Automatic generation of plant distributions for existing and future natural environments using spatial data

Master of Science Thesis

Benny Onrust

MSc Geomatics

Delft University of Technology

Challenge the future

Automatic generation of plant distributions for existing and future natural environments using spatial data

Master of Science Thesis

by

Benny Onrust

in partial fulfillment of the requirements for the degree of

Master of Science
in Geomatics

at the Delft University of Technology,
to be defended publicly on Friday January 30, 2015 at 09:00 AM.

Student name:	Benny Onrust
Student number:	4019512
Email:	b.onrust@student.tudelft.nl
Supervisor:	Drs. C.W. Quak
Graduation professor:	Ass. Prof. Dr. S. Zlatanova
Co-reader:	DI A. Wandl
Delegate of the board of examiners:	Drs. D.J. Dubbeling

Abstract

This research proposes an algorithm for the generation of realistic plant distribution for both existing and future areas using spatial data. The main motivation for this research is to be able to generate realistic plant positions for the 3D visualization of existing and future natural environments. Current techniques for determining plant positions are limited. Plant detection techniques from the remote sensing domain are only able to detect positions for large plants using high-resolution imagery and LiDAR data. Often, the plant type of each detected position is not known. In addition, imagery and LiDAR data are not available for future areas. This research demonstrates that spatial data for future can be generated by using dynamic ecological models. These models can produce height, biomass, and coverage maps for future areas. The proposed algorithm must be able to translate data from existing areas as well as data produces by ecological models to a realistic plant distribution. A realistic plant distribution can contain plant positions for small and large plants with their corresponding plant type. To be able to realize this, an algorithm is proposed that integrates concepts of procedural generation and ecological modeling. Different spatial datasets can be provided as input and are analyzed to obtain information of where certain plant types can or cannot be placed in the target area. The algorithm was tested on both an existing area and a future area generated by an ecological model. The results were validated with statistical and expert validation methods. The statistical validation showed that algorithm is able to map the spatial data input correctly to a plant distribution. The expert validation showed that the generated plant distributions were in general judged realistic and that most issues in each plant distribution were because of missing data input or due to low data quality. In the future, the algorithm could be used for different applications and research. One possibility is to improve the current plant detection technique by providing additional information about the positions of small plants and by providing information about the plant type for each point with this algorithm

CONTENTS

Abbreviations

CA:	Cellular Automata
FoN:	Field of Neighborhood
LCC:	Land Cover Classification
LiDAR:	Light Detection And Ranging
MRC:	Modified Random Clusters
NDVI:	Normalized Difference Vegetation Index
PCG:	Procedural Content Generation
PDD:	Poisson Disk Distribution

1
Introduction

1.1. Motivation

There is much interest in 3D visualization of current and future natural environments [1], because it provides better cognitive understanding of spatial relations (topology) and vertical dimension (geometry) of the environment [2]. This means that a 3D visualization improves the understanding of how a certain natural environment looks like. Therefore, 3D visualization are being used more often in the field of decision-making, and recreational or scientific communication [3][4][2]. In the Netherlands, an example of where a 3D visualization could improve the decision-making and communication process is the Hedwigepolder case. The Hedwigepolder is an area in the Netherlands, which is going to be partly flooded. The location of the Hedwigepolder is shown infigure 1.1 with the marker A in the red area. There is a lot of resistance against flooding the Hedwigepolder by the locals, because they think that the flooded area will be completely lost to the water. According to ecologists, however, that area will partly become a salt marsh, which is a type of natural environment. In the blue area of figure 1.1 a top-down view of a salt marsh is shown. The marsh that arises in the Hedwigepolder area will be connected to the marsh in the blue area [5]. The general public and people involved in the decision-making process do not often have access to this information. The 2D map in figure 1.1 does not provide a lot of detail but a 3D visualization could provide this detail to all people involved.

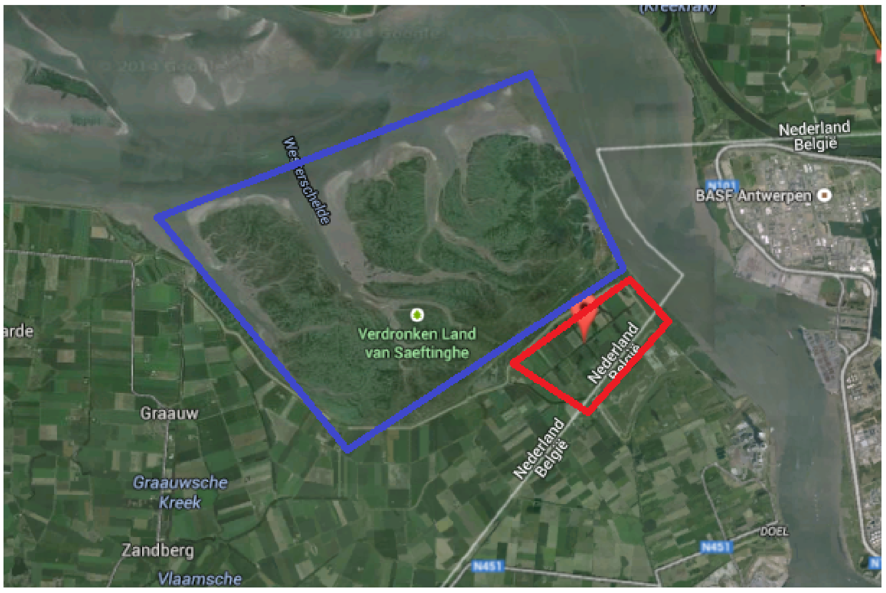


Figure 1.1: Image from GoogleMaps from the Hedwigepolder, the marker A and the red bounding box represent the Hedwigepolder. The area in the blue bounding box represents a salt marsh similar to the area to Hedwigepolder would become after the flooding.

The generation of a 3D visualization of a natural environment is not a standard task. It consists of three problems: generation of plant positions, generation of plant models, and rendering. The focus of this research is on the generation of plant positions. The generation of plant positions is essential in 3D visualization, because on these positions 3D plant models are placed. The complete plant distribution has to be realistic to obtain a convincing 3D visualization of a natural environment. This means that different kinds of plant types, for example pine trees or oaks, have to be placed in the correct location and have correct patterns. The correct locations for each plant type can differ depending on the environmental factor and plant type preferences, e.g. certain types grow only at a certain height. Correct patterns also differ per plant type and can depend on environmental factors and plant preferences. Patterns are different, because some plant type are scattered throughout the environment, while others are aggregated. This information can be derived from spatial data sources sometimes in combination with ecological data.

Therefore, to generate a realistic plant distribution it is necessary to use spatial data sources as input. Using spatial data as input leads to another problem, which is the availability of spatial data for a natural environment. For example, the natural environment in Hedwigepolder case does not yet exist which means that spatial data sources cannot be obtained through remote sensing techniques. This problem could be solved by using ecological models. Ecological models are able to generate spatial datasets for future by modelling ecological processes. Data that is generated are for example height, coverage, or biomass maps. For existing natural environments, data availability is often not a problem.

The current techniques in the 3D geo-visualization domain use spatial data to create 3D visualizations of natural environments. But, the plant positions by these techniques are either generated randomly or are semi-manually placed [6] [7][8] based on the spatial data input. Randomly generated positions do not give a realistic plant distribution with realistic patterns. Manually placing plant positions is not feasible when a plant distribution has to be generated for a large environment. In this 3D geo-visualization domain, Hempenius [9] recently investigated the complete pipeline for 3D visualization of vegetation based on spatial data. In his work, a 3D visualization of a natural environment was created based on a LCC (Land Cover Classification) map. The resulting plant distribution based on the LCC map had several problems and was evaluated by experts as not realistic. One of the problems of the distribution was that it showed grid pattern as can be seen in figure 1.2. There were grid patterns visible in his distribution, because each tile in the LCC map was directly mapped to a plant position. This shows that it is difficult to translate spatial data directly to positions for plants.

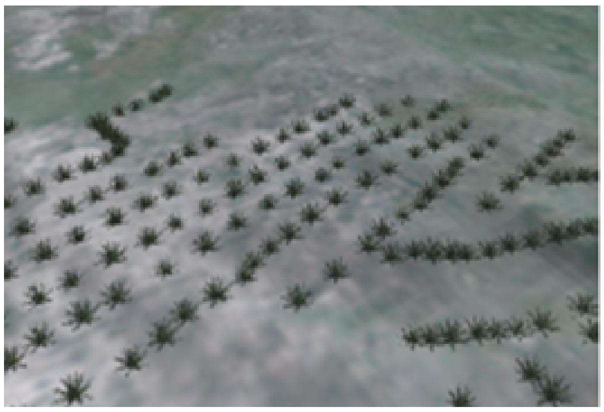


Figure 1.2: A 3D visualization of a plant distribution generated by the work of Hempenius [9]. One of the problems is the gridded outlook of the distributions.

In the remote sensing domain there are several techniques that extract plant positions from spatial data based on imagery [10], LiDAR (Light Detection And Ranging) data [11] or a combination of both [12]. However, these methods are only able to detect large plant objects, like trees, from high-resolution data. Positions for smaller plants, like flowers or small bushes, cannot be detected. In addition, when positions are detected, it is often not possible to provide information about which plant types are located at these positions. Further, because these techniques are based on imagery or LiDAR data they cannot be used for future areas. Such data is not available for future areas, and ecological models are not able to generate imagery or LiDAR data.

Besides the geo-domain, the field of PCG (Procedural Content Generation) has several techniques that are involved in the generation of plant distribution. PCG in general is a about automatically creating content from pre-defined rules and user-defined input parameters [13]. These techniques are able to generate a lot of different content with limited manual work. The main area for which procedural generation techniques are used is in computer gaming. In PCG, a few techniques focus on the creation of complete natural environments. Calculation of plant positions is a part of these techniques, which are called procedural ecosystem techniques. The disadvantage of the current techniques is that they do not generate realistic plant distribution for real environments, because they are not able to process spatial data correctly yet. Procedural techniques are however interesting, because they can produce content when limited data is available. Thus, for this research when the quality/resolution of the data is not sufficient to calculate plant positions. In addition, other techniques of PCG than procedural ecosystem generation techniques have been combined with spatial data input. In terrain generation, techniques have been developed that are able to use parts of existing height maps and combine them into a new map [14] [15]. Other techniques use the underlying height distribution obtained from spatial analyses of a single map to generate a new height map which results in a map which is very similar to the existing height map [16]. This is shown infigure 1.3.

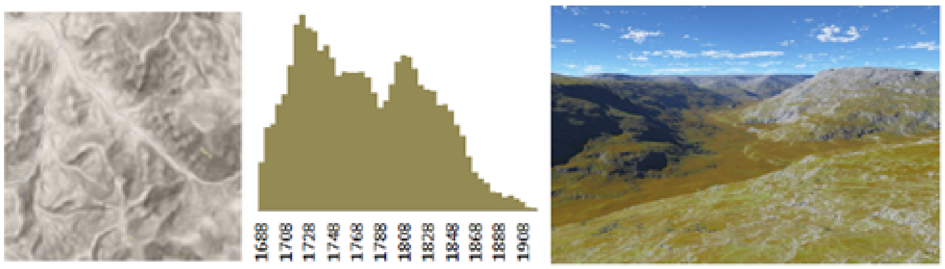


Figure 1.3: Terrain generation using the underlying distribution of an existing height map [16].

