255  NOT Color
  □ Show Roads
 ▼ Density Map
  □ Show DM
 ▼ Border Map
  □ Show BM
 ▼ Height Map
  □ Show HM  □ Show Height Mesh
 ▼ Flow Field
  □ Show FF
 ▼ Grid
  R:  0   G:  0   B:  0  Grid Color
  □ Show plane
 ▼ Background
  R:255   G:255   B:255  BKG Color
 ▼ Camera
  ○ 45 degrees
  ● Top down
```

(a) Appearance panel

```
▼ Functionality
 New scene (auto)
 Build new scene (manual)  End scene
 Clear scene
 ▼ Attraction points
  Start manual insertion  Stop manual insertion
  Remove all
  Auto insertion
  □ Segment proximity check
 ▼ Tree nodes
  Start manual insertion  Stop manual insertion
  Remove all
 ▼ Parameters
  3.000                          Radius of influ
  4.000                          Distance of new
  2.000                          Kill distance
  0                              Minimum angle
  Min: 0 degrees  Max: 360 degrees  Segments angle
  90.000                         Steepness thres
  Min: 0          Max: 10        Height range
  50.000                         Road Type (High
  25.000                         Road Type (Low
  0                              Flow field infl
  4                              Nodes ATTP infl
  0.500                          Snapjoin distan
  0.500                          Kill radius (se
  Change params
 ▼ Flow field
  ● Source
  ○ Sink
  ○ Vortex CW
  ○ Vortex CCW
  Start Editing  Stop Editing  Reset
 ▼ Second phase
  Connect close TN
  Disconnect triangles
  □ Toggle second phase
 ▶ Density map
 ▶ Border map
 ▶ Height map
```

(b) Functionality panel (top)

```
▼ Functionality
 New scene (auto)
 Build new scene (manual)  End scene
 Clear scene
 ▶ Attraction points
 ▶ Tree nodes
 ▶ Parameters
 ▶ Flow field
 ▼ Second phase
  Connect close TN
  Disconnect triangles
  □ Toggle second phase
 ▼ Density map
  Current loaded file: none
  Min: 0 points  Max: 5 points  Points per min an
  Load image  Generate points  Reset
 ▼ Border map
  Current loaded file: none
  Load image  Reset
 ▼ Height map
  Current loaded file: none
  Load image  See steepness/height  Reset
  100                     Mesh detail  Set
  See plant/3D
 ▼ Preloaded maps
  Load Flores
  Load Espinho
 ▼ Grid
  20                      Rows
  20                      Columns
  Update
 ▼ Maps
                          Filename (?)
  Save as GeoJSON
```

(c) Functionality panel (down)

```
▼ Scene statistics
Attraction points:
- Total: 400
- Active: 396
- Deleted: 4
Distribution type: Random
Tree nodes: 2
Distance of new nodes: 1.00
Influence radius: 2.50
Kill radius: 1.00
Minimum angle: 0
Segments angle: 10 - 360
Motorway: 0% - 100%
National route: 0% - 1077477
Regional road: 0% - 0%
Capillarity: 58.50%
Flow field influence: 0%
Next step  Step 1
```

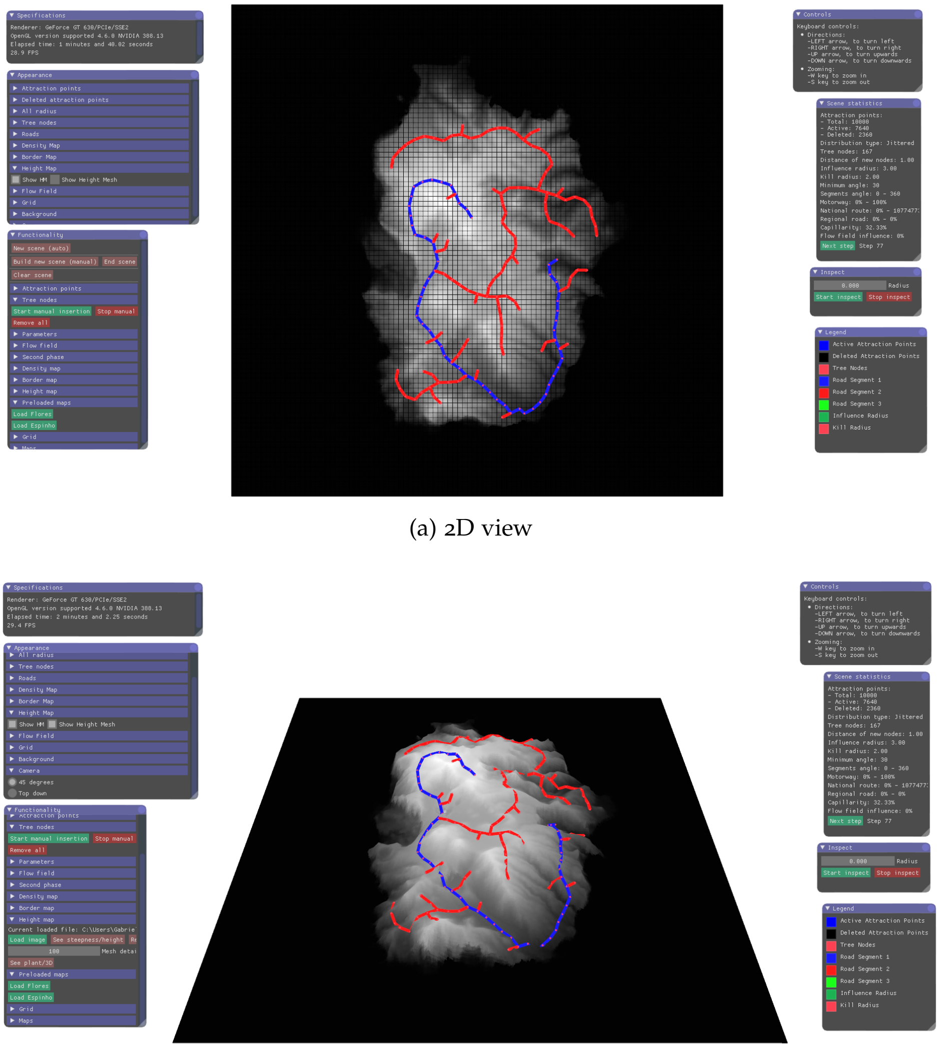
```
▼ Inspect
 0.000          Radius
 Start inspect  Stop inspect
```

```
▼ Controls
Keyboard controls:
 • Directions:
   -LEFT arrow, to turn left
   -RIGHT arrow, to turn right
   -UP arrow, to turn upwards
   -DOWN arrow, to turn downwards
 • Zooming:
   -W key to zoom in
   -S key to zoom out
```

```
▼ Legend
 🟦 Active Attraction Points
 ⬛ Deleted Attraction Points
 🟥 Tree Nodes
 🟦 Road Segment 1
 🟥 Road Segment 2
 🟩 Road Segment 3
 🟩 Influence Radius
 🟥 Kill Radius
```

(d) Statistics and help panels

Figure 92: Toolbox panels



```
Specifications
Renderer: GeForce GT 630/PCIe/SSE2
OpenGL version supported 4.6.0 NVIDIA 388.13
Elapsed time: 1 minutes and 40.02 seconds
28.9 FPS
```

```
Controls
Keyboard controls:
 • Directions:
   -LEFT arrow, to turn left
   -RIGHT arrow, to turn right
   -UP arrow, to turn upwards
   -DOWN arrow, to turn downwards
 • Zooming:
   -W key to zoom in
   -S key to zoom out
```

```
Scene statistics
Attraction points:
- Total: 10000
- Active: 7640
- Deleted: 2360
Distribution type: Jittered
Tree nodes: 167
Distance of new nodes: 1.00
Influence radius: 3.00
Kill radius: 2.00
Minimum angle: 30
Segments angle: 0 - 360
Motorway: 0% - 100%
National route: 0% - 1077477
Regional road: 0% - 0%
Capillarity: 32.33%
Flow field influence: 0%
Next step Step 77
```

(a) 2D view

```
Specifications
Renderer: GeForce GT 630/PCIe/SSE2
OpenGL version supported 4.6.0 NVIDIA 388.13
Elapsed time: 2 minutes and 2.25 seconds
29.4 FPS
```

Current loaded file: C:\Users\Gabrie

```
Controls
Keyboard controls:
 • Directions:
   -LEFT arrow, to turn left
   -RIGHT arrow, to turn right
   -UP arrow, to turn upwards
   -DOWN arrow, to turn downwards
 • Zooming:
   -W key to zoom in
   -S key to zoom out
```

```
Scene statistics
Attraction points:
- Total: 10000
- Active: 7640
- Deleted: 2360
Distribution type: Jittered
Tree nodes: 167
Distance of new nodes: 1.00
Influence radius: 3.00
Kill radius: 2.00
Minimum angle: 30
Segments angle: 0 - 360
Motorway: 0% - 100%
National route: 0% - 1077477
Regional road: 0% - 0%
Capillarity: 32.33%
Flow field influence: 0%
Next step Step 77
```

(b) 3D view

Figure 93: Toolbox views

attracting roads to that direction. The vortex (C) placed in the bottom left and right part of the map, generates a concentric flow field in that area. We may see that roads generated in the vortex area, tend to be concentric/radial by following the vortex generated direction or its perpendicular.

The result shown in Figure 95, shows a denser network with a more complex flow field, where the observations stated above are also noticeable. The sources placed in the map, allowed to create empty areas, just like when dealing with forbidden areas such as parks, existing buildings and other important areas. Vortexes will shape the curvature of the roads in the area they are placed by forcing a radial/concentric layout, like in the right side of the map. Sinks will force the network to grow towards them.

- Radius of influence: 5.0
- Segment length: 1.0
- Kill distance (first phase/second phase): 3.0/1.5
- Minimum angle: 30°
- Number of nodes influenced per attraction point in the second phase: 4
- Attraction points used per phase: 2500
- Road hierarchy thresholds: 1% and 1%
- Flow field intensity: 100%

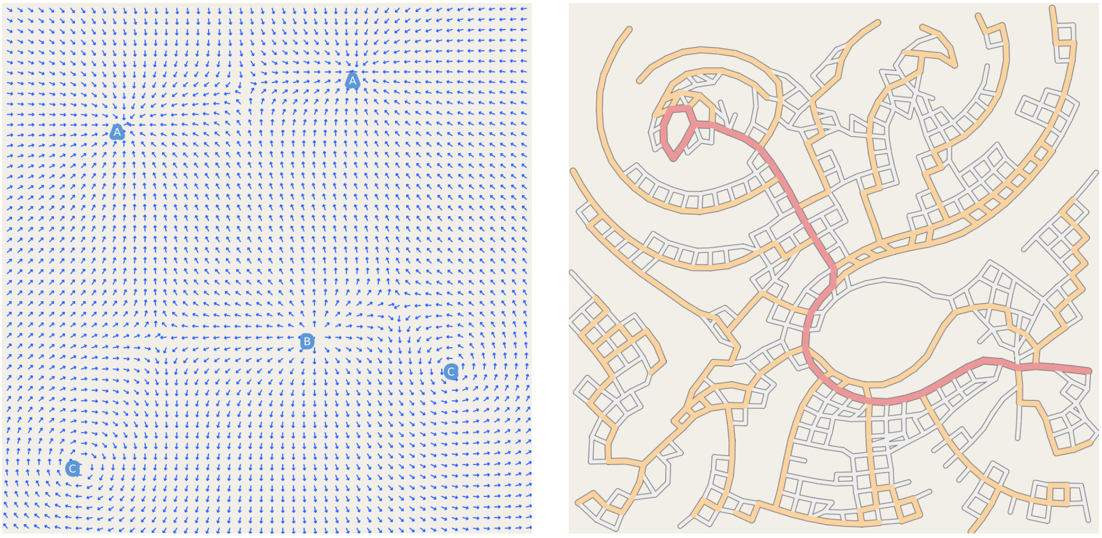
5.3 GRID-LIKE PATTERN

The grid-like pattern exists in many of the recent formed cities, sometimes called *the manhattan pattern*. This pattern consists in having most of its roads perpendicular to each other.

To generate this pattern, the angle was fixed to 90°, and the forward direction is also eligible.

The first test done was regarding the kill distance. The parameters to generate the results in Figure 96 and Figure 97 were:

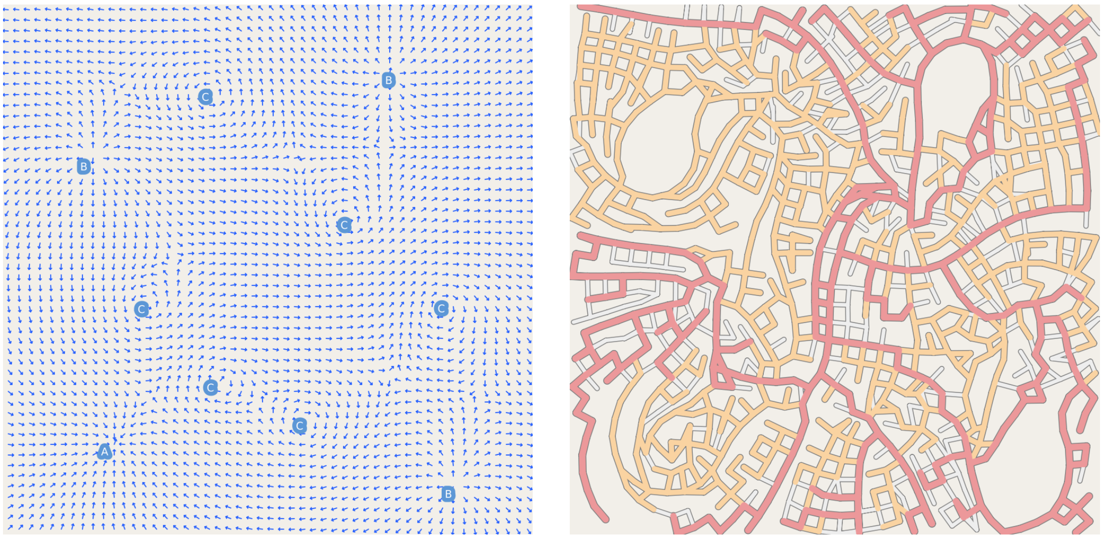
- Radius of influence: 5.0
- Segment length: 1.0
- Number of nodes influenced per attraction point in the second phase: 4



(a) A: Sink; B: Source; C: Vortex

(b)

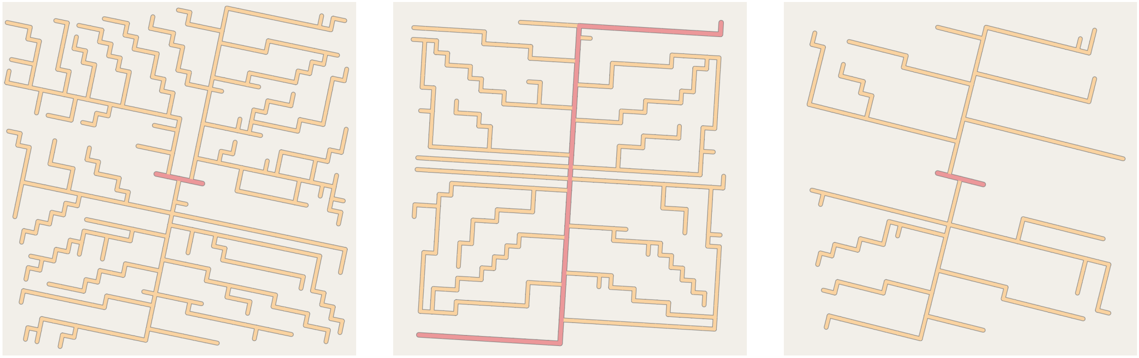
Figure 94: A flow field (*a)*) and the road layout generated.