In city generation, the first set of data-driven techniques extract parts of cities from aerial images and combine these parts to obtain either the general outline of a city [17], or a detailed urban layout [18]. Also, it is possible to use it to add extra details to existing 3D models [19]. The other set of techniques [20] use geographical and sociostatistical (e.g. population density) maps to generate a city. Based on the input data different kinds of roads and houses are placed are placed on the maps. Figure 1.4 shows an example results of this method where for example high-rise buildings represent the places where there is high-population density. These techniques in PCG show that it is possible to combine spatial data with procedural techniques to obtain realistic results.

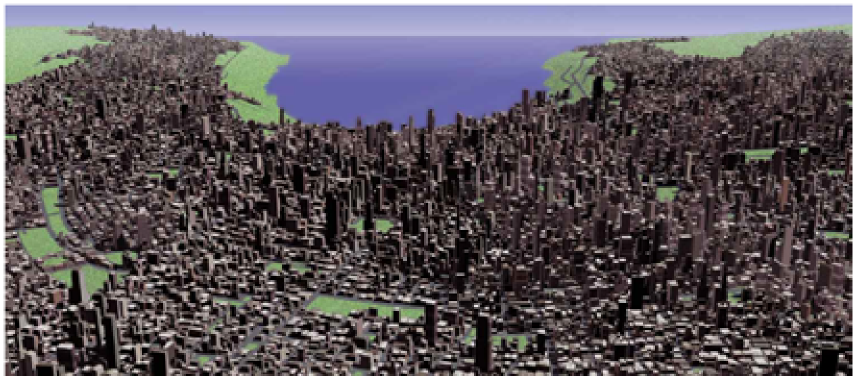


Figure 1.4: City generation based on spatial data input [20].

Based on the previous work, the question remains how to translate spatial data properly to a realistic plant distribution. Not only could it be applied for to creation of a natural environment for 3D visualization, but it could also improve the results of techniques in the field of plant detection, and improve the quality of current datasets by for example adding extra details about plants. Therefore, it is necessary that the proposed method in this research can be applied for a wide range of environments, works with different kinds and combination of spatial data sources (e.g. from point position data to raster maps), and that the results are consistent, logical and realistic.

1.2. RESEARCH OBJECTIVES AND QUESTIONS

The objective of this research is to develop a new method that can generate plant positions and differentiate plant types from spatial data to obtain a realistic plant distribution for an existing or future area. Based on this research objective the following main research question has to be answered:

- How can plant positions be generated and classified from spatial data of existing and future areas to obtain a realistic plant distribution?

To answer the main research question, the following sub research questions have to be answered:

1. What are the current techniques to determine plant positions?
2. What kind of spatial data is necessary to generate and classify plant positions of different size, for both future and existing areas, and how does the data influence the results?
3. How can plant point positions for different plant types be generated based on spatial data?
4. How can we determine where to place which plant type based on spatial data?

1.3. RESEARCH SCOPE

The core of this research is to develop a plant placement algorithm that uses spatial data to generate a plant point distribution for both existing and future areas. The algorithm must be able to generate both small and large plants. The distribution should have a realistic composition and realistic patterns. Realistic composition means that different plant types are placed in the correct location of an area. Realistic patterns mean that the clustering of plants for each plant type is correct. A plant point distribution is realistic when the composition and patterns matches with the spatial data provided for a certain environment. Therefore, realism is not judged on whether plants are in the exact same positions as in the real environment, because it is not possible to place small plants on the exact same positions as in the real environment. Also, it is not possible to validate this for future areas.

The motivation of this project is to use the plant placement algorithm to generate a realistic plant distribution as input for the creation of a realistic 3D visualization of a natural environment. Creating a 3D visualization of the plant distribution and the surrounding environment is however out of scope of this project. This research only focus on the generation of a realistic plant distribution by developing an algorithm that generates point positions with semantics (e.g. the plant type) based on different kinds of spatial data input. Figure 1.5 shows the global process from spatial data to a 3D visualization. The focus of this research project is on the first two parts. Thus, the spatial data and the plant distribution generation.

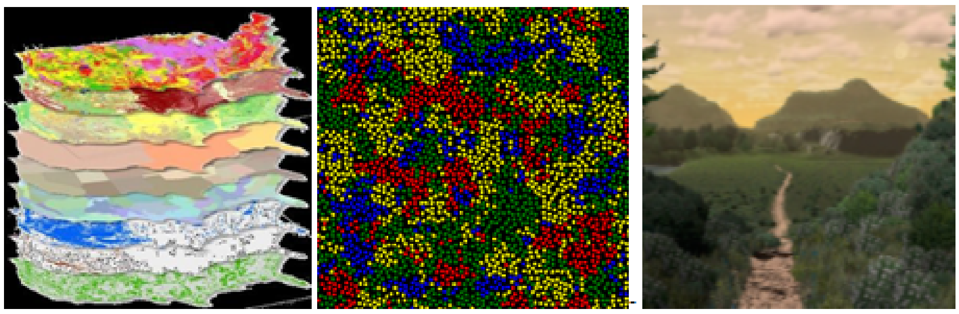


Figure 1.5: Framework for 3D visualization of spatial data. The first image represents a spatial data collection that is used to generate a plant point distribution shown in the second image. The third image shows a 3D visualization of a natural environment [21]. The focus of this research is on the first two images.

The research is conducted in cooperation with NIOZ located in Yerseke, the Netherlands. NIOZ is a research institute that focuses its research on marine ecology. The tests will be applied on data and that is available and can be provided by the NIOZ. This means that the plant distributions are generated for salt marsh environments. However, this does not mean that a method will be developed that is only applicable for salt marshes. The aim is to develop a method that is applicable for a wide-range of different environment types.

1.4. RESEARCH CONTRIBUTIONS

Contributions of this thesis research can be seen from both a scientific as well as from a societal point of view:

1.4.1. SCIENTIFIC CONTRIBUTIONS

The main contribution if this research is that it develops a plant distribution generation algorithm that translates spatial data sources from existing and future areas to a plant distribution where positions can be generated for both small and large plant types. The current techniques are not able to produce realistic results, because they do not work properly with spatial data, they cannot produce positions for every plant, or they do not work with for future areas.

Another contribution of this research is that it gives an overview of how to integrate and combine information from different kinds of spatial data so that it can be used to determine plant positions. In this research, it is investigated which data sources are normally available for natural environments and how these could be used to contribute to the generation of a realistic plant distribution. The current techniques using spatial data only work with high-resolution imagery or LiDAR data.

Moreover, the proposed algorithm in this research is not a replacement for techniques that determine plant positions. Instead, the results of these techniques can be used as input for the proposed algorithm to enrich the input data. For example, the plant positions obtained from plant detection techniques can be enriched by adding semantics to each plant position or by adding data points of smaller plants that are normally not detected by plant detection techniques.

Finally, the results of the proposed algorithm can not only be used for realistic 3D visualization of natural environments, but can also be for other areas like improving the quality of existing tree and plant maps.

1.4.2. SOCIETAL CONTRIBUTIONS

The first societal contribution is that the plant distribution generated by the algorithm can be used in combination with 3D visualization for aiding in decision-making cases. Distribution can be generated for future areas like the Hedwigepolder case to help people understand what happens with an area when its flooded. Thus, it can be used as a tool to communicate to the society how future natural environments develop.

The communication of how natural environments will look in the future is also possible to show with a drawn 2D aesthetical map. However, the main disadvantage of aesthetical maps is that a professional has to draw it, which cost a lot of time and money. Also, with such a map it is only possible to show one situation of the area at a certain level detail. The proposed algorithm is able to generate plant distributions automatically. This means that that many distributions for different areas can be generated in short amount of time just by changing the data input. In addition, the distribution are generated with highest detail, e.g. a position for each plant and the user is able to decide for himself which part or parts of the area he want to investigate. Thus, the results of this algorithm are preferred over aesthetical maps, because it is much faster and it allows the user to self explore the distribution. Further, each distribution can easily be combined with a 3D visualization in comparison with aesthetical maps.

Currently, companies are generating detailed tree maps of complete countries for various applications like tree management canopy assessment, and risk and utility management. These maps are limited to trees and only provide information about the position, height, and crown area of the trees. These datasets could be improved the algorithm of this research by adding information about the types of the plants or by adding position data of smaller plants.

Finally, this project was carried out partly at NIOZ. The research institute has great interest in how plant positions can be determined from the spatial data generated by their ecological models. They are interested in this, because they want to turn the output of their ecological model in realistic 3D visualizations. To generate a 3D visualization, it is necessary to have a plant distribution. They want to use the 3D visualizations to promote their work and nature in general to the society.

1.5. THESIS OUTLINE

This thesis consists of five chapters, which are organized as follow:

Chapter 1 introduces the research topic, the problem statement, objectives and scope of this research.

Chapter 2 presents the literature findings of techniques involved in plant position generation or data generation for future areas. This chapter starts with discussing plant detection techniques, procedural ecosystem generation techniques, and ecological models. This is followed with an overview of each technique with their advantages and disadvantages. At the end, a preliminary framework is presented for the proposed algorithm in chapter 3. This chapters answers the first subresearch question.

Chapter 3 presents the plant placement algorithm. This chapter starts with an overview of the algorithm explaining its criteria, input data, parameters and flow. Next, a detailed technical explanation is given about the working of the algorithm. This chapter answers the second, third, and fourth subresearch question.

Chapter 4 generates results for real environments using the plant placement algorithm of chapter 3. The results are discussed with statistical and expert validation. This chapters investigates whether the results of the algorithm are realistic and matches with the input spatial data. It answer the questions whether the results are realistic and correct.

Chapter 5 concludes the research by answering the research questions and presenting future work.

2 LITERATURE FINDINGS

This chapter introduces the literature findings of this research. The aim of this chapter is to explain the current techniques that are involved in the generation of plant positions and the techniques for the generation of spatial data for future areas. An overview is given of each type of technique including the advantages and disadvantages for each. This chapter first starts with explaining plant detection techniques, and second the procedural ecosystem generation techniques. These techniques generate or detect point positions for plants. This is followed with the discussion of ecological models which are able to generate data for future areas. In the end, it is discussed which aspects of each technique could be used for the proposed algorithm in chapter 3. In the last section, a preliminary framework for the algorithm is proposed based on the discussed aspects. This chapter is related to the second subresearch question.

2.1. PLANT DETECTION

Plant detection techniques are sets of techniques that detect plant position from spatial data sources. These techniques use existing spatial data sources, like aerial images, LiDAR or a combination, to determine the positions of plants. There are many techniques for detection of plants using aerial images or LiDAR data [22]. These techniques are not discussed in detail here. It is more important to understand the main idea behind these techniques.

The main challenge of plant detection based on imagery is to work with the contrast between trees and their background. Figure 2.1 shows an aerial image of trees. By eye, it is possible to determine the location of each tree. This is possible, because in the image trees have a higher value (white) in the middle of their location and lower values surrounding the middle (black). The contrast between these values is used by the imagery-based plant detection techniques to determine the locations of each single plant. Further, figure 2.1 shows an example output of a plant detection algorithm. I this case, it is clearly visible that trees have higher values in the middle of their position and lower values around it.