(a) A: Sink; B: Source; C: Vortex

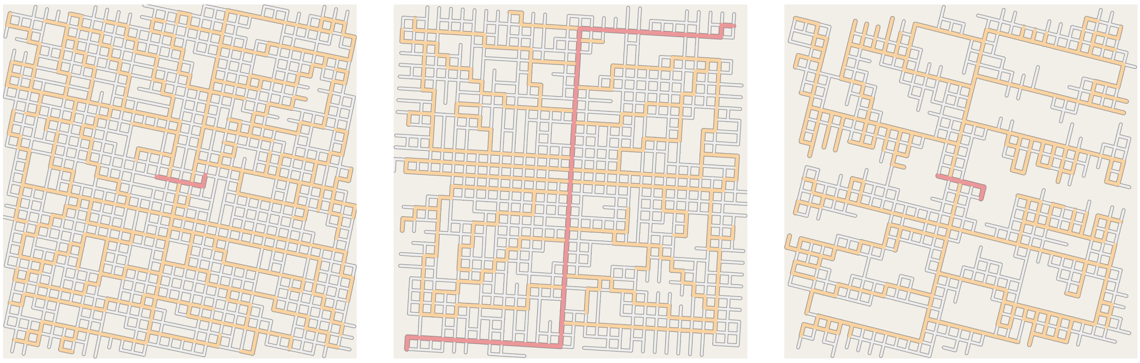
(b)

Figure 95: A more complex flow field (*a)*) and the road layout generated.



(a) (b) (c)

Figure 96: First phase: Kill distance: a) 2; b) 3. c) 4.



(a) (b) (c)

Figure 97: Second phase: Kill distance: a) 1.0; b) 1.5; c) 2.0.

- Attraction points used per phase: 2500
- Road hierarchy thresholds: 1% and 1%

To draw some conclusions regarding different values of the kill distance, three different road maps were generated. As the kill distance increases, the percentage of dead-ends in the network increases. The road density also decreases, meaning that the smaller the kill distance is, the denser the network becomes.

It may be stated that the ratio between the kill distance and the radius of influence, the capillarity, is extremely important to control the output in terms of road density and connectivity ratio.

The angle restriction may not be wanted in both phases of the algorithm. To add some randomness and to appear more organic, the angle restriction in the second phase can be toggled off. In this case, it can be expected that the secondary and tertiary roads connecting the previously generated roads may have different angles.

In Figure 98, the radius of influence was increased to 5, the kill distance set to 1, and the angle restriction was disabled in the second phase. The radius of influence was increased

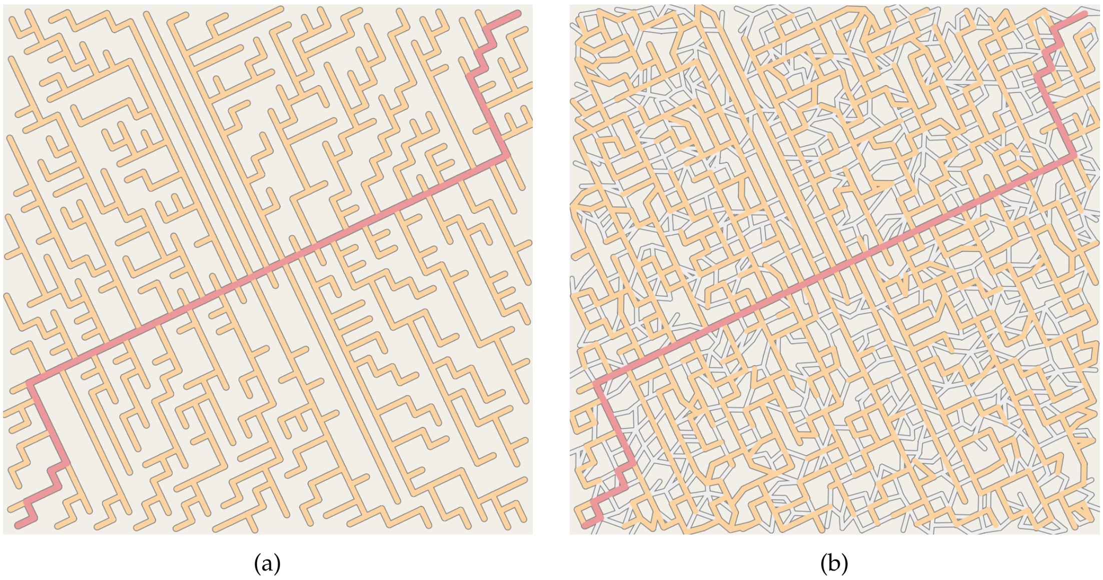


Figure 98: a) First phase; b) Second phase.

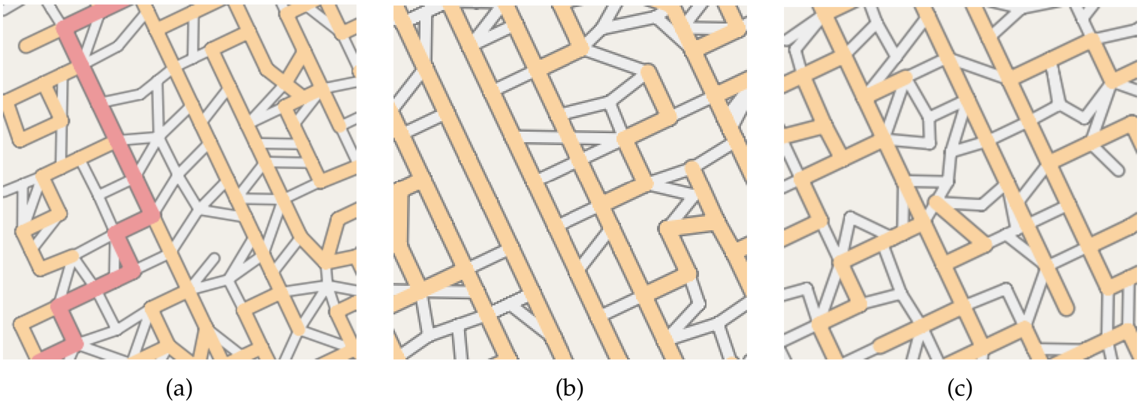


Figure 99: One phase grid pattern details: a) perpendicular/rigid connections; b) organic connections.

to have a denser network as shown in Figure 98 a). The second phase without the angle restrictions makes the road network more diverse and inter-connected than the previous examples. In Figure 99 it may be seen that it still can create perpendicular connections, due to snapping/merging, but it also creates segments with different angle values.

5.4 RADIAL PATTERN

The radial pattern mostly found in old cities is one of the most well-known. It is defined by having primary roads forming a loop in the outer part of the city and also expanding inwards in a radial way. They are inter-connected using secondary and tertiary roads. Its use ranges from small to large cities. Nowadays, expanding cities still keep this layout. Figure 16, in section 2.2 shows three different cities where this layout is used.

5.4.1 *Using Angular Constraints*

The first approach was to try to generate a radial layout using only angular constraints. The initial road was placed in a circular shape encircling a large area with attraction points inside it.

The parameters used were:

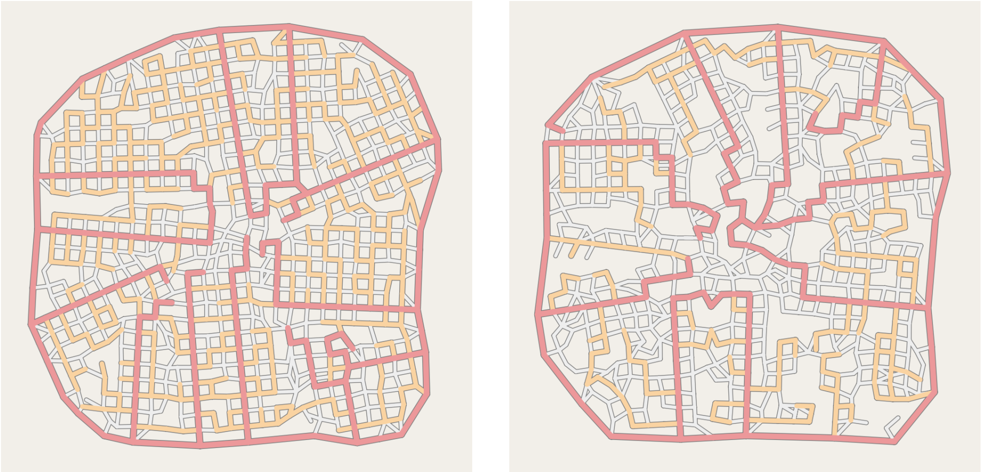
- Radius of influence: 3.0
- Segment length: 1.0
- Valid angle range: 90° (first phase) / $80^\circ - 110^\circ$ (second phase)
- Number of nodes influenced per attraction point in the second phase: 4
- Attraction points used per phase: 2500
- Road hierarchy thresholds: 1% and 1%

It was intended to test and generate two different radial layouts. In both, the angle was restricted to 90° in the first phase to generate a concentric network, with roads growing towards the center, and with secondary roads perpendicular to the primary ones. In the second phase, the angle restriction range was relaxed to allow the tertiary roads to be more organic. The first one, in Figure 100 a), was generated with a lower kill distance to generate a dense network in the first phase. Both the primary and secondary roads have a grid-like pattern, but in the second phase, the new connections will make the network more diverse due to the angle restriction relaxation. In Figure 100 b), the kill distance was increased to generate fewer primary and secondary roads, so that in the second phase, the tertiary roads become the most generated roads of the network. It was intended to generate a radial road map more organic than the previous one.

Both road maps presented a radial pattern also combining with a grid-like pattern or with a more loose tertiary road network.

5.4.2 *Adding Flow Fields*

When using a flow field, the suggested direction can be direction of the flow field or its perpendicular. As noticed in the real road radial layouts, the roads tend to grow towards the center and they are connected between themselves using roads perpendicular to them.



(a)

(b)

Figure 100: Kill distance (first phase/second phase): a) 1/0.5; b) 2/1;

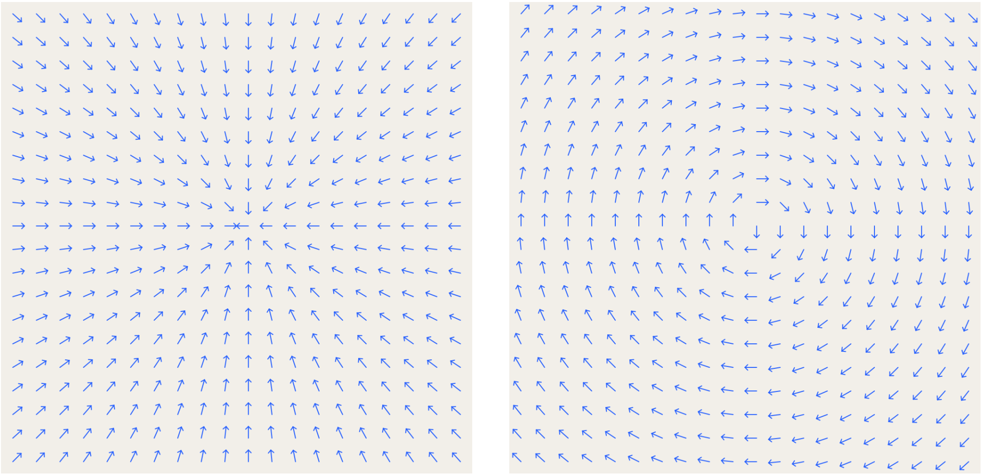
There are two flow fields primitives that generate fields as described: sinks and vortexes. A sink will generate an attractive flow field towards it. A vortex will generate a concentric flow field around a point. Figure 101 presents the flow field corresponding to each one.

From the parameters used in section 5.4.1, using a kill distance of 0.5/0.25 (first phase/second phase), with a sink and a vortex, we obtained the results presented in Figure 102. In this case, since segments are allowed to grow either along the flow direction or perpendicular to it these primitives are identical.

Both results start from a primary road that encircles the scene. Attraction points are placed inside the starting road. In both maps, multiple primary roads grow towards the center, some of them are connected between themselves by a primary road as well. Secondary roads tend to be perpendicular to them and a small number of tertiary roads are created to make extra connections. The parameters used were purposely aimed to generate dense networks. Both results present a highly connected and dense network. Comparing these results with Figure 100 we see that the network presents a more evenly distributed connections throughout the map.

To obtain a sparser network, the kill distance was now increased to 2 and 1 (first phase/second phase). Both primitives were tested again. The generated road maps are presented in Figure 103. The road map using the sink, successfully connected four different primary roads from different sides of the map in its center. Secondary roads and tertiary roads are less abundant than in the previous maps due to increasing of the kill distance, yet, due to considering a considerably high amount of nodes as attraction point in the second phase, the generated roads are still numerous.

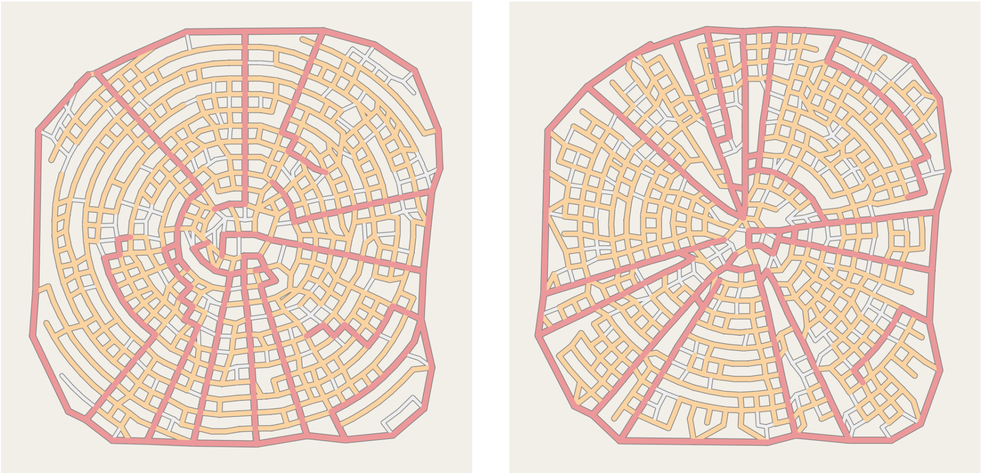
Using the same parameters used above in Figure 103, but varying the amount of attraction points, Figure 104 was generated. Both maps in Figure 104 were generated using sinks. By



(a)

(b)

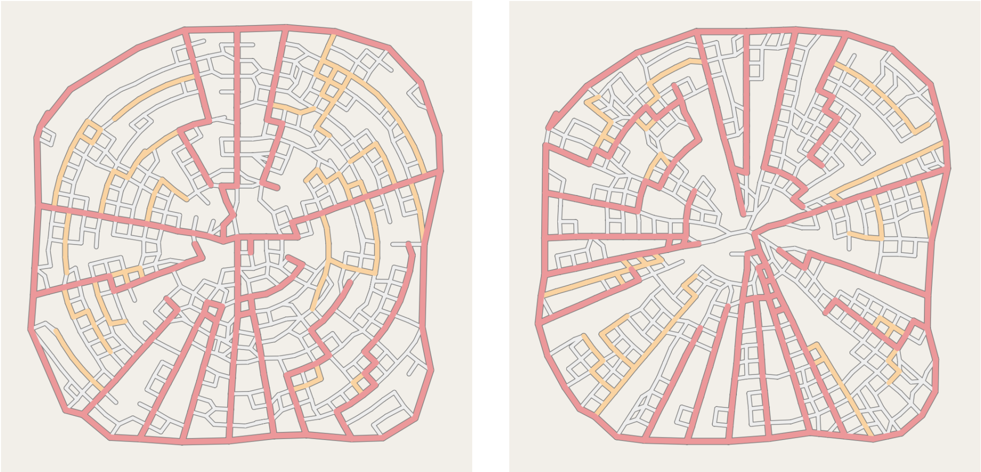
Figure 101: Flow fields generated using: a) Sink; b) Vortex (Clockwise);



(a)

(b)

Figure 102: Radial road maps generated using: a) Sink; b) Vortex (Clockwise);



(a)

(b)

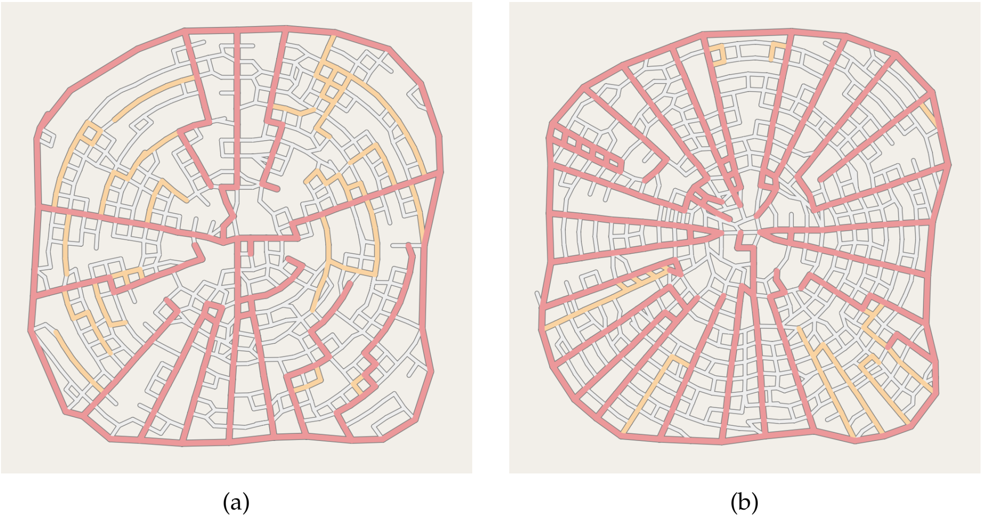
Figure 103: Radial road maps generated using: a) Sink; b) Vortex (Clockwise);

increasing the attraction points, Figure 104 b) presents less roads that do not follow the flow field direction due to snapping and merging operations. The high density of attraction points results in a homogenised network with few roads having different directions. Due to keeping the road hierarchy thresholds as before the map will present more primary roads, because each road is influenced by more attraction points than before.

In smaller cities the radial layout may only be noticed in the most important roads. To generate a similar road map, this time the flow field will only be used in the first phase. The kill distance was set to 2, the radius of influence to 3 to generate a sparse network in the first phase. In the second phase, the kill distance was decreased to 0.5 to generate a dense network with a large number of tertiary roads. The minimum angle in the second phase was set to 70° to limit the roads to be more perpendicular and straight when connecting the existing roads, but also allowing different angles to appear more organic. The road map generated is presented in Figure 105. As expected, this *mixed* generation allows a stricter and straighter first phase primary and secondary network, while the tertiary roads in the second phase connect the first phase roads in a more organic radial way without compromising its density or connectivity. This layout can be used in rural environments or in historical areas, where the roads may not be too symmetrical.

5.5 ESPINHO

To explore how the algorithm performs in a coastal scenario, it was chosen the shoreline of the City of Espinho, Aveiro, Portugal as a case study. Coastal cities present special characteristics in their road network, such as, having a large avenue alongshore and in most cases, minor roads perpendicular to the main avenue.


(a)

(b)

Figure 104: Radial road maps generated using different amounts of attraction points (per phase): a) 2500; b) 10000;

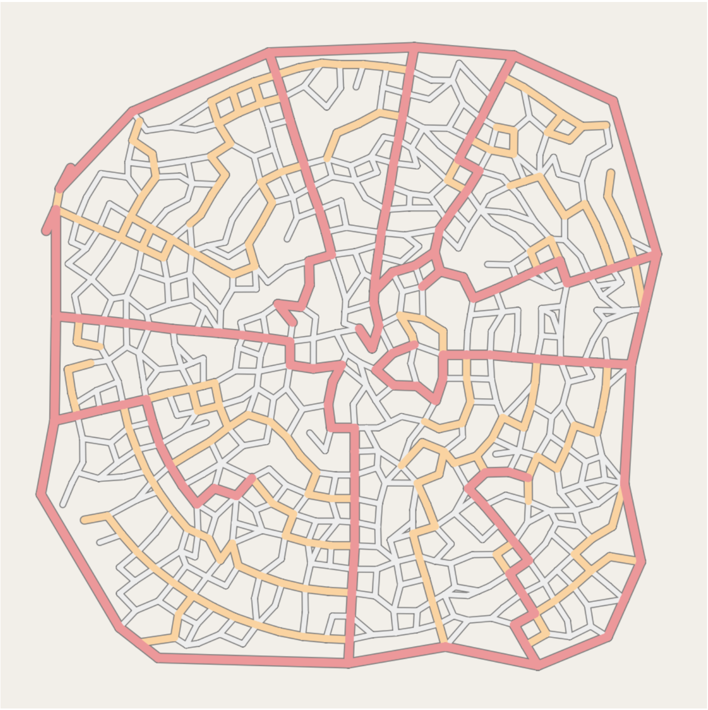
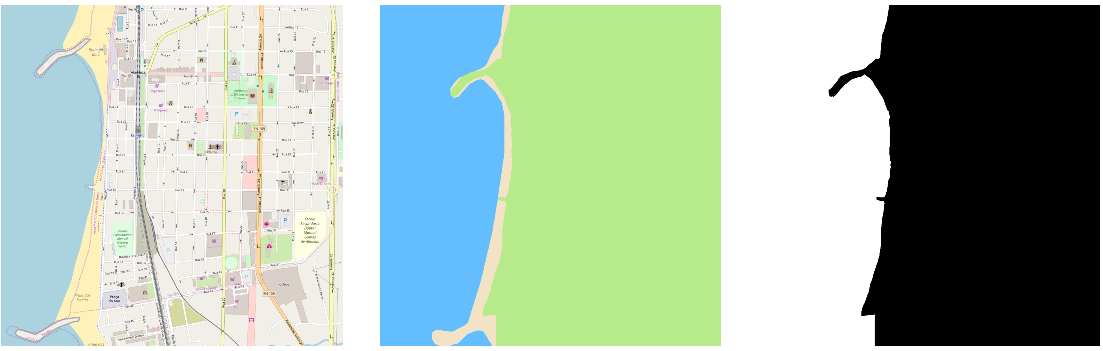


Figure 105: Radial road map using flow field only in the first phase.



(a) City Of Espinho map. Source: OpenStreetMap Foundation

(b) Simple map with three distinct areas.

(c) Border map used by the algorithm (black: land; white: water and sand).

Figure 106: Espinho maps

In Figure 106, it is shown the map of Espinho, a simplified map distinguishing two different land types and sea, and a binary image corresponding to the permitted and forbidden areas. The latter will be used by the algorithm as a border map.

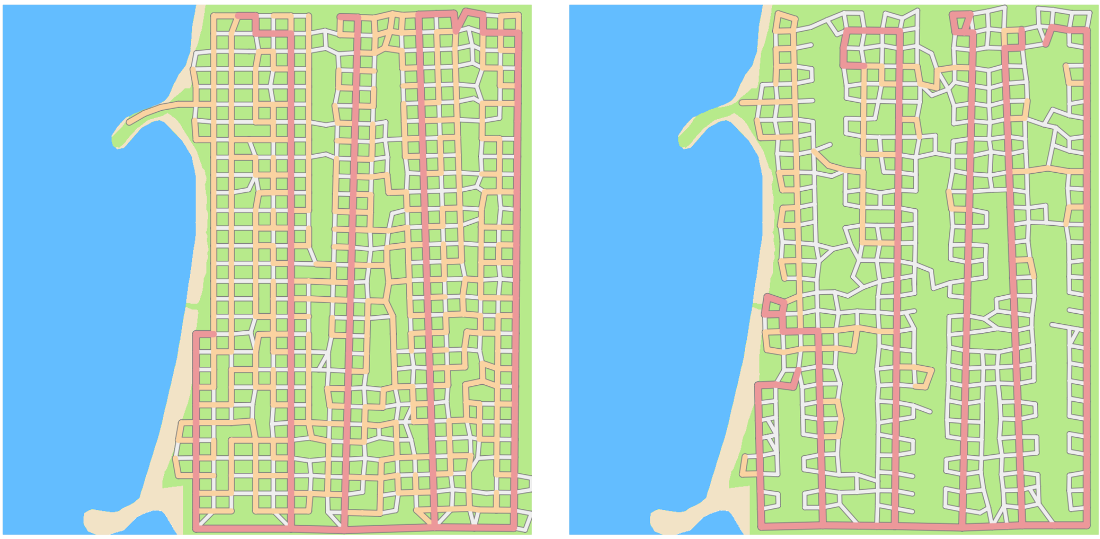
As in the previous section first we present results using only angular constraints, and them introduce examples with flow fields. Every test described in this section will use the attraction point validation method to scatter the second phase attraction points.

5.5.1 *Using Angular Constraints*

In both maps, the angle restriction of the first phase was set to 90° to force the network to grow perpendicularly as the initial road. In the second phase, the angle restriction was decreased to allow different angles in the tertiary roads. Both the road maps present a grid-like pattern.

To create these examples, initial segments were manually placed at the bottom of the map, horizontally. To study the impact of the kill distance, the first tests, seen in Figure 107 were done using the following set of parameters:

- Radius of influence: 4.0
- Segment length: 1.0
- Valid angle range: 90° (first phase) / 80° – 100° (second phase)
- Number of nodes influenced per attraction point in the second phase: 2
- Attraction points used per phase: 2500
- Road hierarchy thresholds: 1% and 1%22



(a) Kill distance (first phase/second phase): 1.0/0.5

(b) Kill distance (first phase/second phase): 2.0/1.0

Figure 107: Espinho road map generated with different kill distances.

Figure 107 a) presents a more dense network and also more shaped as a grid. Figure 107 b), due to having an higher kill distance, generated a sparse first phase network, being complemented with more loose tertiary roads in the second phase.

Regarding the kill distance difference in the two results, with a lower kill distance, the attraction points have a longer lifespan. This will result in a denser road network. Also, the percentage of attraction points influencing a node, will be higher, and this is the reason why there are more secondary roads than when using a higher kill radius. By keeping the same hierarchy threshold, regarding the second phase, we may see that it is enough to change not only the number of roads created but their importance in the network.