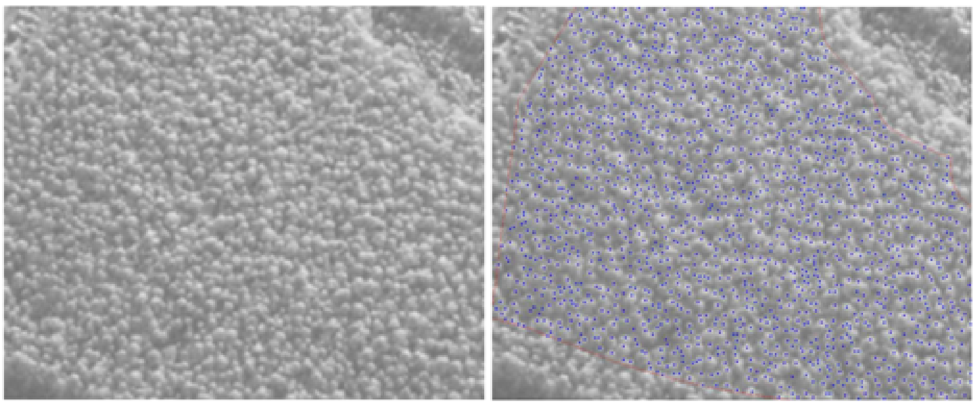


Figure 2.1: Aerial images of a forest. The contrast between the different trees is used by the plant detection techniques to determine the positions of the trees. On the right, a result of such a technique is shown. (Google Image)

The same principle is applied for LiDAR-based plant detection techniques [23]. The LiDAR system generates a large point cloud for an area. From these points intensity values can be determined. Figure 2.2 shows example point clouds with varying intensity values; higher points have a larger intensity value, while lower points have a smaller intensity value. Between trees, there is a drop in intensity values, which is used by the LiDAR techniques to determine the positions of plants. Further, it is possible to combine the contrast of the aerial images with that of the LiDAR point clouds to increase the accuracy of plant detection.

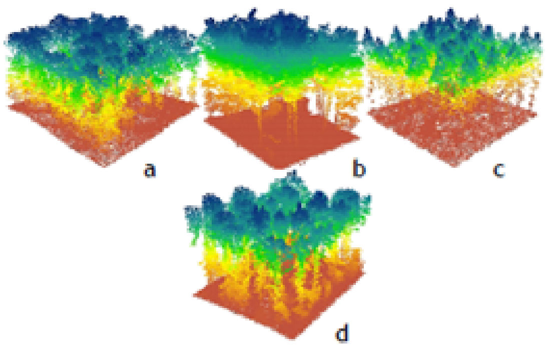


Figure 2.2: Example point clouds of trees. These point clouds show that higher points have a different intensity than lower points [11].

To obtain accurate results for either imagery-based or LiDAR-based techniques, it is necessary to use high-resolution data. When low-resolution data is used the contrasts between the plants cannot be properly detected. Because of this lack of contrast, only large plants can be detected. In figure 2.3, two aerial images based on the infrared band are shown. The first image only contains several small groundcover plants. The second image also contains trees. In the first image, it is not possible to detect any plant position. In the second image, the trees are easily detectable, because they consist of many pixels which capture the contrast between the background and trees. Other disadvantages of both imagery-based and LiDAR-based techniques are that imagery and LiDAR data is only available for existing environments, and detected positions cannot be given a semantic meaning often, e.g. assigning a certain plant type to a position.

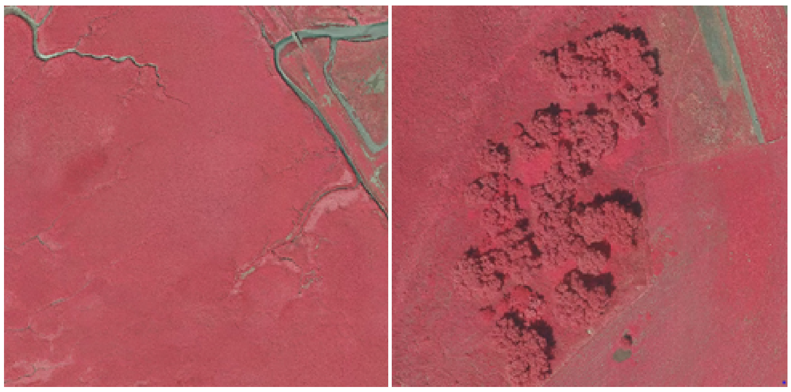


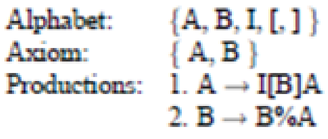
Figure 2.3: Infrared aerial images with on the left an area with only different types of groundcover plants on the right an area which contains trees as well. The trees can be easily detected. It is not possible to detect the position of any other plant type.

2.2. PROCEDURAL ECOSYSTEM GENERATION

There are two main approaches in procedural ecosystem generation: local-to-global and global-to-local. The local-to-global techniques generate ecosystems by simulating plant growth and plant competition. During the simulation, plant positions can change. The global-to-local techniques use several global rules to place plants with limited interaction between the plants. After, a point is placed its position is fixed.

2.2.1. LOCAL-TO-GLOBAL

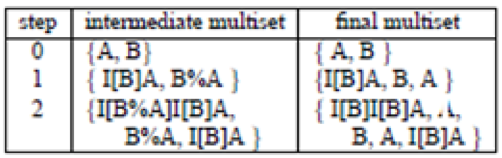
The local-to-global approach uses simulation over time to generate a plant distribution. The first local-to-global methods were introduced by Deussen [24] and Lane [25]. Their work is the basis for the current local-to-global methods. The main technique behind the local-to-global techniques is the multiset L-system. An L-system is a technique that rewrites iteratively an input string based on pre-defined rules [26]. The multiset L-system rewrites a set of strings which are stored together. Figure 2.4 shows an example multiset L-system.



Alphabet:	{A, B, I, [,] }
Axiom:	{ A, B }
Productions:	1. A → I[B]A
	2. B → B%A

Figure 2.4: A multiset L-system [25].

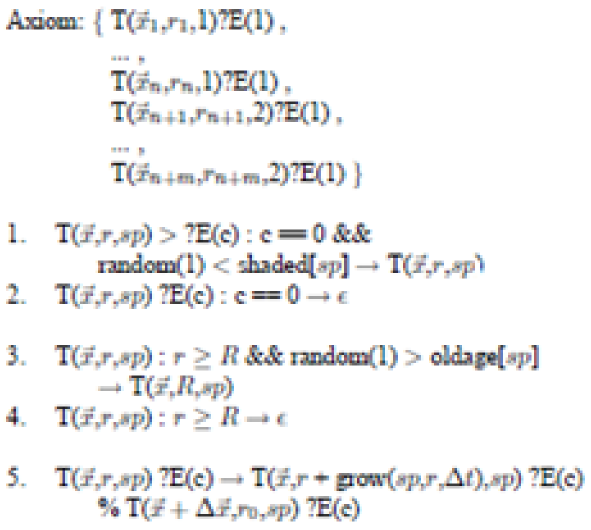
In the multiset L-system there are two important inputs: the axiom and the productions. The axiom is the input string set of the L-system. The productions are the rules that rewrite each string in the axiom. In figure 2.5, the working of the multiset L-system is demonstrated.



step	intermediate multiset	final multiset
0	{A, B}	{ A, B }
1	{ I[B]A, B%A }	{I[B]A, B, A }
2	{I[B%A]I[B]A, B%A, I[B]A }	{ I[B]I[B]A, A, B, A, I[B]A }

Figure 2.5: The process of the multiset L-system [25].

In ecosystem generation, each string of the multiset system represents a single plant. The rules are used to simulate plant growth, plant competition and plant propagation. Figure 2.6 shows a example multiset L-system for generating a plant distribution over time.



Axiom: $\{ T(\vec{x}_1, r_1, 1)?E(1),$

$\dots,$

$T(\vec{x}_n, r_n, 1)?E(1),$

$T(\vec{x}_{n+1}, r_{n+1}, 2)?E(1),$

$\dots,$

$T(\vec{x}_{n+m}, r_{n+m}, 2)?E(1) \}$

1. $T(\vec{x}, r, sp) > ?E(c) : c == 0$ && random(1) $<$ shaded[sp] $\rightarrow T(\vec{x}, r, sp)$
2. $T(\vec{x}, r, sp)\ ?E(c) : c == 0 \rightarrow \epsilon$
3. $T(\vec{x}, r, sp) : r \geq R$ && random(1) $>$ oldage[sp] $\rightarrow T(\vec{x}, R, sp)$
4. $T(\vec{x}, r, sp) : r \geq R \rightarrow \epsilon$
5. $T(\vec{x}, r, sp)\ ?E(c) \rightarrow T(\vec{x}, r + \mathrm{grow}(sp, r, \Delta t), sp)\ ?E(c)$ % $T(\vec{x} + \Delta\vec{x}, r_0, sp)\ ?E(c)$

Figure 2.6: Multiset L-system of local-to-global method [25].

In the example, a plant is represented by two modules: $T(x, r, sp)$ and $E(c)$. The T modules give information about characteristics of the plants where x is the position of the plant, r the size (radius) of the plant, and sp the type of the plant. The other module $E(c)$ is a communication module that states if a plant is dominated by another plant. The value one means that the plant is not dominated and zero that the plant is dominated by another plant. The rules use the variables in the modules to decide how to update each module. For example, rule 2 removes a plant from the set if it is dominated by another plant, e.g. when c is equal to 0. The final distribution is obtained by iterating over the rules X times. This number is defined by the user. The number of iterations influences the final plant distribution that is obtained. Example results of the above multiset L-system are shown in figure 2.7. The distributions shown are from the same simulation but stopped at a different iteration.