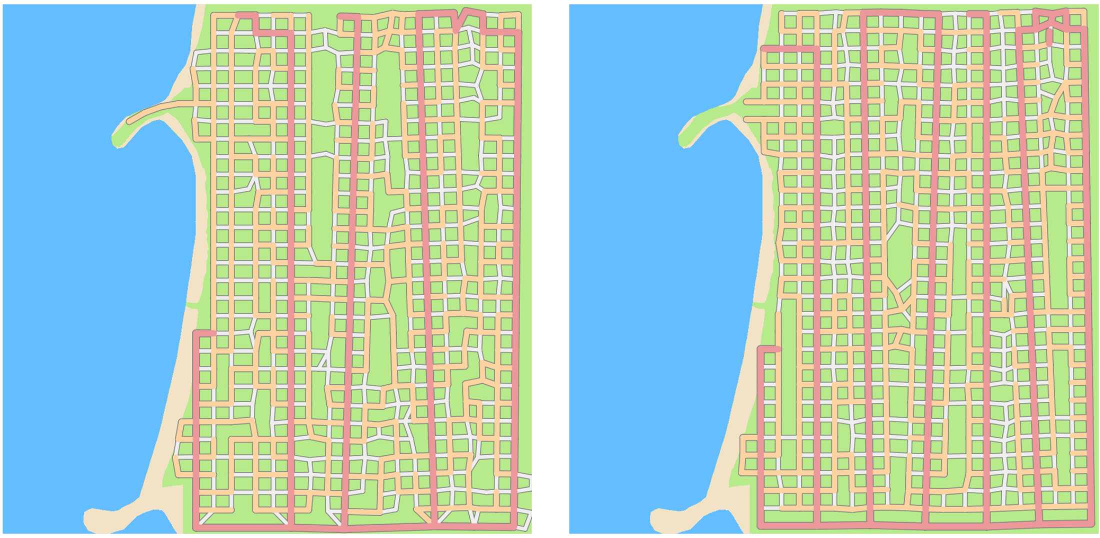
Both road maps can suit well a coastal area. Dense networks could be used for large cities and sparse networks for small coastal towns.

From the set of parameters described above, with a kill distance of 1, it was tested the influence of the number of attraction points in the results. Figure 107 a) was generated with 2500 attraction points, in Figure 108 10000 attraction points were used.

The difference when using an higher amount of attraction points, is that the new roads will not be as different from each other as before, because not only it were used more attraction points but because the distribution is uniform.

5.5.2 *Adding Flow Fields*

To test the influence of flow fields in this example, a flow field was created as shown in Figure 109 a). The flow field consists in having multiple sources along the land border so



(a)

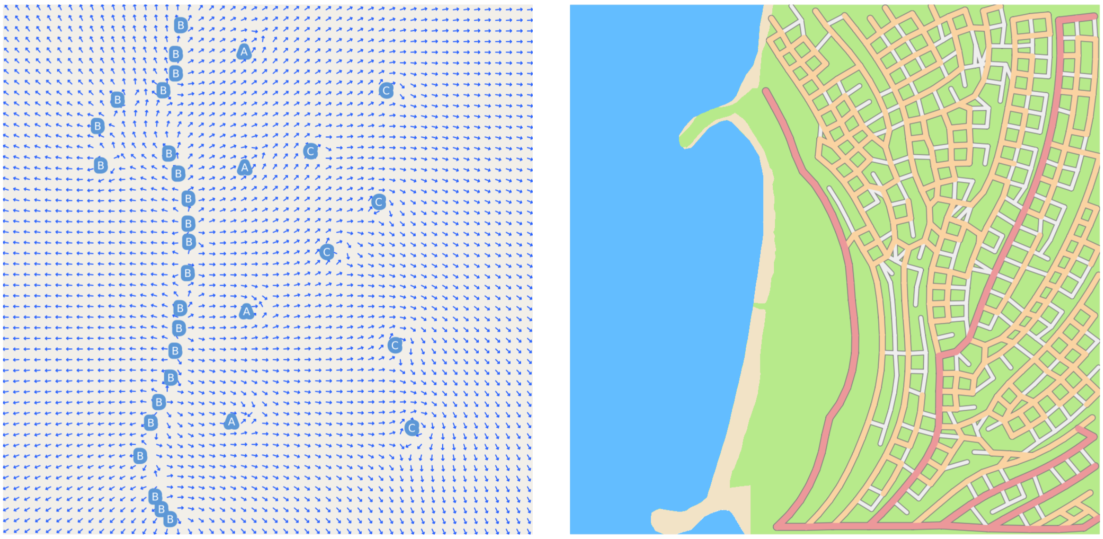
(b)

Figure 108: Espinho road map: a) 2500 attraction points; b) 10000 attraction points.

that the flow field will take into account the terrain shape. More primitives were added such as sinks and vortexes so that the flow does not get too much uniform. From the initial segment at the bottom of the map, it generated a few more primary roads. The one closest to the shore stands out by following the coast shape, resembling the boulevards near coastal cities. Contrasting the organic layout and direction of primary and some secondary roads, most of secondary and tertiary roads follow a grid-like pattern that is also found in cities like Espinho. Below are shown the parameters used to generate the map generated in Figure 109.

- Radius of influence: 4.0
- Segment length: 1.0
- Kill distance: 1.0
- Minimum angle: 10°
- Number of nodes influenced per attraction point in the second phase: 2
- Flow field influence: 100%
- Attraction points used per phase: 2500
- Road hierarchy thresholds: 1% and 1%

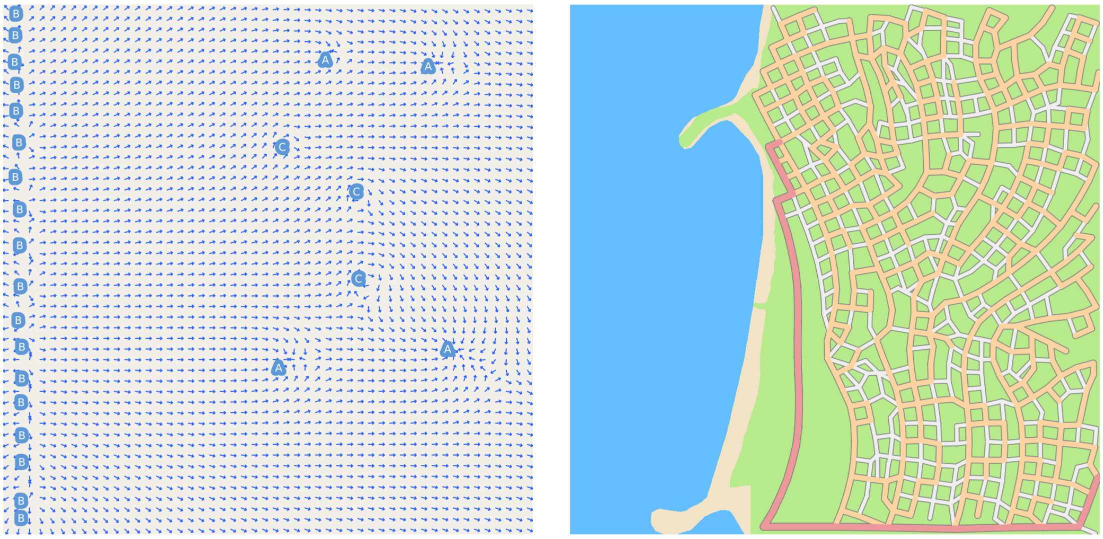
Another example with a different flow field can be seen in Figure 110. The primary road along the coast now grows closer to it. The secondary roads are now more curvy near the sinks and the vortexes, yet the grid-like pattern is also very present among them and tertiary roads.



(a) A: Sink; B: Source; C: Vortex

(b)

Figure 109: A flow field (*a*)) and the road layout generated.



(a) A: Sink; B: Source; C: Vortex

(b)

Figure 110: A flow field (*a*)) and the road layout generated.

5.6 FLORES ISLAND

To observe how the algorithm performed when dealing with elevation data, information maps related to the Flores island, Azores, Portugal, were obtained. In this case study, two information maps were used: a height map and a border map.

To obtain the height data of Flores island, a 30-meter resolution elevation data from the Shuttle Radar Topography Mission by NASA (National Aeronautics and Space Administration) was used, within the N39°00.00′ W32°00.00′ tile where the island is located.

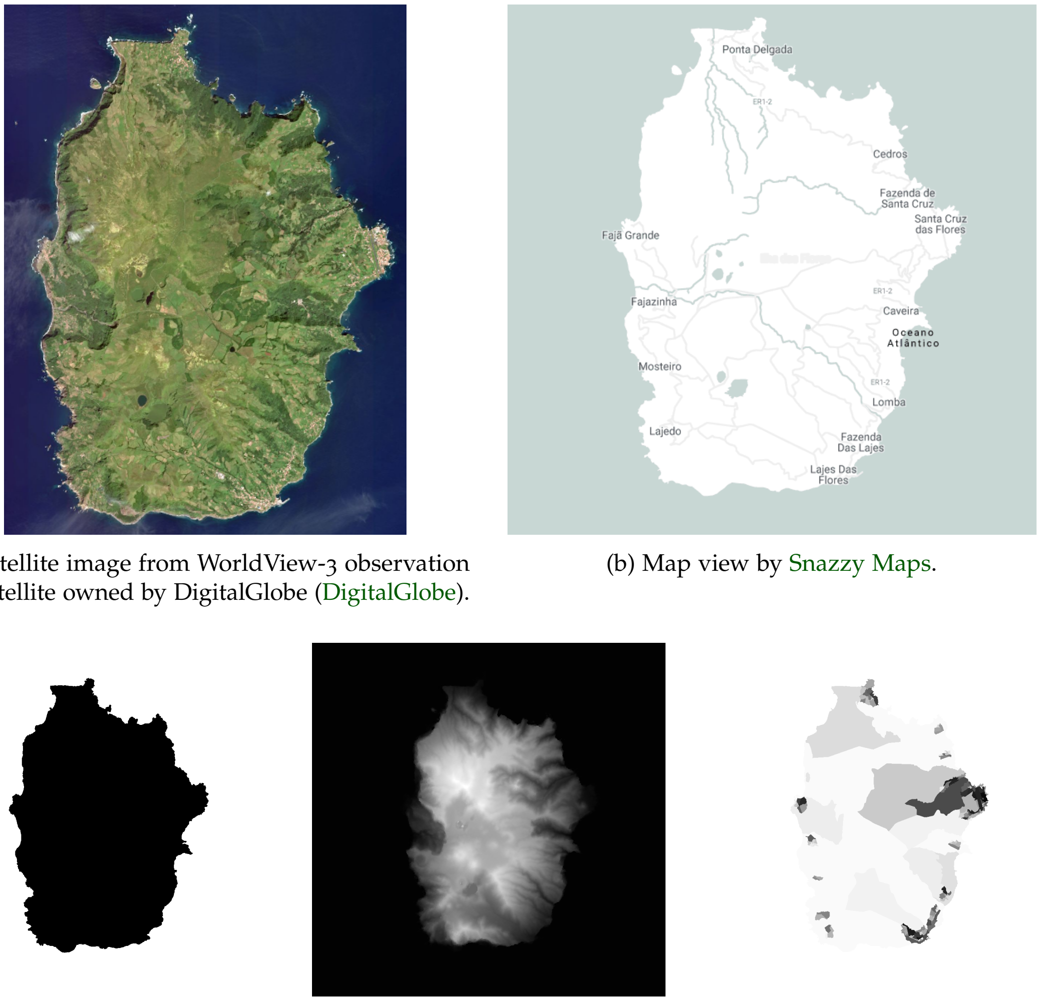
The border map was obtained using a *shapefile* from the Portuguese census of 2011 by Instituto Nacional de Estatística (Instituto Nacional de Estatística). The *shapefile* was then converted to a raster image using an open-source GIS software called QGIS (QGIS). From that data, it was also obtained a density map, where the density increases from white to black. All maps are shown in Figure 111.

The majority of the tests were done using a $100x100$ grid. When using the information maps of Figure 111, it means that 1 unit will correspond to 215 meters approximately.

Figure 113 shows a result of executing only the first phase of the algorithm, aiming to generate a sparse road network, starting from a road placed in the south part of the island. The segment length was set to 1 (215 meters) and the radius of influence to 3 (645 meters). The first road grew along the coast to the north direction until it met a low elevation area in the west side of the island corresponding to the civil parish of Fajãzinha. The secondary roads started to grow from the low elevation areas in the south towards the north by passing through the west side of the island. It was noticed that the secondary roads went around the Morro Alto Natural Reserve which has a relatively high altitude area. The east side of the island was almost untouched by the road network, whereas the north side was included in the network due to it being more irregular regarding height.

The results in Figure 113 were obtained using the following parameters:

- Radius of influence: 3.0
- Segment length: 1.0
- Kill distance (first phase/second phase): 2.5/2.5
- Minimum angle: 30°
- Number of nodes influenced per attraction point in the second phase: 1
- Attraction points used per phase: 20000 (approximately)
- Road hierarchy thresholds: 1% and 0.5%
- Maximum steepness: 10°



(a) Satellite image from WorldView-3 observation satellite owned by DigitalGlobe (DigitalGlobe).

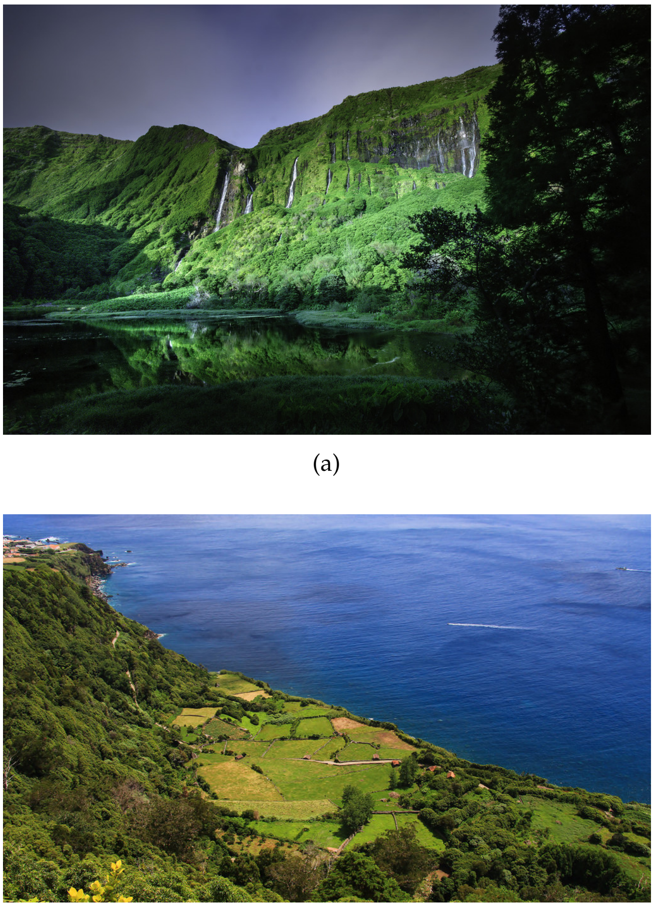
(b) Map view by Snazzy Maps.

(c) Border map.

(d) Height map.

(e) Density map.

Figure 111: Flores island maps



(a)

(b)

Figure 112: Flores island places: a) Fajãzinha (located at west). Source: Rui Pedro Vieira (2012); b) Fajã do Conde (located at east) Source: Ann Collier (2015).

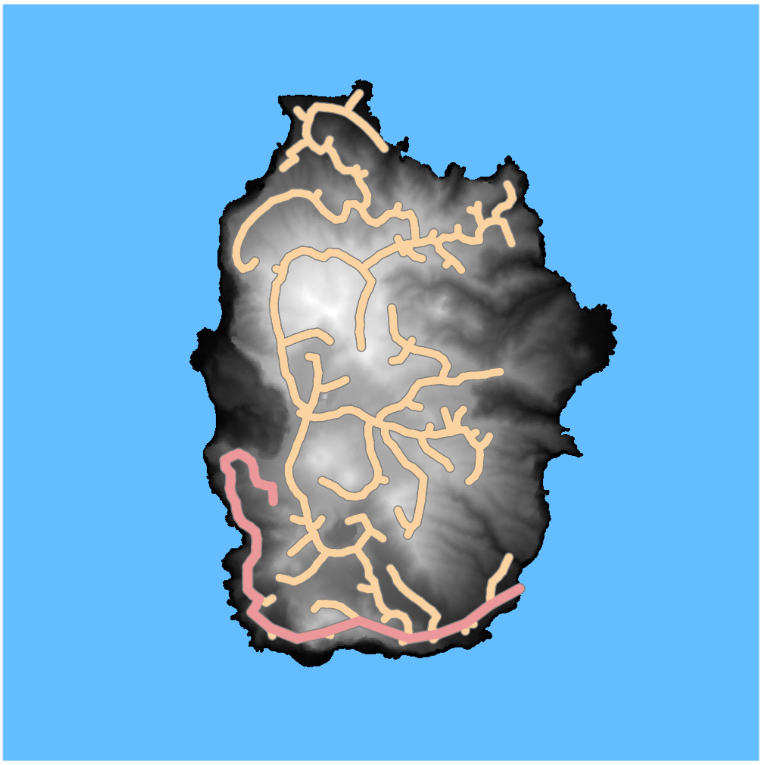


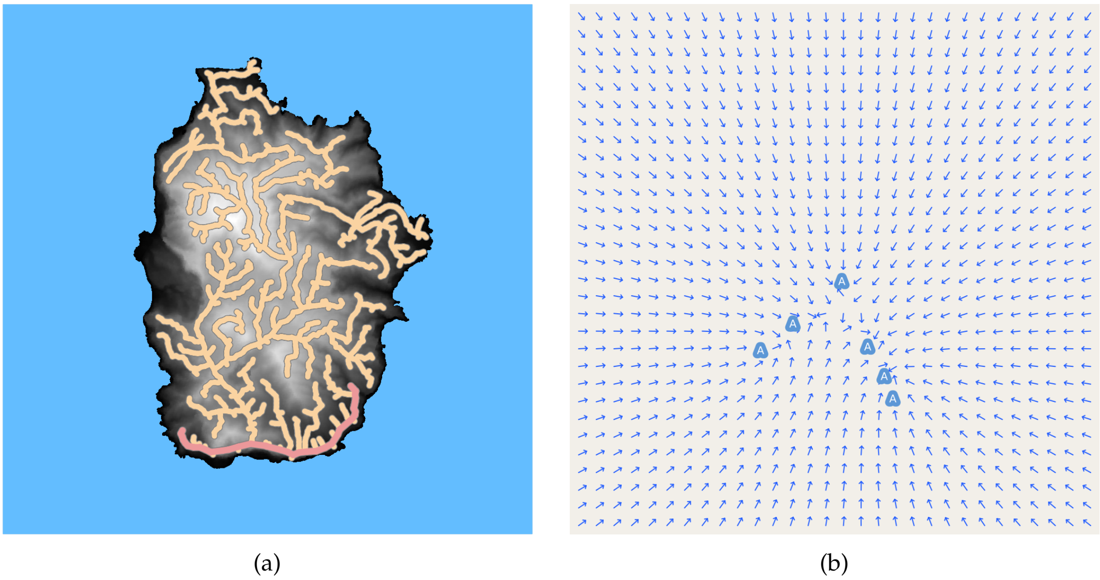
Figure 113: Road map using Flores information maps (first phase only).

The get finer results, the resolution was increased to $150x150$.

Now, 1 unit will correspond to 143 meters (approximately). The parameters used were almost similar to before, but the maximum steepness, the number of attraction points and the number of nodes influenced per attraction points were increased. It was also used a simple flow field at 85%. The initial road was also placed in the south coast of the island. Even if the maximum steepness is higher, the primary road did not develop any further, however, secondary roads developed through to almost all parts of the island. The steepness relaxation is still high enough to be restrictive when laying out new segments, as seen in the road following the low elevation area corresponding to the river in the parish of Fazenda.

The result is seen in Figure 114.

- Radius of influence: 3.0
- Segment length: 1.0
- Kill distance (first phase/second phase): 2.5/0.5
- Minimum angle: 30°
- Number of nodes influenced per attraction point in the second phase: 2
- Attraction points used per phase: 40000 (approximately)



(a)

(b)

Figure 114: a) Road map; b) Flow field used. A: Sink.; (first phase only).

- Road hierarchy thresholds: 1% and 1%
- Maximum steepness: 20°
- Flow field influence: 85%

6

CONCLUSION

A procedural road layout generation algorithm based on the concept of attraction points introduced in SCA has been presented. The algorithm extensively uses this concept to grow a network of roads.

The first goal was to extend SCA to produce a graph structure as opposed to a tree structure. This is achieved by a two phase algorithm, where in the first phase the algorithm produces what is mostly a tree like structure, followed by a second phase that performs the inter connection between the segments created in the first phase, effectively creating a graph like structure.

In order to create different types of road layouts, some parameters such as the angle constraint were introduced. Road hierarchy is obtained naturally with this proposal, since the relevance of a road can be defined based on the number of attraction points it consumes. Further enhancements were introduced in the algorithm such as the ability to use information maps to constraint and define the distribution of the attraction points in the scene. Border maps and population density maps serve this purpose. To constraint and guide the growing of the road network flow fields and elevation maps were used. Flow fields constraint the direction of new roads helping to create cleaner road layouts, whereas elevation maps limit the growth of roads above a certain steepness.

In the tests performed the parameter setting had an intuitive visual impact on the generated road layouts, which was one of the goals of this work. The examples presented show the algorithm being capable of generation different types of layouts resulting in mostly plausible road networks.

Overall this work can be seen as a first step into the potential of exploring the concept of attraction points as the basis for a procedural road layout generation algorithm.

6.1 FUTURE WORK

The weakest points of our approach are the border and height maps which require further development. The approach presented in this work has only used them at a basic level, barely scratching their full potential.

Regarding the height maps, it seems more plausible to have the new segments following along the path of least resistance (regarding height variation), unless there is a strong concentration of attraction points for a steeper, yet valid, segment. Furthermore, the algorithm should have a strategy to find paths even without attraction points. This could be achieved adding attraction points between the nodes to connect, but a more autonomous solution would be preferable.

Both border and height maps could be more integrated with the concept of flow field either by defining, or influencing it. A flow field should provide directions around forbidden areas, following their contours. For instance, considering the example for Espinho (section 5.5), the flow field would follow the coastal line defined in the border map. Regarding height maps, the flow field could also be defined based on the gradient information.

The concept of flow field per se also needs to be further explored. As is the direction of the road can either follow the flow or be perpendicular to it. This last option tends to cause undesirable road segments when the degree of curvature of the flow field is high. Using the gradient of the flow could help limiting this cases, and could eventually lead to the creation of roundabouts. New primitives could be defined to work in a similar way to templates in previous works.

As is the algorithm is mainly oriented to draw roads in populated areas, and less suitable to generate roads between these areas. Limiting, or penalising, bifurcations could be an option for further work in this context.

Finally user testing is required to evaluate the degree of flexibility and ease of use. This would allow to fine tune the set of parameters and evaluate their degree if intuitiveness.

BIBLIOGRAPHY

Christopher Alexander, Sara Ishikawa, Murray Silverstein, Joaquim Romaguera i Ramió, Max Jacobson, and Ingrid Fiksdahl-King. *A pattern language*. Oxford University Press, 1977.

Ann Collier. Fajã do conde, 2015. URL https://www.flickr.com/photos/ailognom/22317644062. [Online; accessed November 04, 2018].

Architekten von Gerkan. URL http://www.gmp-architekten.de/start.html.

Artem Pavlenko. Mapnik. URL https://mapnik.org/.

B.F. Weber GmbH. URL http://www.bfweber.de/.

Carlos Campos. *Modelação procedimental de ambientes rodoviários para simulação de condução*. PhD thesis, Faculdade de Engeharia da Universidade do Porto, 2015.

Guoning Chen, Gregory Esch, Peter Wonka, Pascal Müller, and Eugene Zhang. Interactive procedural street modeling. In *ACM SIGGRAPH 2008*, volume 27, 2008. ISBN 9781450301121. doi: 10.1145/1399504.1360702.

DigitalGlobe. Worldview-3 satellite. URL http://worldview3.digitalglobe.com/.

Arjan Egges and Jody Hausmann. *Automatic Generation of Cities in Real-time*. Citeseer, 2010.

Free Association Design. Network clouds and relational mappings, 2010. URL https://freeassociationdesign.wordpress.com/2010/04/09/network-clouds-and-relational-mappings/. [Online; accessed January 13, 2018].

E Galin, A Peytavie, N. Maréchal, and E. Guérin. Procedural generation of roads. *Computer Graphics Forum*, 29(2):429–438, 2010. ISSN 14678659.

GISgeography. Network flow roads, 2016. URL http://gisgeography.com/wp-content/uploads/2016/10/Network-Flow-Roads.png. [Online; accessed January 13, 2018].

Google LLC, a. URL https://www.google.pt/maps/place/Barcelona,+Espanha/@41.3967082,2.1840859,14.75z/data=!4m5!3m4!1s0x12a49816718e30e5:0x44b0fb3d4f47660a!8m2!3d41.3850639!4d2.1734035. [Online; accessed January 13, 2018].

Google LLC, b. URL https://www.google.pt/maps/place/Hamina,+Finl%C3%A2ndia/@60.5697729,27.1977987,17z/data=!4m5!3m4!1s0x46910fd5e845cf7d:0xb2b621cf22dff92f!8m2!3d60.5693374!4d27.1878318. [Online; accessed January 13, 2018].

Hexnet. Al-kufrah plan, 2017. URL https://hexnet.org/files/images/hexnet/libya-1.jpg. [Online; accessed January 13, 2018].

Franz Hogenberg and Georg Braun. Palmanova plan, 1593. URL http://journals.openedition.org/belgeo/docannexe/image/11877/img-13.jpg. [Online; accessed January 13, 2018].

Instituto Nacional de Estatística. Censos 2011. URL http://mapas.ine.pt/download/index2011.phtml.

Internet Engineering Task Force. The geojson format. URL https://tools.ietf.org/html/rfc7946.

George Kelly and Hugh McCabe. Citygen: An interactive system for procedural city generation. *Fifth International Conference on Game Design and Technology*, pages 8–16, 2007. doi: 10.1.1.97.4544.

Joseph B. Kruskal. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. *Proceedings of the American Mathematical Society*, 7(1):48, 1956. ISSN 00029939. doi: 10.2307/2033241. URL http://www.jstor.org/stable/2033241?origin=crossref.

Aristid Lindenmayer. Mathematical Models for Cellular Interaction in Development. *Journal of Theoretical Biology*, 18:280–315, 1968.

Manuel Lindorfer, Christian Backfrieder, Christoph Kieslich, Jens Krösche, and Gerald Ostermayer. Environmental-sensitive generation of street networks for traffic simulations. In *Proceedings - UKSim-AMSS 7th European Modelling Symposium on Computer Modelling and Simulation, EMS 2013*, pages 457–462, 2013. ISBN 978-1-4799-2578-0. doi: 10.1109/EMS.2013.77.

Craig Martek. *Procedural generation of road networks for large virtual environments*. 2012.

National Aeronautics and Space Administration. Shuttle radar topography mission. URL https://www2.jpl.nasa.gov/srtm/.

G. Nishida, I. Garcia-Dorado, and D. G. Aliaga. Example-Driven Procedural Urban Roads. *Computer Graphics Forum*, 35(6):5–17, 2016. ISSN 14678659. doi: 10.1111/cgf.12728.

Omar Cornut. Dear imgui. URL https://github.com/ocornut/imgui.

OpenStreetMap Foundation. Openstreetmap. URL https://www.openstreetmap.org/.

Orange Smile. Moscow plan, 2017. URL http://www.orangesmile.com/destinations/img/moscow-map-big.gif. [Online; accessed January 13, 2018].

Yoav I H Parish and Pascal Müller. Procedural modeling of cities. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01*, SIGGRAPH '01, pages 301–308, New York, NY, USA, 2001. ACM. ISBN 158113374X. doi: 10.1145/383259.383292.

Ken Perlin. An image synthesizer. *ACM SIGGRAPH Computer Graphics*, 19(3):287–296, jul 1985. ISSN 00978930. doi: 10.1145/325165.325247.

Przemyslaw Prusinkiewicz and Aristid Lindenmayer. Graphical modeling using L-systems. In *The Algorithmic Beauty of Plants*, pages 1–50. Springer New York, 1990. doi: 10.1007/978-1-4613-8476-2{_}1.

QGIS. Qgis. URL https://qgis.org/.

Rui Pedro Vieira. Fajãzinha, 2012. URL https://www.flickr.com/photos/ruipedrovieira/23404940050. [Online; accessed November 04, 2018].

Adam Runions, Martin Fuhrer, Brendan Lane, Pavol Federl, Anne-Gaëlle Rolland-Lagan, and Przemyslaw Prusinkiewicz. Modeling and visualization of leaf venation patterns. *ACM Transactions on Graphics*, 24(3):702, 2005. ISSN 07300301. doi: 10.1145/1073204.1073251.

Adam Runions, Brendan Lane, and Przemyslaw Prusinkiewicz. Modeling trees with a space colonization algorithm. In *Natural Phenomena*, NPH'07, pages 63–70, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association. ISBN 9783905673494. doi: 10.2312/NPH/NPH07/063-070.