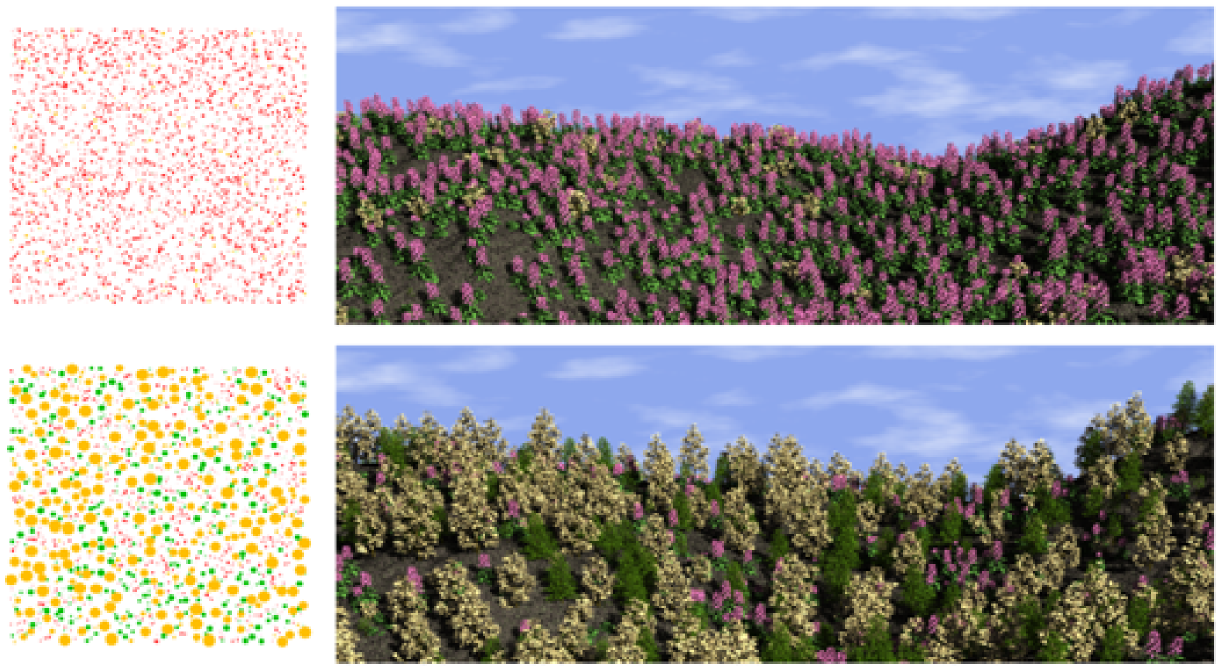


Figure 2.7: The simulation of a natural environment over time based on a local-to-global technique. In this example it is shown that the white trees slowly take over the whole environment [25].

Based on the multiset L-system from the previous example, other local-to-global techniques have been developed. Benes introduced two extensions [27][28], the first aimed at creating a stable ecosystem where no single plant type or small subset of plant types would take over the whole environment, in the case of sudden unexpected changes in the environment. The second extensions simulated plant competition between groups of plants instead of individual plants. Ch'ng [29] created a system that supported advanced competition for resources, space, sunlight, and several other ecological processes.

The main advantage of local-to-global methods is that they can show ecological mechanisms over time, because these methods simulate an environment over time. Each iteration is a time step. Questions of how a plant distribution changes based on ecological processes can be answered with this. However, the current ecological processes that are used are still limited when comparing it with the modeled ecological processes in dynamic ecological models to achieve a realistic plant distribution. Further, it is difficult to general apply these methods, because setting the parameters for these systems requires a lot of expertise, knowledge of how plants grow and interact with each other, and a lot of testing and tweaking. In addition, these parameters are different for each environment. Another disadvantage is the lack of controllability of the method, because the user has to stop the simulation, but is not able to know how the number of iterations affects the final point distribution.

2.2.2. GLOBAL-TO-LOCAL

The global-to-local approach uses global parameters to calculate once fixed plant position for a specific environment and specific moment in time. The global-to-local approach does not use simulation over time instead a plant distribution is calculated once. This means that a plant will not die, grow or change location: the position is fixed.

One of the first technique in the global-to-local category was introduced by Hammes [21]. His technique uses a terrain map where for each tile in the map probabilities are calculated that represent the likelihood of that tile for a certain ecotype. The ecotypes used by Hammes in his example are bush, marshland, small bushes and grass, grass on steep slopes and exposed rocks. The likelihood for a ecotype is calculated by a function using the parameters of elevation, relative elevation, slope and slope direction. For each tile, the ecotype with the highest probability is selected. Each tile receives a texture based on the assigned ecotype and depending on the ecotype, plants are randomly scattered in that tile. The amount of plants depends on the ecotype. Example results are shown in figure 2.8 where different textures can be clearly identified, as well as the tiles where the plants are randomly scattered.

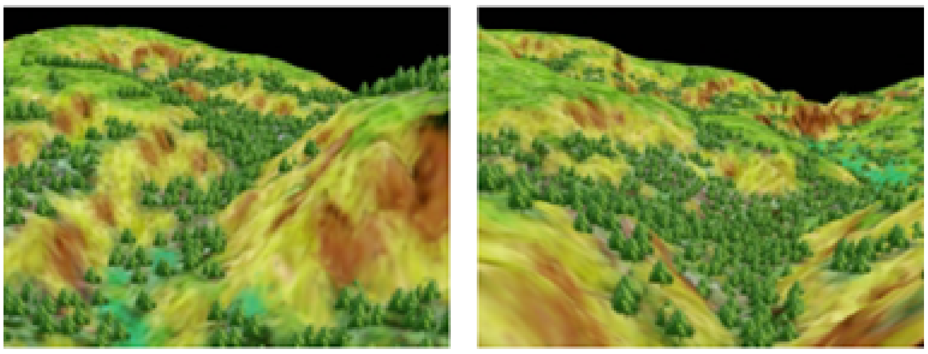


Figure 2.8: Results of the technique of Hammes [21].

This method shows the possibility of using the characteristics of a height map to determine where to place plants and to calculate an ecotype per tile. However, the actual placement of plant positions is random. If an ecotype contains plants then the positions of these plants are randomly determined. Also, all the plants belong to the same plant type in this method. No difference is made between different plant types.

The second method that is discussed is developed by Lane [25]. This method uses a dart throwing mechanism to place plants of different plant types. In this method, each plant type has its own probability field. The method starts with placing plants from largest plant type to smallest plant type. For each plant, a random position is chosen based on the probability field of each plant type. After a plant is placed the probability fields of every plant type is updated. How each probability field is updated depends on how the neighborhood kernel is defined for the plants type. Figure 2.9 shows examples of neighborhood kernels. A plant type can have different neighborhood kernel for each plant type.

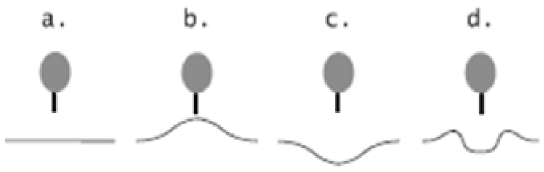


Figure 2.9: Kernels for representing plant influences [25].

The usage of neighborhood kernels between different plant types and the process of this method is shown in figure 2.10. In the example there are two plant types defined. Both have their own probability field. After the first plant is placed the probability field is updated of both plant types with a different neighborhood kernel. Plant type A has a positive effect on the plants of its own, and a negative effect on the plant of plant type B.

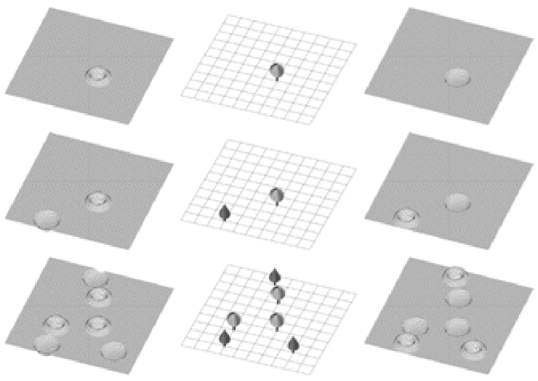


Figure 2.10: Kernels for representing plant influences [25].

The idea of neighborhood operations for global-to-local techniques was further elaborated in Alsweis et al. [30] . In his work, neighborhood operations were formalized in a model: the FoN (Field-of-Neighborhood) model. This model describes the area that is influenced by for each plant. This area is called the zone of influence. The zone of influence is dependent on the size of the plant. Larger plants have a larger zone of influence and vice versa. Further, the actual positions of the plants were calculated using the PDD (Poisson Disk Distribution) technique in combination with Wang tiling and the FoN model. The PDD technique is a method that generates points in an area, where each point has a certain minimum distance to each other. An example PDD is given in figure 2.11. Poisson disks are generated by randomly placing points in an area which are then either accepted or rejected based on the required minimum distance that is defined between the points. The described process is dart throwing, which is slow. There are several improvements to increase the performance, but these are not important for this research.

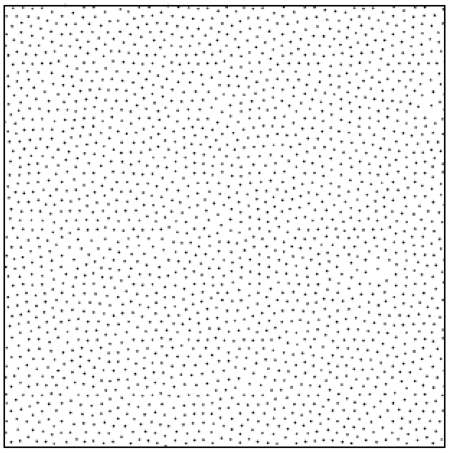


Figure 2.11: Poisson disk distribution [31].

Even with these improvements for PDD generation, it is not feasible to generate a large amount of points for a large area. To reduce the amount of points with PDD generation, the Wang Tiling method is used. A Wang tiling consists of square tiles with color-coded edges. These colored tiles are called Wang tiles. A Wang tiling is generated by matching the borders between each tile. Thus, the edges between two tiles have the same color. PDD is combined with the Wang tiling method by generating a PDD for each Wang tile. The points in the color-coded edges are copied to the edges on the other Wang tiles containing the same color. The colored borders are thus repeating the same point positions to create seamless borders between each tile. The complete Wang tile generation process with the final PDD is shown in Figure 2.12. One problem of using this Wang tiling technique in combination with the PDD generation is the corner problem. This means that it possible that point can overlap in the area where the corners are touching each other. This is normally solved by throwing points way in all the corners areas until there are no overlapping points anymore. This is done after the whole distribution is generated.

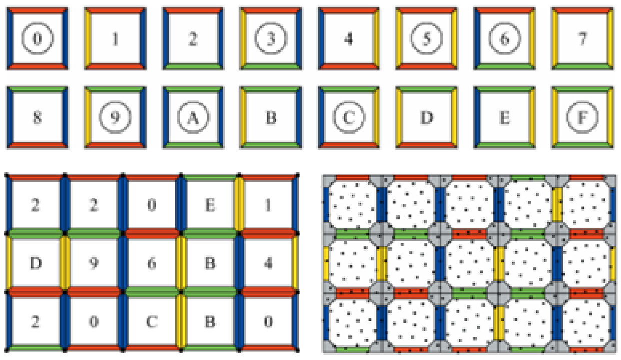


Figure 2.12: The image on the top represents all the possible Wang tiles. Each Wang tile has its unique combination of colors. The tiles with the circle are used in the Wang tiling shown in the image in the bottom left where each tile matches its neighboring tiles. Finally, based on the repeating colored borders and tiles a final point distribution is obtained in the image on the right. Tiles that have the same number or letter also have the same point distribution [31].

The FoN model is incorporated in this process by giving the points during the PDD generation a random size and based on that size the zone of influence is calculated which represents in this case the minimum distance that this point has to the other points. Thus, this means that larger point have a larger minimum distance to the other points. Figure 2.13 shows Wang tiles that contain a PDD where points have different zone of influences and an example result. In the method of Alsweis the classification process of the generated points with Wang Tiling and PDD was not discussed as that was not part of the aim of his research. The aim of his research was to generate a seamless PDD distribution for a large area in combination with the FoN model.