Snazzy Maps. URL https://snazzymaps.com/style/55/subtle-greyscale-map. [Online; accessed January 13, 2018].

Dan Stanislawski. The origin and spread of the grid-pattern town. *Geographical Review*, 36(1): 105–120, 1946. ISSN 00167428. URL http://www.jstor.org/stable/211076.

Jing Sun, Xiaobo Yu, George Baciu, and Mark Green. Template-based generation of road networks for virtual city modeling. *Proceedings of the ACM symposium on Virtual reality software and technology - VRST '02*, page 33, 2002. doi: 10.1145/585746.585747.

Edward Teng and Rafael Bidarra. A semantic approach to patch-based procedural generation of urban road networks. In *Proceedings of PCG 2017 - Workshop on Procedural Content Generation for Games, co-located with the Twelfth International Conference on the Foundations of Digital Games*, 2017. URL http://graphics.tudelft.nl/Publications-new/2017/TB17.

Soon Tee Teoh and Teoh Soon Tee. Algorithms for the automatic generation of urban streets and buildings. *Proceedings of the 2008 International Conference on Computer Graphics and Virtual Reality (CGVR 2008)*, 1:122–128, 2008.

ST Teoh. Autopolis: Allowing user influence in the automatic creation of realistic cities. *Advances in Visual Computing*, pages 118–129, 2007. ISSN 03029743.

The Khronos Group, Inc. Open graphics library. URL https://www.opengl.org/.

Basil Weber, Pascal Müller, Peter Wonka, and Markus Gross. Interactive geometric simulation of 4D cities. *Computer Graphics Forum*, 28(2):481–492, 2009. ISSN 14678659.

Jason Weber and Joseph Penn. Creation and rendering of realistic trees. *SIGGRAPH '95 - Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, pages 119–128, 1995. ISSN 0097-8930. doi: 10.1145/218380.218427.

Jakub Wejchert and David Haumann. Animation aerodynamics. *SIGGRAPH Comput. Graph.*, 25(4):19–22, July 1991. ISSN 0097-8930. doi: 10.1145/127719.122719. URL http://doi.acm.org/10.1145/127719.122719.

Yong-Liang Yang, Jun Wang, Etienne Vouga, and Peter Wonka. Urban pattern: Layout design by hierarchical domain splitting. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2013)*, 32:Article No. xx, 2013.

A

PAPER PUBLISHED IN ICGI 2018

The award for best paper at the 1st International Conference on Graphics and Interaction that took place between 15 and 16 November 2018 at the Faculty of Sciences of the University of Lisbon, was awarded to the researchers Gabriel Fernandes and António Ramires Fernandes that authored the paper entitled ”Space Colonisation for Procedural Road Generation”.

Space Colonisation for Procedural Road Generation

Gabriel Dias Fernandes
Departamento de Informática
Universidade do Minho, Portugal
a71492@alunos.uminho.pt

António Ramires Fernandes
Centro Algoritmi
Universidade do Minho, Portugal
arf@di.uminho.pt

***Abstract*—Procedural road layout generation has been traditionally approached using L-Systems, with some works exploring alternative avenues. Although originally conceived for biological systems modelling, the adequacy of L-Systems as a base for road generation has been demonstrated in several works.**

In this context, we present an alternative approach for procedural road layout generation that is also inspired by plant generation algorithms: space colonisation. In particular, we use the concept of attraction points introduced in space colonisation as the base of our approach to produce road layouts, both in urban and inter-city environments. As will be shown, the usage of attraction points provides an intuitive way to parameterise a road layout. Our method can also be used with flow-fields, demographics, geographical, and boundary maps to further fine tune the layout.

***Keywords*—procedural road generation, space colonisation, L-systems**

I. Introduction

The procedural generation of content arises from the human curiosity and need to describe natural phenomena with maths. Fibonacci sequences and the golden ratio are well known nowadays by being present in grow patterns in nature.

In 1968, Aristid Lindenmayer [1], a biologist, introduced L-Systems to describe cellular algae growth. Later, Lindenmayer and Prusinkiewicz ([2]) used L-systems to generate fractals, and other geometrical patterns, as well as 3D bushes and trees. The procedural generation of trees and bushes, using L-Systems, was the focus of [3] which achieved very realistic results. Given the flexibility of L-Systems, they started to be widely used in several topics among procedural generation.

In procedural generation of roads, L-Systems were also the starting point of the early generation algorithms, with a significant amount of the subsequent works being based on this approach.

In this work, an alternative approach to procedural street layout generation is presented, combining different ideas from previous works but using a different base algorithm.

The algorithm used as the cornerstone of this work is the *Space Colonisation Algorithm* (SCA). Presented in [4] by Runions et. al., SCA was designed to procedurally generate trees. Our motivation is explore the concept of attraction points introduced in SCA to direct the growing of roads.

We opted for SCA because it is a more intuitive and predictable algorithm than L-systems. With SCA it is feasible to imagine the final output based on the input parameters, something which is hard when considering the axioms and rules for an L-system.

By design SCA returns a tree like structure, which we need to transform into a graph like structure to be able to simulate a road layout. This is our main contribution: to extend SCA for road layout generation.

The remainder of this document is organised as follows: section II provides an overview of previous related work; in section III we present the algorithm in which we based our approach; section IV details our extension to SCA so that it is suitable for road layout generation; section V presents some results obtained with more complex settings; section VI presents our conclusions and suggests some avenues for further research.

II. Related work

Concerning the procedural generation of road networks, Parish and Müller [5] produced one of the earliest works. They presented a system called *CityEngine*, capable of generating a city road network based on L-Systems and maps as inputs. These maps contain information regarding population density and land-water boundaries. L-systems were extended to be able to consider local and global constraints. Furthermore, they present a system to procedurally generate buildings and the appropriate textures, making it a very complete system.

Sun et. al. [6] focus on highways, and use road templates to produce radial, population-based, raster configurations. These templates can be used together to produce a mixed layout. The authors also use land/water/vegetation and elevation maps.

Citygen [7] is an interactive engine, fully configurable by the user. The user defines the primary roads by creating a graph, which is superimposed in an elevation map. Roads are generated based on several algorithm variations such as a minimum elevation strategy. Inside a city, cells are found by considering enclosed regions of the primary graph. Then a variant of L-systems is used to fill the inside of those cells.

Autopolis, a system presented in [8] and [9] by Teoh, starts by asking the user to define a set of initial parameters such as the number of commercial and industrial centres, airports, among other items. The system then uses a set of heuristic to place these items and grows a network of roads to include residential centres starting with highways to connect the centers, followed by main and secondary roads. Highways are created using the shortest path weighted with cost. Vector

978-1-5386-2080-9/17/$31.00 ©2018 IEEE

fields are created based on either elevation, coastlines, or existing roads, for the main and secondary roads.

In [10], by Chen et. al., the generation and modification of the road network is based on interactively created tensor fields. The roads will be created following the directions described by the tensor field. To achieve more organic results, Perlin noise [11] is used to provide some controlled randomness.

Webber et. al. in [12] propose a 4D simulation, including time to simulate the growth of cities. Regarding street generation their concern is that the street layout must make sense at any time t. Therefore, their strategy is based on expansion of the previous street layout in a two stage process: expanding major streets and filling quarters with minor streets. To determine which streets to grow at any particular time a probabilistic approach is used based on the distance to the nearest growth centre. Direction, length, and other street parameters are chosen based on the street pattern which can be radial, organic or grid based.

Aiming towards the generation of a single realistic road, Galin et. al. [13] proposed a flexible algorithm to deal with obstacles such as forests, rivers and mountains. By specifying a starting and ending point of the road to be generated, the goal is to find the shortest path, weighted by cost, between those points. The algorithm relies on the existence of a cost function that can be evaluated at an point on the map to determine the best solution. For instance the cost function may dictate that is cheaper to build a bridge than to contour the water areas, or build tunnels, or bridges to deal with steep elevation changes. Later, in [14], this work was extended to generate hierarchical road networks.

Lindorfer et. al. [15] method also starts by creating highways between highly populated centres. The regions delimited by the highways are then partitioned to create local transport roads. A modified environment sensitive L-system algorithm is used to create the street maps. As in Sun et. al [6] they also use patterns, namely the raster and radial pattern.

Yang et. al. [16] propose an interactive system where the user starts by specifying main roads. The map is then divided in cells limited by these roads. For each cell the user can then specify a template to fill the cell.

Campos [17] based his method in the procedures used in road engineering, producing road paths definitions according to design standards, current practices in road design, thus producing roads similar to those found in the real world.

Nishida et. al. [18] approach is based on the extraction of patches from real road maps, where a patch is defined as being a meaningful road structure. An interactive system was built in which the user selects areas from real-road maps from which patches are extracted and used in the new road layout. Later, Teng and Bidarra [19], extended the core patch idea to include semantics, and allow a higher level editing based on these semantics.

In conclusion, there is some diversity on the approaches taken, although a very significant number uses L-systems and/or extensions. The earliest works explore L-Systems, or variants, as the main procedural algorithm to generate both the roads and buildings in cities. Other algorithms include the shortest path algorithm, minimum spanning trees, templates, and even tensor fields. More recent works are based on pattern extraction from real world road layouts.

III. Space colonisation for tree growth

The essence of the SCA is the notion of attraction points, which promote branch growth and bifurcation. This led us to consider the application of attraction points to procedural road layout generation. In this section we will provide a short introduction to the algorithm in which our work is based.

A. Space colonisation algorithm

SCA is presented by Runions et. al. in [4] and in summary consists in a tree generation algorithm that promotes space colonisation.

A tree is composed by *tree nodes* connected between them. The tree growth will be influenced by a set of *attraction points* that will dictate how the tree will grow and develop.

Initially, attraction points are scattered on the desired volume, together with an initial tree node or nodes. The tree will grow starting from the existing tree nodes, towards attraction points that influence them. An attraction point can only influence the closest tree node, being required that the distance between the two is less than a given distance called *radius of influence*.

Although an attraction point can only influence one tree node, a tree node can be influenced by many attraction points.

For any tree node under the influence of attraction points, a new node is created along the direction given by the average of all normalised directions between the current tree node and all attraction points that influence it. The new tree node will be placed at a distance D, the segment length, from the predecessor tree node.

All attraction points used to influence a new node creation that are at a distance from the node which is less than a predefined threshold, called the *kill distance*, are eliminated.

All these steps are repeated until all attraction points are consumed, the tree is no longer attracted by any points, or the user decides to stop the algorithm.

Figure 1 presents the algorithm graphically.

B. Leaf venation patterns

Prior to the space colonisation algorithm, Runions et. al. published "*Modeling and visualisation of leaf venation patterns*" ([20]). The leaf venation patterns can be distinguished by two types: open and closed patterns, as seen in Figure 2. The open venation pattern consists on a vein network tree like structure, where existing vein nodes do not connect to each other, except when a new node is generated. It will result in a more or less sparse network that it is not intra-connected. The algorithm that generates the open venation pattern, is essentially the space colonisation algorithm and is the method that gave rise to the latter work [4]. As for the closed venation pattern, this is a more usual pattern found in nature. A dense vein graph network may have a ladder-like pattern (*percurrent*) or a cracked/netlike pattern (*reticulate*).

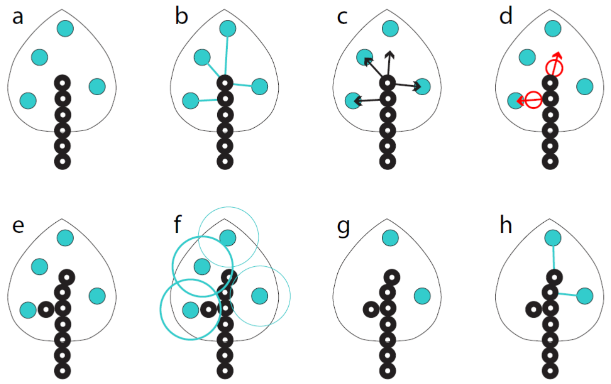


Figure 1: Space colonisation steps. Black dots - Tree nodes; Blue dots - Attraction points; Blue lines - Influence link; Black arrows - Normalised directions; Red dots - New tree nodes; Blue radius - Kill distance. Source: [4]

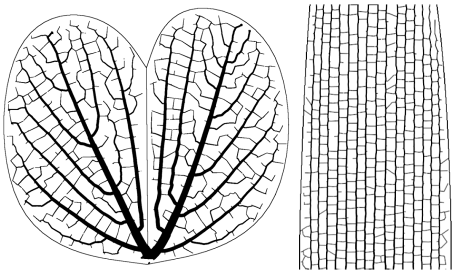


Figure 2: Left: Reticulate pattern (Orchid leaf); Right: Percurrent pattern (Grass leaf). Source: [20]

Getting into further detail about the closed venation pattern, it is based on the open venation pattern but it also allows that more than one vein may grow towards the same source. This feature is to promote the formation of anastomoses, the connection of two different veins. To be similar to nature, anastomoses should be done when the two or more veins are close enough to each other, yet they are relatively far from each other. To do so, the algorithm will use the notion of relative neighbourhood. Two points, s and v, are relative neighbours of each other if for every any other point u from the set of all points that are closer to s than v, v is closer to s than to u.

In Figure 3, we may see that v, a and b are relative neighbours of s, and we can confirm it by checking their relative distances shown by lines. The green area admits the points that are both closer to s than v, and that v is closer to them than to s, in other words, that area must be empty otherwise v would be promptly discarded as neighbour of s as said before. The blue area consists in every point that is closer to v than to s, which are promptly discarded as well.

As already seen in Figure 2, this algorithm produces extremely realistic results in terms of generating a venation network that not only tries to mimic botanical elements visually but also how do they work biologically (auxin sources, anastomoses).

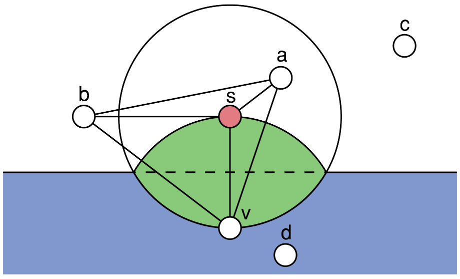


Figure 3: Relative neighbourhood example. Source: [20]

This could have been the perfect algorithm to solve the problem of inter-connecting the main roads of a network, however, the relative neighbourhood method which is the main reason why the closed venation pattern is so realistic and optimal, is not flexible enough in terms of controlling its output in a road layout context.