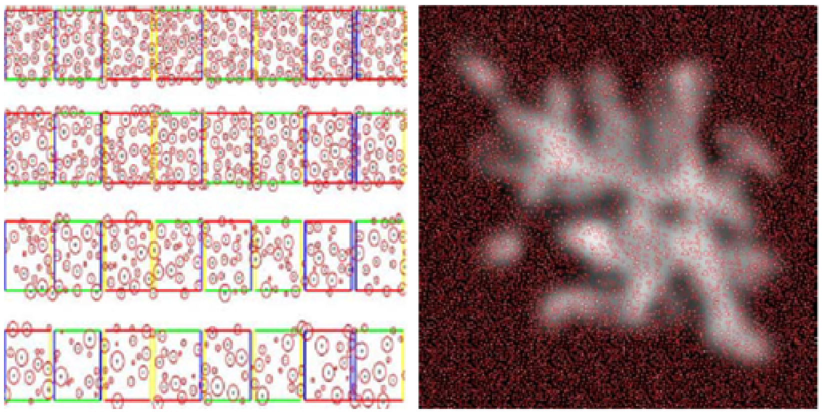


Figure 2.13: Wang tiles with different zones of influences used for generating the example result on the right [30].

The last method discussed is developed by Weier [32], which implemented a method that uses the PDD generation technique with Wang tiling and also classifies the point to a plant type. In his method, he first creates a Wang tiling and fills it with points using the PDD generation technique. Next, for each points the likelihood for each plant type is calculated. The likelihood of a point is based on height and soil factors, and the size of the point. During the last step, each point gets a plant type assigned that has the highest likelihood for that point. These plant types are not fixed yet. This is followed with a neighborhood calculation. For each points its neighbors are retrieved and a certain neighborhood influence is applied on the likelihood of each plant type of these neighbors. The amount of neighborhood influence of each point is dependent on its own likelihood. This means if the likelihood of the assigned plant type is large than that point has a large influence on its neighbors. After the neighborhood influences are included in the likelihoods of each point, each point is classified again. This process is repeated until most of the points do not change plant type anymore. Results of this method are shown in Figure 2.14.

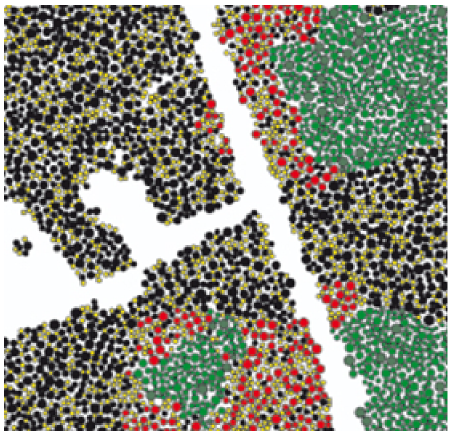


Figure 2.14: Results of the technique of Weier [32].

The main advantage of the global-to-local methods is that they can quickly distribute possible plant locations over an area by using the Wang tile approach in combination with PDD generation when no prior information is given about where plants are located. Also, some methods have examples of usage of existing data sources to improve realism. Nevertheless, the usage of real data sources is limited: only height maps and some soil maps are used. No attention has been given to possible usage of other maps or parameters. Further, the current classification of points does not yield a realistic result. Often, the points are semi-randomly classified. There is no ecological or data-driven basis behind determining where a specific plant type should be placed besides the neighborhood operation. The neighborhood operations give the possibility to generate different patterns, but the correct configuration can only often be obtained through extensive testing instead of a data-driven approach.

2.3. ECOLOGICAL MODELS

This section discusses ecological models, which generate spatial maps or point data. Two types of ecological models are discussed: neutral and dynamic landscape models. First, the neutral landscape models are discussed. These models focus on generating specific patterns for every type of plants based on shape metrics. The result of these models is a categorical raster map. The second type of model is the dynamic landscape model. These models generate spatial maps with information about for example, coverage, height, and/or biomass using complex dynamic ecological processes. The other possibility is that it generates a point map, but that depends on the dynamic model.

2.3.1. NEUTRAL LANDSCAPE MODELS

Neutral landscape models generate categorical maps where each category is a plant type. These models are used to generate spatial patterns for each plant type. The models generate maps, which are not influenced by spatial constraints (e.g. height, soil conditions) or dynamic ecological processes (e.g. erosion). Conditions at each location on the map are equal. Therefore, these models are called neutral. This property makes it possible to investigate how shape metrics influence the shape of patterns. Shape metrics are a measure of the shape of patterns. Neutral models have two important parameters for each plant type: a shape metric value describing the patterns of the plant type and an occupation value that describes how much space is occupied by the plant type on the input map. In this subsection, two types of neutral models are explained: the MRC (Modified Random Clusters) models and the fractal-based models.

MRC MODEL

The MRC technique generates spatial patterns using a percolation map [33]. A percolation map is a grid where each tile has been given a random number drawn from a uniform distribution and each random number is compared with a global threshold value p. Tiles with a random value lower than p are assigned a value of 0. Otherwise, they are assigned a value of 1. The result is a percolation map. The value for the threshold p is based on shape metric input. In the percolation map, clusters are identified using neighborhood rules (e.g. 4-connected or 8 connected neighborhood). Clusters are only identified for the tiles with a value of 1. After the cluster generation, each cluster is assigned a certain plant type. Assigning plant types to a cluster is based on the amount of occupation each plant type has been given by the user. Clusters are assigned in an ordered approach so that the occupation value is approximately met for each plant. In the last step, the remaining black pixels are classified. This is achieved by taking the 8-connected neighborhood for each black pixel and count the number of occurrences of each plant type of the pixels neighbors. The plant type with the most occurrences is assigned to that pixel. In case of a tie, a plant type is selected randomly. When there are no neighboring plant types, a plant type is selected randomly where each plant type's probability is based on its occupation value. The complete process is visualized in figure 2.15.

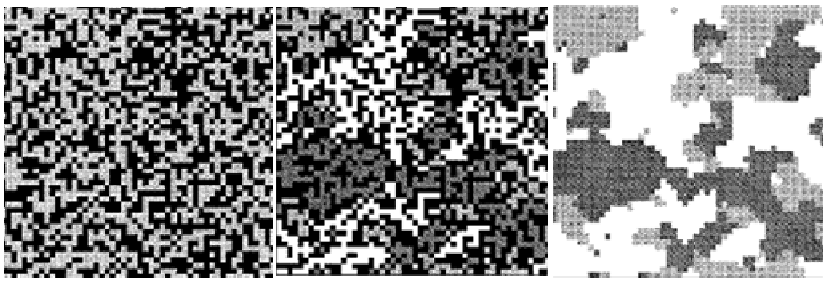


Figure 2.15: The first image shows a percolation map with p=0.52 as threshold (white pixels are marked). The second image detects clusters in the percolation map and assigns each cluster a plant type. In the third image, the rest of the map is filled [33].

The parameters that control the output of the MRC method are the threshold value p, the neighborhood criteria, and the number of plant types with their occupation value. A low threshold value p results in a high fragmentation of clusters, i.e. smaller patches for each type. The neighborhood criteria could be changed to obtain patterns that are oriented in a certain direction by using a different neighborhood kernel. In the MRC model, each type can have a different occupation value. Figure 2.16 shows different examples of the MRC model with high fragmentation, diagonal-neighborhood criteria, and different occupation values.

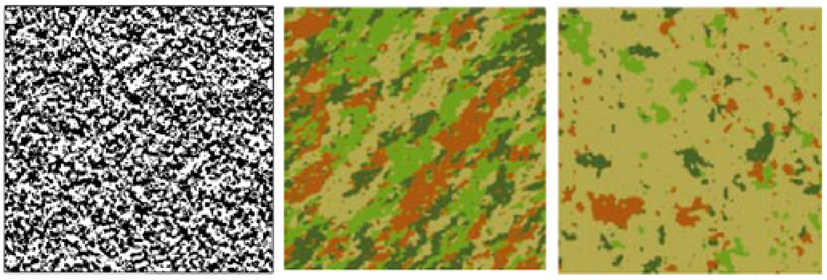


Figure 2.16: Maps with different fragmentation, neighborhood kernel and occupation [33].

The MRC technique is able to generate patterns of different fragmentation, orientation, and occupation. Occupation values can be different for each plant type, but the fragmentation and orientation of the patterns of each plant type are the same for the whole environment. Also, neutral models are by definition not able to deal with varying occupation for a single plant type.

FRACTAL-BASED MODEL

The fractal-based model uses a fractal map for each plant type to generate a categorical map [34]. The fractal map is generated by a fractal algorithm that is controlled by the Hurst parameter which defines the shape of the patterns. A high Hurst value means a high correlation and a low value a low correlation in the fractal map. Often, the fractal maps are generated by the midpoint displacement technique. The midpoint displacement technique assigns a random value to each corner of the input map. Next, new points are placed between each corner. The value of these points are based on the average of the two corners and a random value based on the Hurst parameter. A low Hurst results in a high random value and a high Hurst in a low random value. The process of adding a random value to the new points is also called displacement. After this, the map contains four sub squares for which the same process could be applied. The selection of new points and the creation of new sub squares are shown in figure 2.17.

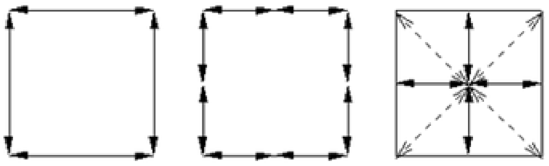


Figure 2.17: Principle of mid-point displacement. Each tile is subdivided in four smaller tiles where each the midpoint value is a measure of the four corners [35].

This process is repeated until the desired resolution is obtained, e.g. when every tile of the input has a fractal value. A fractal map is generated for each plant type. A fractal value represents the likelihood that a certain plant type is placed in that tile. The next step is to generate a categorical map with the fractal maps and occupation values for each plant type. First, for each plant type, the X highest fractal values are selected from the fractal map. X is a percentage which is equal to the occupation value of the plant type. Figure 2.18 shows this step.

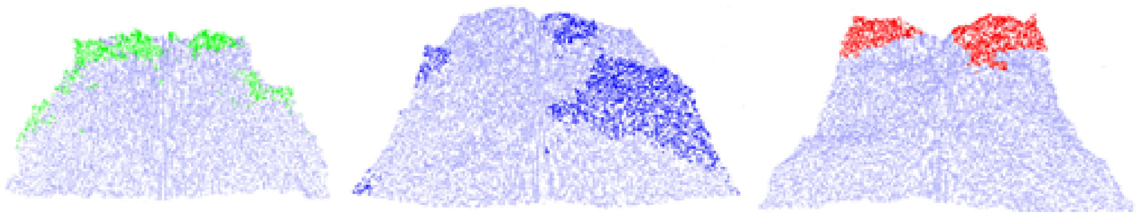


Figure 2.18: Based on a global threshold defined for each plant type, the midpoint displacement maps are thresholded [34].

Second, a check is made which tiles on the map got more than one plant type assigned to them. In that case, the type with the highest fractal value is assigned to that tile. The types that not get assigned, because they had a lower fractal value, perform a search on the set of tiles that are not classified that have the highest fractal value. The tile with the highest fractal of the set of not classified tiles is assigned to that plant type. Figure 2.19 shows the conflicting tiles in dark gray based on the classified map of figure 2.18 and the final result. The yellow tiles are the tiles found by the search.