Hence, we chose to use SCA as the base for our approach, and to further parameterise and customise to suit our needs of generating a road topology.

IV. Space colonisation in the context of road layout generation

This section covers our proposed extension to SCA. As in SCA [4] we use attraction points to shape the final network. The start of the process is to create a cloud of attraction points (in 2D). This cloud can be influenced by information maps (IV-C1). Attraction points will be used for road segment growth, much like in the original algorithm. We also need to specify a set of initial segments.

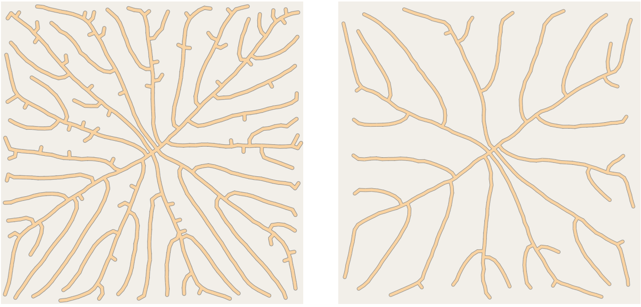
Similarly to many of the previous road generation methods, our approach is a two phase algorithm. The first phase creates a 2D network of roads in what is mostly a tree like structure (IV-A). The second phase further produces the connections between nodes created in the first phase (IV-B), effectively creating the required graph structure. Additional configuration features, namely a flow field and information maps are discussed next (IV-C).

A. First phase

The first phase of the algorithm aims to generate what is mostly a tree shaped road network. When the road network cannot grow further, due to consuming all the attraction points or by not being influenced by them anymore, we have two options: we add more attraction points and continue growing the initial road network, or we proceed to the second phase.

In here we explore some of the possible parameterisation for the first phase, starting with the original SCA's parameters, followed by our extensions.

1) Capillarity: Different topologies require different settings. For instance, the main roads of a country or a network that represents a city map differ in terms of density, or *capillarity*. Working towards these layouts can be achieved



(a) Kill radius = 2 (b) Kill radius = 4

Figure 4: Capillarity influence result.

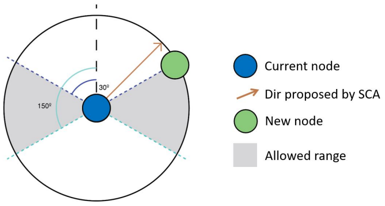
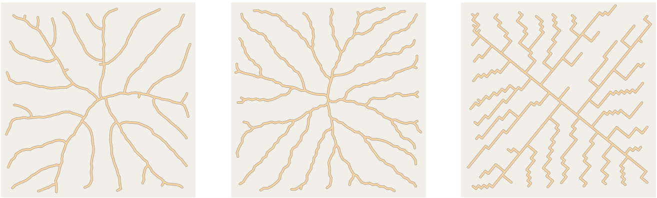


Figure 5: Angle control example.



(a) No restrictions (b) range [30°, 45°] (c) Angle set to 90°

Figure 6: Angle restriction effect on the road network.

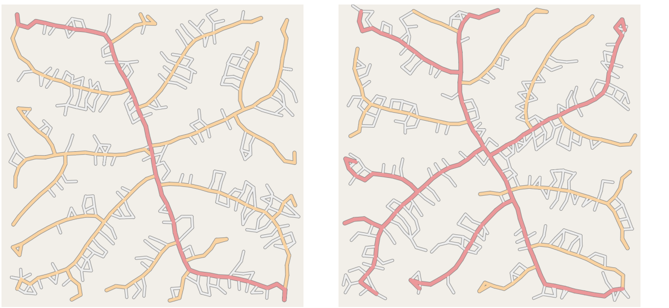


Figure 7: Automatic road hierarchy: red primary ; yellow secondary; grey:tertiary roads.

with SCA original parameters: *radius of influence* and *kill radius*. A network can have more or less capillaries depending on the ratio between the *radius of influence* and *kill radius* parameters. As discussed in [4] the *radius of influence* should be larger than the *kill radius*. The control of this feature is very intuitive, for instance, a dense network is generated by having a large ratio between the *radius of influence* and the *kill radius*, otherwise, we have a sparse network. In other words, with a low *kill radius* an attraction point will tend to live longer than with a higher radius, causing the growth of more road segments.

Two full-grown road network results based only in capillarity are presented in Figure 4. These were both generated using the same uniform distribution of attraction points and *radius of influence* set to 5, with the first node placed in the middle of the grid and with different *kill radius*.

2) Angle constraints: To control the appearance of the initial road layout we extended SCA to include parameters to control the angle between segments of the road. These parameters impose a maximum and minimum angle for newly created nodes and segments. Proposed directions outside the defined range are snapped to the valid range as shown in Fig. 5. With this feature we can create a rigid grid-like city map, or a more organic look as shown in Figure 6. As an extra option we also allow the road to grow in a straight direction, even when setting angle constraints. This effect can be seen most pronounced in Figure 5 c).

3) Road hierarchy: In real-world road networks, there is a hierarchy that classifies roads by their relevance. An automated procedure creates this hierarchy, classifying roads based on the number of attraction points consumed by their segments. We can cope with any number of hierarchical levels based on user defined thresholds. We can further restrict road hierarchical classification per phase, for instance, considering roads created in the first phase as being either primary or secondary, and relegate all roads created in the second phase to being tertiary roads, reducing the number of threshold parameters to one. Using attraction point count per segment as the base for defining a hierarchical level is an intuitive approach that is directly related to the density of attraction points in specific areas. As seen in IV-C1, the distribution of attraction points can be made to match population density providing more important roads in higher population density areas.

The road hierarchy algorithm starts by considering the initial segments defined by the user as primary roads. Road classification is performed on each new segment as follows: when the new segment implies a bifurcation, we evaluate the road relevance for the new segment depending on the number of attraction points that influenced the creation of the segment; otherwise, when the segment is just extending the previous road, we set its relevance equal to the previous segment. We also found it convenient to ensure that the relevance of a new segment can never be set higher than the previous segment. Figure 7 shows two examples of the automatic road classification in the first phase. With this approach we obtain long primary roads, and shorter roads with less relevance.

4) Snapping and Merging: Mainly due to the angle limitation feature it is possible that newly created nodes are too close, or even superimposed, to other nodes or segments as is the case in Fig. 6 c).

If the proposed node is too close to any other road segment or node, by a given distance, called *snapping distance*, one of two things will happen: if the nearest part of the road network to the proposed node is a previously existing node, the proposed node will be merged with this node, if the nearest

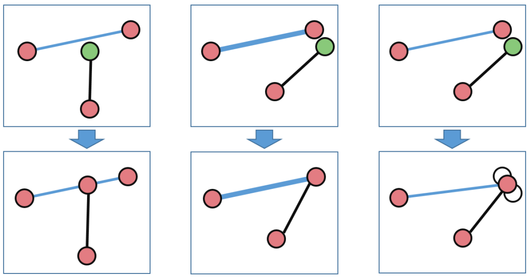


Figure 8: Snapping modes. Left: snap to segment; Middle: snap to higher relevance node; Right: snap both nodes to average position.

part is in the midst of a road segment, it will snap to it creating a new node in that position.

Node merging takes in to account the road hierarchy. When merging two nodes belonging to roads with the same relevance, the new location is the average location of both nodes. Otherwise, the new position will be the position of the node of the most relevant road to maintain the most relevant road direction.

Figure 8 illustrates this procedure, where road thickness relates to road relevance. This is similar to the local constraints defined in [5].

B. *Second phase*

In general, the first phase tends to produce a road network shaped mostly like a tree. To get an inter-connected road graph network we run a second phase. In here we detail our approach to achieve this.

1) Road inter-connection: The first step in this phase is to directly connect nodes created in the first phase. These connections will take into account two different aspects: the nodes to be connected should be close enough to each other, and the suggested connection should make an angle with the previous node connections that should be as close as possible to the *ideal angle*. The *ideal angle*, is the angle that allows the new segment to be the most distant to any of the other existing segments of the node, to promote a distributed network.

A node can have multiple ideal angles based on different scenarios depending on the cardinality of the node connections. When there is none, the ideal angles are perpendicular or parallel to the previous connection with the node. When there is at least one descendent from the node, the ideal angles are the average between the subsequent directions of the connections from and to this node.

Figure 9 presents two examples on how the connecting mechanism works. Nodes A and B are both inside the valid area, in which case the connection will be from the current node to B because it is closer to the ideal angles.

When applying this procedure as described we noted that there was a tendency to form consecutive triangular connections, something that is not observable in real roads. Therefore, we added a restriction to prevent the formation of triangular connections in this step.

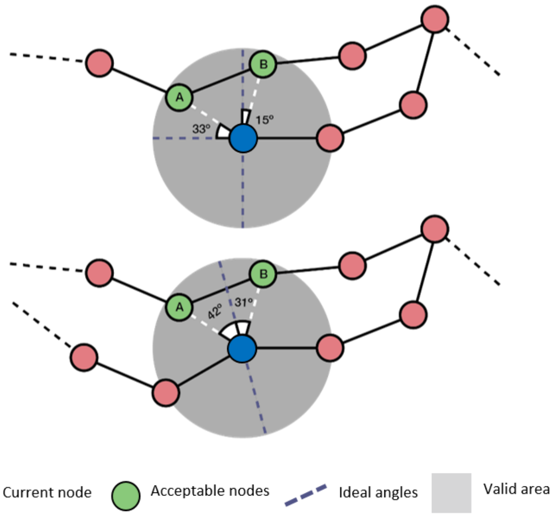
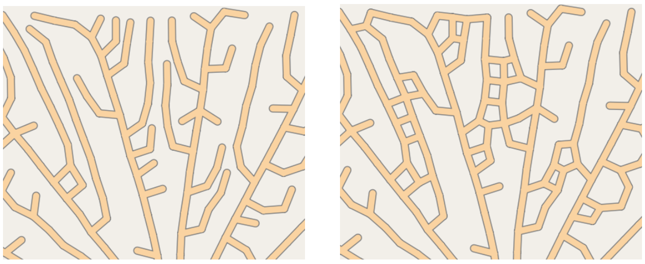


Figure 9: Node connection mechanism.



(a) Before

(b) After

Figure 10: Road inter-connection impact on the network.

An example of the application of this step is presented in Figure 10, showing both the network at the end of the first phase and the end result of this step.

2) Attraction point validation: In the general case, to continue to grow the road network in the second phase we need to add more attraction points. Since there is already a formed road network, this attraction point distribution cannot be completely random, otherwise new attraction points may cause the network to grow roads that can cross the original network, or create road segments too close to existing ones.

To prevent this we've added a parameter to control the minimum required distance to accept a new attraction point. Attraction points inside this safety margin will be removed, see Figure 11.

3) Some notes regarding attractors: In here we present some extensions we made to SCA in order to inter-connect the roads from the first phase to make the attraction points more influential.

To achieve the desired inter-connection between the first phase nodes, these nodes will also be considered as attractors. Furthermore, attraction points will attract the N closer nodes instead of just the closest node, unlike the original SCA (we found that $N = 2$ provides more pleasing results).

To guarantee that the road network will successfully connect

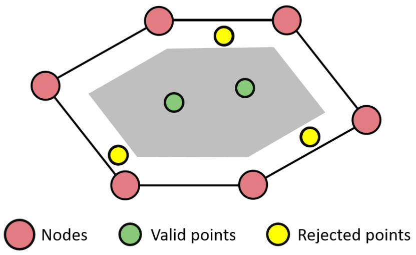


Figure 11: Attraction point validation example.

itself we must extend the *lifespan* of the attraction points relative to the first phase. This can be achieved setting a lower kill radius in this phase. By default we set it at half the value from the first phase.

C. Additional features

In addition to the features described before, it is possible to complement the results with external information. This information is given through maps representing information about population density, type of terrain and elevation. These maps will condition and change how the network will grow.

As an alternative to setting angles to condition direction, or in combination with them, it is also possible to use a flow-field to impose restrictions on the direction for new segments.

1) Information maps: A population density map is a grey level map where lighter areas imply higher population density. These maps will condition the placement of attraction points. When attraction points are placed they are accepted/rejected based on a probability taken from the intensity.

The border map has information about illegal areas to grow roads, such as rivers, lakes, sea, city and natural parks. This will dictate which areas are available to support the road network. It can be extended to differentiate areas so that the algorithm can behave differently when generating the road network according to those areas. Attraction points placed in illegal areas will not be considered for road generation, and road segments have to be checked to ensure they do not cross these illegal areas.

The height map contains the height of each point in the grid/terrain. Darker tones are associated with low elevations, whereas lighter tones correspond to higher altitudes. These maps are used to determine the steepness of the terrain in order to decide if a road can grow in a particular direction based on a parameter *maximum steepness*. An example is presented in sec. V-B.

2) Flow field: Many scientific areas use vector fields to represent some models and phenomena such as force, wind movement, gravitational force, *et cetera*. In our context, we call them *flow fields* since they influence the direction of new segments.

To build flow fields we use flow primitives, based on the work by Wejchert and Haumann [21]. The primitives will shape the field, influencing the direction of the vectors in the grid, corresponding to the building blocks of the flow field: *uniform* which causes a flow in straight lines, parallel to the direction specified by the primitive; *source*, which creates a flow away from the location of the primitive; *sink* which flows towards the primitive; and *vortex* which causes a circular flow around the primitive. Each primitive is described by a linear equation, and the cumulative result at each grid point can be computed as the sum of the influences for all primitives. Under our approach, a flow field is defined by placing these primitives. Figure 12 shows a flow field created with three primitives, and Figure 13 shows an example of a road layout created with the flow field.

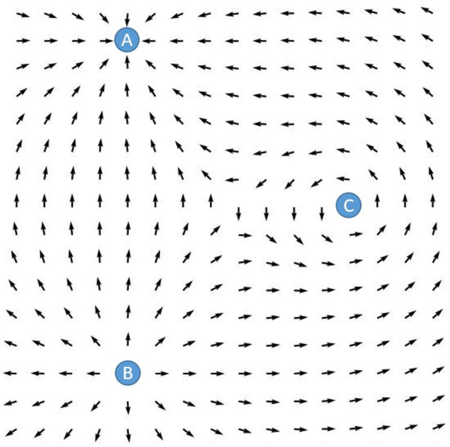


Figure 12: Flow field with primitives: A is a sink; B a source, and C a vortex.