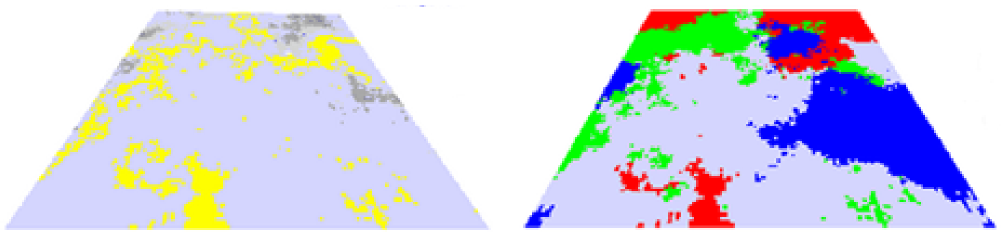


Figure 2.19: Thresholded maps are combined, which leads to conflicts and these are resolved by taking the highest value for that position. Types that lost a conflict are placed on an empty tile with the highest probability [34].

The fractal-based model is able to generate patterns of different fragmentation between the plant types. This is controlled by the Hurst parameter. In addition, the occupation per plant type on the map can be different. A disadvantage is the usage of the midpoint displacement function to generate fractals, because this has the undesired side effect that only a square map can be categorized. Further, varying occupation values are not supported for a single plat type and the method can only be applied to raster maps.

2.3.2. DYNAMIC LANDSCAPE MODELS

Dynamic landscape models simulate information to show how a landscape develops over time. These models use complex ecological principles and processes to perform these simulations [36]. There are two types of dynamic landscape models: cellular-based and individual-based. Cellular models generate maps with information about for example biomass, coverage, and height. Individual models generates a point set.

The cellular models uses the principle of CA (cellular automata) to generate maps. A simple CA is a system where each tile of the map is in one of the pre-defined states. Examples states could be vegetation or no vegetation. Tiles can change its state based on transition rules. In each iteration a new state is calculated for each tile based on the transition rules. Transition rules calculate the new state based on the states of the neighboring tiles. A well-known example of cellular automata and the use of transition rules is The Game Of Life [37]. This example has two states and the transition rules are based on the dominance of the neighboring states. The transition rules are shown in figure 2.20.

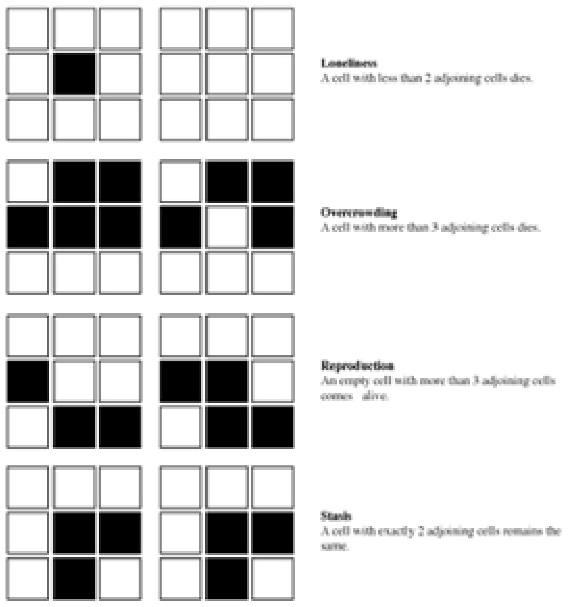

Loneliness
A cell with less than 2 adjoining cells dies.
Overcrowding
A cell with more than 3 adjoining cells dies.
Reproduction
An empty cell with more than 3 adjoining cells comes alive.
Stasis
A cell with exactly 2 adjoining cells remains the same.

Figure 2.20: Cellular automata: game of life rules [37].

In dynamic landscape models these transition rules represent complex biological and environmental processes, like erosion or plant growth. States of CA can also be continuous. This gives the possibility to use continuous number for a tile to represent the amount of biomass or coverage. This is implemented in the dynamic models. The output of these model is dependent on the input map, e.g. the initial state of the map. A dynamic model could be used to generate maps for future areas if the ecological process that are modeled are present in that area. Only a correct initial situation has to be provided as input to the model to be able to generate realistic maps for future areas. Figure 2.21 shows the process of how a map is generated by an ecological model.

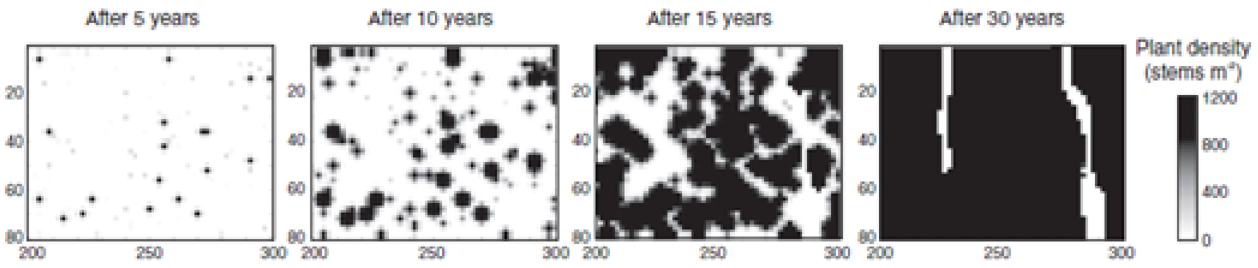


Figure 2.21: Cellular automata: The process of generating a map with a dynamic cellular ecological model is shown. The map changes over time based on certain ecological processes [38].

Individual-based models generate point sets by modeling interactions between individuals. Modeling interactions on individual level is very difficult and therefore these models are rarely available for complex environments. Interactions are modeled similar as the transition rules in CA. Figure 2.22 shows an example individual-based model with different time steps.

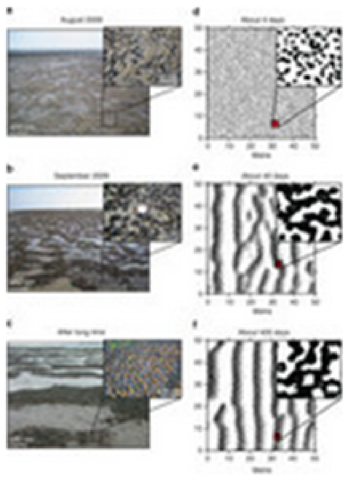


Figure 2.22: Individual mussel bed model [39].

The main advantage of dynamic models is that they can generate data for future areas. Data that is generated is similar as what can be obtained for existing areas with remote sensing technique. But, the data is often generalized, because a dynamic model is also an abstraction of reality. This means that coverage data is often limited to a single plant type. The usage of an individual-based model is preferred, because they already provide point locations of each individual. Often, such models are not available, because a realistic model is difficult to implement.

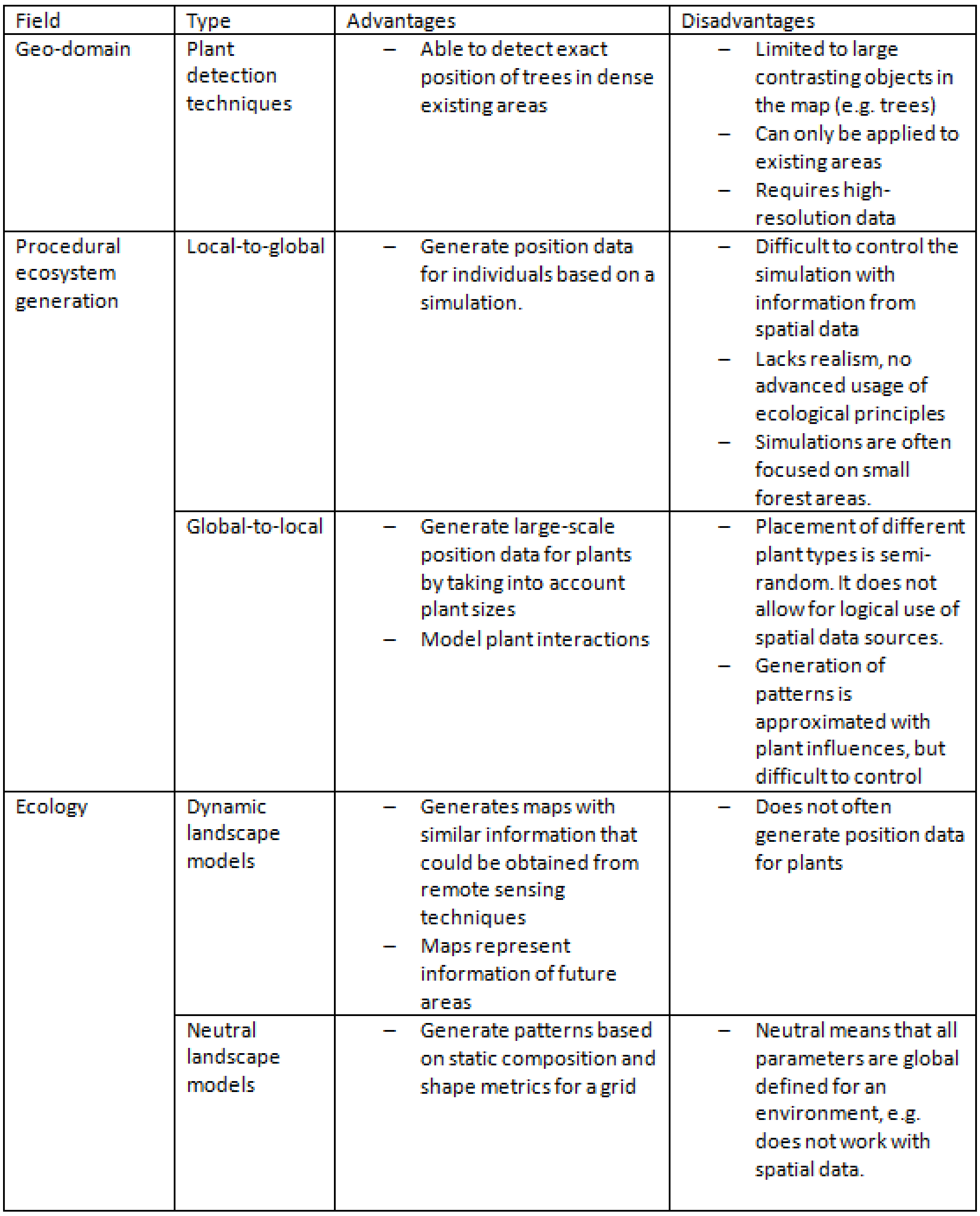
2.4. DISCUSSION

In this chapter, three types of techniques have been discussed: plant detection techniques, procedural ecosystem generation techniques, and ecological models. An overview of these techniques is given in table 2.1. The first techniques discussed were the detection techniques. Plant detection methods are able to obtain accurate plant position using aerial images, LiDAR data or a combination of both. However, these techniques have several major limitations. Detection of plant positions is limited to large plants, such as trees. This is because the plants should be composed of many pixels/LiDAR points to differentiate these plants effectively from each other and the background. Therefore, high-resolution data is necessary. Small plants cannot be detected accurately, since they cannot be differentiated from the background. This is because a pixel in an aerial image could be composed of several small plants. The final disadvantage is that these techniques cannot be applied to future landscapes, because aerial images or LiDAR point sets are not available for areas that do not yet exist.

The second group of techniques that was investigated are the procedural ecosystem generation techniques. Procedural ecosystem generation techniques can be divided in two groups: local-to-global and global-to-local. The local-to-global techniques determine the plant positions by simulating plant growth and plant competition. The results of these techniques are hard to predict and to control. Also, these techniques are not generally applicable, because they only have been used so far for small forest areas using simple ecological principles. Global-to-local techniques are able to generate realistic positions of plants by using a combination of Wang tiling and PDD generation techniques. The classification process of the plant locations is not realistic, because plant types are semi-randomly assigned and the generated patterns are limited despite the usage of neighborhood kernels. Further, it is not possible to use information provided by data sources correctly for the classification. The information is not mapped correctly, because the input information is used as probabilities instead of ground-truth. A correct mapping means that if at a certain height plant type A has 60 percent coverage than in the final distribution plant type A should have 60 percent coverage at that height. This cannot be achieved with the current procedural techniques. In short, procedural ecosystem generation techniques are able to generate realistic point positions when no prior information is available of where points are located, but the classification process is not realistic.

The last group of techniques that was investigated are the ecological models. These were investigated, because they are able to generate data for areas that do not yet exist. There were two types of models investigated: dynamic and neutral models. Dynamic ecological models are able to generate maps with information about coverage, biomass and height for future areas. Still these maps need to be translated to actual position point for plants. One type of dynamic model is able to generate plant position data for future areas. However, these models are not commonly available and are often limited that only one type is modeled. Neutral models generates patterns on a grid for different plant types based on a global coverage value and shape metric for each plant type. Some of these models are able to match the input coverage value correctly with the output coverage for each plant type on the grid, while generating realistic patterns based on the shape metric input. The main disadvantage of these models is that they cannot be used in combination with data input of real environments, because the coverage parameter for a single plant type must be static. Also, patterns for a single plant type are all approximately the same. It is not possible to have larger or smaller patterns based on a certain location. Finally, this process is applied on a grid resulting in a LCC map and there it is not possible to obtain actual point positions.

Table 2.1: Summary of the discussed techniques



Field	Type	Advantages	Disadvantages
Geo-domain	Plant detection techniques	– Able to detect exact position of trees in dense existing areas	– Limited to large contrasting objects in the map (e.g. trees) – Can only be applied to existing areas – Requires high-resolution data
Procedural ecosystem generation	Local-to-global	– Generate position data for individuals based on a simulation.	– Difficult to control the simulation with information from spatial data – Lacks realism, no advanced usage of ecological principles – Simulations are often focused on small forest areas.
	Global-to-local	– Generate large-scale position data for plants by taking into account plant sizes – Model plant interactions	– Placement of different plant types is semi-random. It does not allow for logical use of spatial data sources. – Generation of patterns is approximated with plant influences, but difficult to control
Ecology	Dynamic landscape models	– Generates maps with similar information that could be obtained from remote sensing techniques – Maps represent information of future areas	– Does not often generate position data for plants
	Neutral landscape models	– Generate patterns based on static composition and shape metrics for a grid	– Neutral means that all parameters are global defined for an environment, e.g. does not work with spatial data.

2.5. PRELIMINARY FRAMEWORK

The purpose of this research is to develop an algorithm that is able to generate realistic plant distribution for existing and future areas. Spatial data is required to achieve this, because it provides essential information about the area for which a plant distribution is generated. The firs t problem is that no spatial data is available for future areas, because remote sensing techniques cannot be applied to areas that do not yet exist. In this chapter, it is shown that ecological models are able to generate spatial data for future areas. Next, from this spatial data positions of plant have to be determined with their appropriate plant type. As is shown with the plant detection techniques it is often not possible to detect plant positions of small plants. Also, for future areas this is not possible at all. However, in the procedural ecosystem generation section it is shown that there are point generation technique that are able to approximate positions when limited data is available. These points still need to be assigned a plant type based on the spatial data. This is not possible with the current procedural techniques. Techniques from neutral modeling are able to generate a grid map where each tile is occupied by a plant type based on the coverage and shape metric values for each plant type. In general, the input values are correctly mapped to the output map. Disadvantage is that the techniques from neutral modeling only work with a static value for both the coverage and shape metric of each plant type. In spatial data cases these values are not static. Based on this overview, the following framework of the plant placement algorithm is proposed:

First, spatial data input is collected that represents information about the environment for which the plant distribution is generated. This data could be generated by remote sensing techniques or ecological models. Second, all possible plant positions are generated with a procedural point generation technique. Third, these points are classified to a plant type based on a technique from neutral modeling. In both the point generation and classification, spatial data should be used as input to give information of where point should be approximately generated and how they should be classified. The techniques from the point generation and classification require some improvements to work correctly in the algorithm. For example, the point generation technique should be able to use existing point data if that is available and the classification technique has to work with the non-static input from spatial data. The preliminary framework is visualized in Figure 2.23.

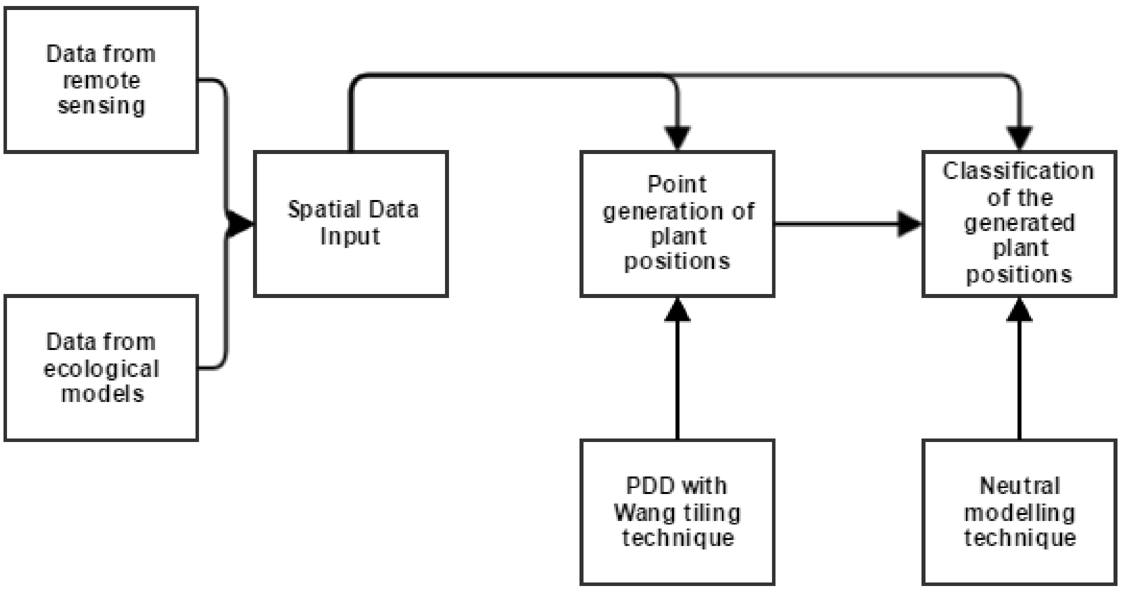

Data from remote sensing
Data from ecological models
Spatial Data Input
Point generation of plant positions
Classification of the generated plant positions
PDD with Wang tiling technique
Neutral modelling technique

Figure 2.23: Preliminary framework

3
Plant placement algorithm

The aim of this chapter is to answer the question of how can spatial data be translated to a point distribution for plants and the question of which types of spatial data can be used. This chapter is divided into three parts to answer these questions. The first part explains the design of the plant placement algorithm. The design part discusses the requirements, system flow, and system parameters of the algorithm. Requirements are about criteria, assumptions, and data input of the algorithm. The system flow gives an overview of the components of the algorithm and their interaction with each other and the data input. System parameters part presents all the parameters used in each component. The second part of this chapter explains the working of each component in detail. The algorithm consists of four components: data processing 1, point generation, data processing 2, point classification. In the final part of this chapter, preliminary results are explainied. The content of this chapter is related to the second, third, and fourth subresearch question.

3.1. Design

This section discusses the design of the plant placement algorithm. It starts with presenting criteria and assumptions of the algorithm. Criteria are properties which the algorithm should meet. These properties are used in the discussion of the validation of the next chapter. The algorithm should meet the following criteria:

- Spatial data-driven: The algorithm should be able to handle and combine different spatial data sources to reconstruct existing or future areas. Data for existing areas is the output from remote sensing techniques, and data for future areas is the output from dynamic ecological models. Input data should be handled logical; meaning that what is used as input should be visible in the output. For example, if a data source puts constraints on where a plant can be placed then this should match with the output.

- Adaptive: The algorithm should be able to work with different plant types and different plant levels. Different plant levels are for example trees and groundcover plants. Also, different types of areas should be supported.

- Effective: The algorithm should be able to generate a realistic plant distribution for existing and future areas. A distribution is realistic when the composition and pattern formation is similar as for existing plant distributions for the same types of area.

The choice for these criteria is based on the research objectives defined in the introduction chapter. To generate a realistic plant distribution of existing or future areas, related spatial data sources are necessary, because these sources contain real-world information of the appearance of the plant distribution for these areas. Therefore, the algorithm should process data sources that directly influences the output of a plant distribution. The algorithm should be adaptive, because it has to work with a wide variety of natural environments.

Assumptions of the algorithm specify which conditions are made about the areas that are used by the algorithm to generate a realistic plant distribution. The algorithm has the following assumptions:

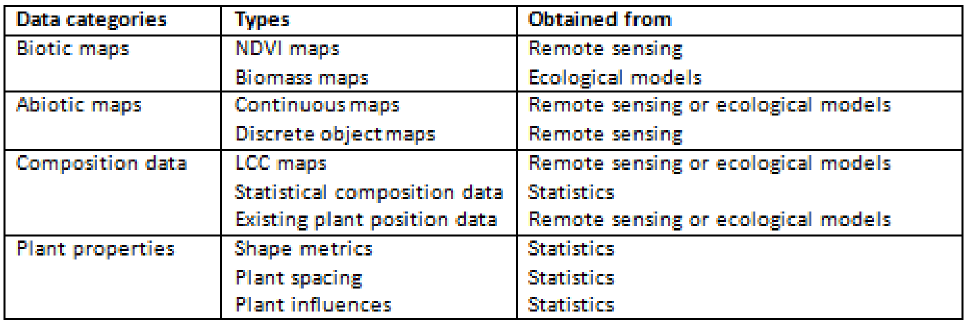
- Plant patterns are fractal in nature

- Plant types in the same plant level have approximately the same plant size, e.g. having equal spacing between the plants
- Direct plant influences, like in paragraph 3.1.4 only occurs by a plant level on its lower plant levels. For example, trees on the groundcover plants.

3.1.1. DATA INPUT

This subsection presents the data input for the plant placement algorithm.. In the next sections, it is discussed how these different data sources are processed by the algorithm. The data input can be in the form of maps generated by either remote sensing techniques or dynamic ecological models, or statistics. Further, the data input can be grouped into four categories: biotic maps, abiotic maps, composition data, and plant properties. Table 3.1 summarizes these categories with subtypes and how these subtypes can be obtained.