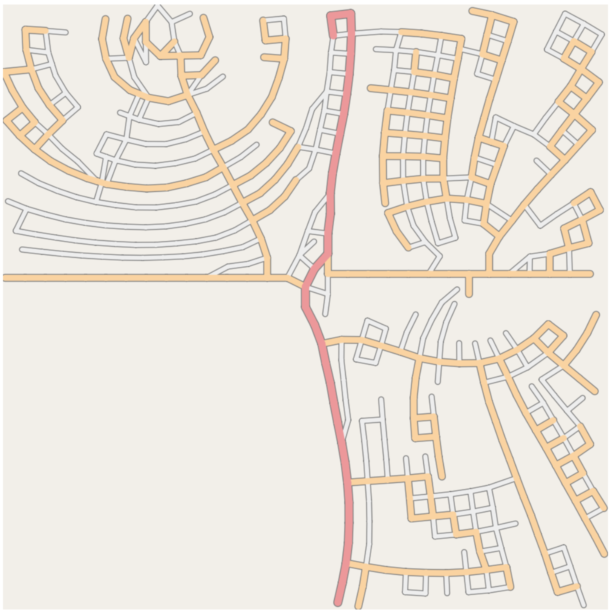


Figure 13: Road layout for flow field from Figure 12.

When using a flow field the direction of new segments is limited to either follow the flow or perpendicular to it.

The flow field can be used as a replacement of the angle restriction parameters, or in a a combination of influence for

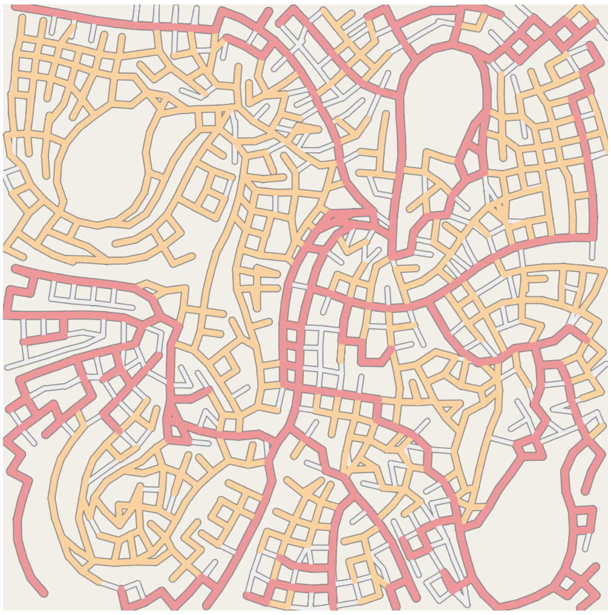


Figure 14: Town with parks.

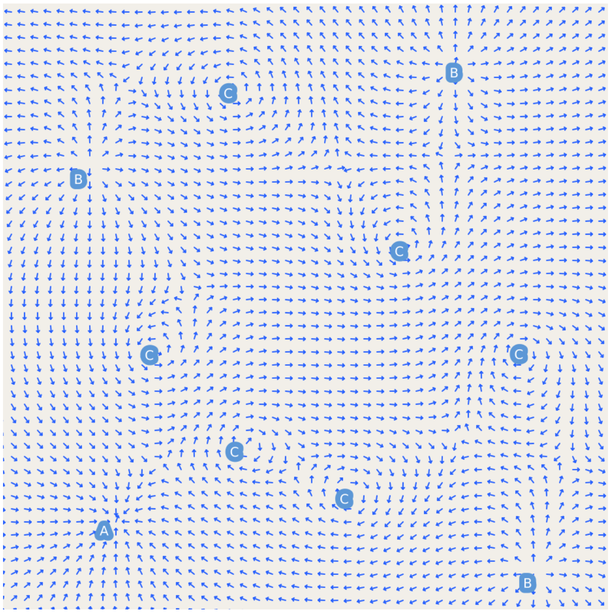


Figure 15: Flow field for road layout in Figure 14

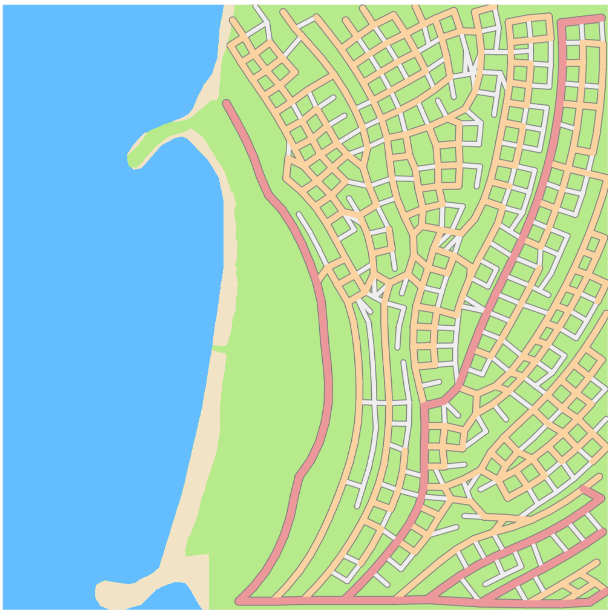


Figure 16: Coastal area

the direction of the new road segments.

V. Additional Results

A. *Flow fields*

In here we present more complex road layouts based on flow fields. Figure 14 shows a layout generated with the flow field presented in Figure 15. The sources (label B) in the flow field tend to create empty spaces and vortices (label C) create curved roads. These primitives create patterns that can simulate various layout types. For instance, Figure 14 resembles a town with large parks in the empty areas, where as Figure 16 could be a more modern, grid like, coastal area.

An example where the initial segments are placed as a ring road around the relevant area and also using a flowfield can be seen in Figure17. Attraction points were placed mainly inside the area delimited by the initial ring road.

B. *Dealing with elevation data*

The algorithm can also deal with height taken from elevation maps based on a parameter that controls the *maximum steepness* of generated roads. Figure 18 shows elevation information in grey scale. From the elevation and a border map of the island, we tested the growth of the road network limited by the steepness of the terrain setting *maximum steepness* to $10°$. The solution found covers the whole island, still there are some connections missing. This leads to the conclusion that an extra step based on path finding might be required to perform these connections.

VI. Conclusion

We based our method on the concept of attraction points from SCA. An extension to SCA was presented as a two phase algorithm, both phases using a modified SCA algorithm. The first phase grows mostly a tree like structure, and the second performs inter road connectivity, effectively transforming the initial tree structure into a road layout graph. The parameterisation of the algorithm to suit different road layouts was presented and exemplified. We find the parameter setting to be intuitive as there is a direct connection between the parameter values and the geometry of the road layout. For instance, road hierarchy is directly related to the density of the distribution of attraction points, and the kill radius also provides an intuitive way of controlling the capillarity of the roads. We also showed how our proposal could use a flow field and information maps to further tune the road layout.

Regarding future work we believe that it would be inter-

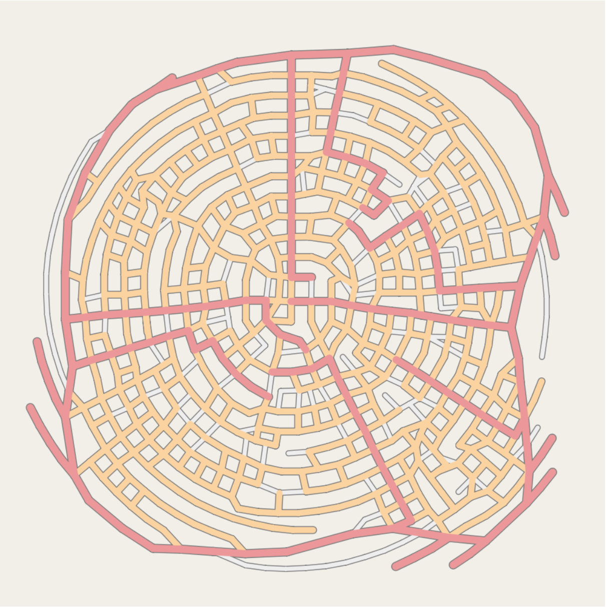


Figure 17: Defining initial segments as a ring road.

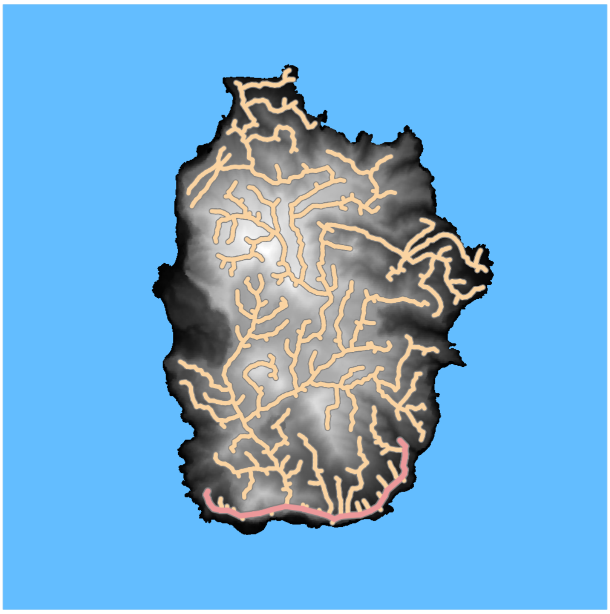


Figure 18: Road layout with elevation map.

esting to further parameterise the distribution of the attraction points taking into consideration points of interest, such as, for instance, shopping malls, train and bus stations. Allowing a radius of influence for the primitives in a flow fields, and exploring different primitives might also provide interesting road layout patterns.

Acknowledgments

This work has been supported by COMPETE: POCI-01-0145-FEDER-007043 and FCT – Fundação para a Ciência e Tecnologia within the Project Scope: UID/CEC/00319/2013.

References

[1] A. Lindenmayer, "Mathematical Models for Cellular Interaction in Development," *Journal of Theoretical Biology*, vol. 18, pp. 280–315, 1968.

[2] P. Prusinkiewicz and A. Lindenmayer, "Graphical modeling using L-systems," in *The Algorithmic Beauty of Plants*. Springer New York, 1990, pp. 1–50.

[3] J. Weber and J. Penn, "Creation and rendering of realistic trees," *SIGGRAPH '95 - Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, pp. 119–128, 1995.

[4] A. Runions, B. Lane, and P. Prusinkiewicz, "Modeling trees with a space colonization algorithm," in *Natural Phenomena*, ser. NPH'07. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2007, pp. 63–70.

[5] Y. I. H. Parish and P. Müller, "Procedural modeling of cities," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01*, ser. SIGGRAPH '01. New York, NY, USA: ACM, 2001, pp. 301–308.

[6] J. Sun, X. Yu, G. Baciu, and M. Green, "Template-based generation of road networks for virtual city modeling," *Proceedings of the ACM symposium on Virtual reality software and technology - VRST '02*, p. 33, 2002.

[7] G. Kelly and H. McCabe, "Citygen: An interactive system for procedural city generation," *Fifth International Conference on Game Design and Technology*, pp. 8–16, 2007.

[8] S. Teoh, "Autopolis: Allowing user influence in the automatic creation of realistic cities," *Advances in Visual Computing*, pp. 118–129, 2007.

[9] S. T. Teoh and T. Soon Tee, "Algorithms for the automatic generation of urban streets and buildings," *Proceedings of the 2008 International Conference on Computer Graphics and Virtual Reality (CGVR 2008)*, vol. 1, pp. 122–128, 2008.

[10] G. Chen, G. Esch, P. Wonka, P. Müller, and E. Zhang, "Interactive procedural street modeling," in *ACM SIGGRAPH 2008*, vol. 27, 2008.

[11] K. Perlin, "An image synthesizer," *ACM SIGGRAPH Computer Graphics*, vol. 19, no. 3, pp. 287–296, jul 1985.

[12] B. Weber, P. Müller, P. Wonka, and M. Gross, "Interactive geometric simulation of 4D cities," *Computer Graphics Forum*, vol. 28, no. 2, pp. 481–492, 2009.

[13] E. Galin, A. Peytavie, N. Maréchal, and E. Guérin, "Procedural generation of roads," *Computer Graphics Forum*, vol. 29, no. 2, pp. 429–438, 2010.

[14] E. Galin, A. Peytavie, E. Guérin, and B. Beneš, "Authoring hierarchical road networks," *Computer Graphics Forum*, vol. 30, no. 7, pp. 2021–2030, 2011.

[15] M. Lindorfer, C. Backfrieder, C. Kieslich, J. Krösche, and G. Ostermayer, "Environmental-sensitive generation of street networks for traffic simulations," in *Proceedings - UKSim-AMSS 7th European Modelling Symposium on Computer Modelling and Simulation, EMS 2013*, 2013, pp. 457–462.

[16] Y.-L. Yang, J. Wang, E. Vouga, and P. Wonka, "Urban pattern: Layout design by hierarchical domain splitting," *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2013)*, vol. 32, p. Article No. xx, 2013.

[17] C. Campos, "Modelação procedimental de ambientes rodoviários para simulação de condução," Ph.D. dissertation, Faculdade de Engeharia da Universidade do Porto, 2015.

[18] G. Nishida, I. Garcia-Dorado, and D. G. Aliaga, "Example-Driven Procedural Urban Roads," *Computer Graphics Forum*, vol. 35, no. 6, pp. 5–17, 2016.

[19] E. Teng and R. Bidarra, "A semantic approach to patch-based procedural generation of urban road networks," in *Proceedings of PCG 2017 - Workshop on Procedural Content Generation for Games, co-located with the Twelfth International Conference on the Foundations of Digital Games*, 2017. [Online]. Available: http://graphics.tudelft.nl/Publications-new/2017/TB17

[20] A. Runions, M. Fuhrer, B. Lane, P. Federl, A.-G. Rolland-Lagan, and P. Prusinkiewicz, "Modeling and visualization of leaf venation patterns," *ACM Transactions on Graphics*, vol. 24, no. 3, p. 702, 2005.

[21] J. Wejchert and D. Haumann, "Animation aerodynamics," *SIGGRAPH Comput. Graph.*, vol. 25, no. 4, pp. 19–22, Jul. 1991. [Online]. Available: http://doi.acm.org/10.1145/127719.122719