Table 3.1: Overview of the data input that can be used for the plant placement algorithm



Data categories	Types	Obtained from
Biotic maps	NDVI maps	Remote sensing
	Biomass maps	Ecological models
Abiotic maps	Continuous maps	Remote sensing or ecological models
	Discrete object maps	Remote sensing
Composition data	LCC maps	Remote sensing or ecological models
	Statistical composition data	Statistics
	Existing plant position data	Remote sensing or ecological models
Plant properties	Shape metrics	Statistics
	Plant spacing	Statistics
	Plant influences	Statistics

This is a general overview of the different kinds of data that can be used. For example, many spatial maps fall under the category of abiotic maps, because they represent information like height or soil, which is often available. The next paragraphs discuss each data category in more detail.

BIOTIC MAPS

Biotic maps represents "living things" in the environment. For this algorithm, the use of biotic maps is limited to NDVI and biomass maps. NDVI maps are calculated from aerial images which have the infrared band included. A NDVI value is calculated per pixel of the aerial image with Formula 3.1. NIR is the infrared value and VIS the red band value of the aerial image.

$$NDVI = \frac{(NIR - VIS)}{(NIR + VIS)} \tag{3.1}$$

NDVI values range from -1 to 1. Values close to -1 represent water, close to 0 bare ground, and close to 1 vegetation. However, higher NDVI values do not always mean that there are more plants, because NDVI measures the greenness of the plants. This means that an area where there are plants which have many green leaves will receive a higher NDVI value than an area with plants that has a limited amount of green leaves. Figure 3.1 shows an example aerial image, the corresponding infrared band, and the calculated NDVI values.

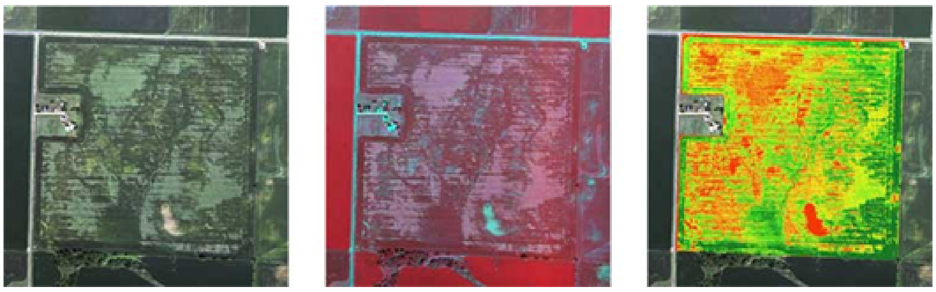


Figure 3.1: aerial to NDVI. First image is the aerial image. The second image includes the infrared band. Third image is the NDVI map. (image from zoomaerial.com.au)

NDVI maps can only be calculated for existing landscapes, because it is not possible to take aerial images of a future area. For future areas , biomass maps can be used to represent approximately the same information as NDVI. Whether biomass information is available depends on the ecological model used. Often, an ecological model generates a map of biomass values or a map with continuous coverage values. Biomass gives information of the amount organisms at a certain location. A higher value indicates more vegetation or larger vegetation (e.g. lots of small plants versus a tree). Figure 3.2 shows a biomass map generated by an ecological model [38].

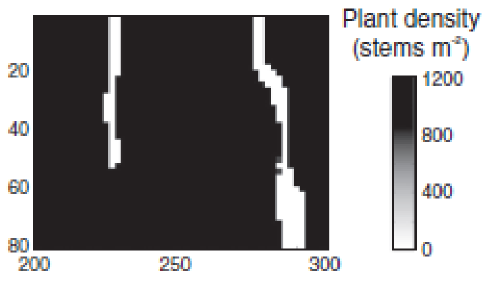


Figure 3.2: Biomass generated by a cellular dynamic model of Temmerman[38].

Biomass maps are normally not available for existing areas, because it is difficult to measure biomass accurately for a large area by remote sensing techniques. In the plant placement algorithm, NDVI and biomass maps have one or two purposes. This depends on the availability and quality of other data sets. The first purpose is to assess where there is vegetation on the map. The second purpose is to give information about the composition in the tile. This is only possible if there is statistical coverage data available that is related to either NDVI or biomass values.

ABIOTIC MAPS

Abiotic maps represent information about an environment of "non-living things". Many maps fall under this category. There are two types of abiotic maps in the plant placement algorithm: continuous abiotic maps (e.g. height or soil maps) and discrete object abiotic maps (e.g. man-made features). Figure 3.3 shows examples of both continuous and discrete abiotic maps. The purpose of including these maps is to apply constraints on the placement of plants. These maps are able to apply constraints if there is statistical composition data available that is related to the information that is represented by the maps. Continuous maps relate statistical composition data by using the value in each tile of the map. For example, when there is height relation with the composition of the plants then the tile value in a height map can be used directly to obtain the composition statistics. Discrete object maps relate statistical data differently than continuous map. This is because a value in a tileof an object map only gives an indication of the presence of an object. Often statistical composition data is related with the distance or orientation to an object. This means that for each potential plant location the distance have to be calculated to a certain object. Then, the distance is used as value to relate statistical data. Abiotic continuous maps are available for both existing and future areas. Discrete object maps are often only available for modelling existing area, because ecological models cannot generate object maps. But, the user could create a discrete object map himself to investigate the effect of an object in a future area. For existing areas, various geo-object detection techniques can be used to detect for example roads or houses [40], or detection of rivers [41].

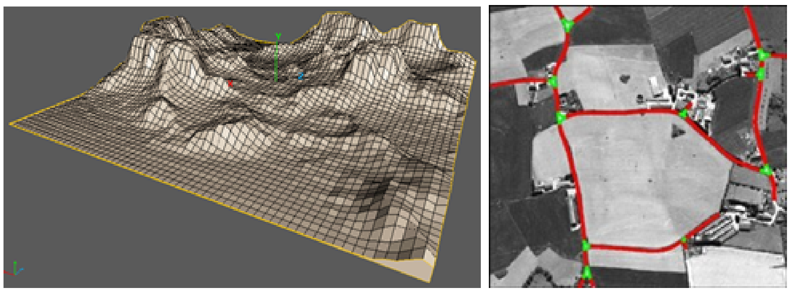


Figure 3.3: Abiotic maps: continuous (height) map and discrete object (road) map (both from Google Image).

Composition data

Composition data represents information which show where specific plant types grow and how much they occur at certain places in an environment. This kind of information can be derived from several data sources: discrete and continuous LCC, statistical data, and existing point data of plants. In most cases, discrete LCC maps give coverage information not about specific plant types, but about group of plant types called ecotypes. Ecotypes can for example be a forest, wetland, or lowland. Thus, an ecotype is set of plant types and normally does not provide additional information about the composition within an ecotype. Another disadvantage of discrete LCC maps is that there are strict borders between each category on the map. Continuous LCC maps do not have this disadvantage, because for each tile on the map they give a probability of how likely a certain ecotype occurs. For continuous maps, it is also more often the case that individual plant types are represented instead of ecotypes in discrete maps. Continuous LCC maps are available for both existing and future areas, but when generated by an ecological model the map often only contains a single category instead of multiple categories for regular LCC maps. Further, discrete maps are more commonly available then continuous LCC maps. Besides providing information about the composition of an area, LCC maps are also able to give information of the presence of vegetation in general. Figure 3.4 shows an example of both discrete and continuous LCC maps.

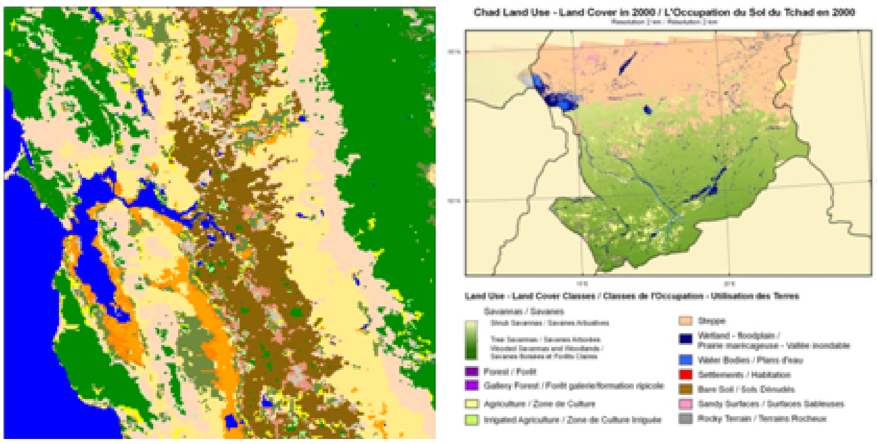

Chad Land Use - Land Cover in 2000 / L'Occupation du Sol du Tchad en 2000
Land Use - Land Cover Classes / Classes de l'Occupation - Utilisation des Terres
Savannas / Savanes
Shrub Savannas / Savanes Arbustives
Tree Savannas / Savanes Arborées
Wooded Savannas and Woodlands / Savanes Boisées et Forêts Claires
Forest / Forêt
Gallery Forest / Forêt galerie/formation ripicole
Agriculture / Zone de Culture
Irrigated Agriculture / Zone de Culture Irriguée
Steppe
Wetland - floodplain / Prairie marécageuse - Vallée inondable
Water Bodies / Plans d'eau
Settlements / Habitation
Bare Soil / Sols Dénudés
Sandy Surfaces / Surfaces Sableuses
Rocky Terrain / Terrains Rocheux

Figure 3.4: discrete and continuous LCC maps (images from usgs.gov).

Statistical coverage data is another source for composition data. To use the statistical data, it must be related spatial maps based on either abiotic or biotic factors. Figure 3.5 shows an example where statistical composition data is related to the height of a terrain. Thus, in that case statistical data must be related to a height map of the target area. It is important to note that often this statistical data is calculated from similar areas as the target area. This means that it is possible that the statistical data does not always lead to satisfactory results, which will be shown in the next chapter during the test phase.

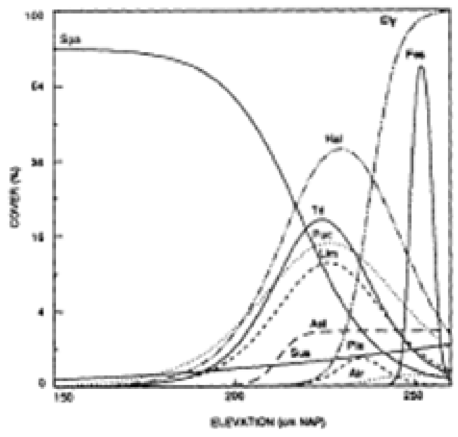


Figure 3.5: Statistical composition data related to the height of an area [42].

The last source for composition data are existing plant point positions. This data source is slightly different in comparison with the other composition data sources. Instead of providing discrete or continuous composition values on a raster map, the point data gives exact position of plants with their plant type. There are two cases these of how these point positions can be processed. In the first case when all the positions are known for every plant type, then the input point data could be used directly for a 3D visualization. In the second case only point data for a subset of all plant types is available. Then, the plant types for which the point data is available can be removed from the point classification part of the plant placement algorithm. The input of existing positions is possible for both modelling of existing and future areas. Such data could be obtained through reconstruction/remote sensing techniques or field measurements for existing areas, or dynamic individual ecological models for future areas. Figure 3.6 shows examples of point data obtained through remote sensing techniques.

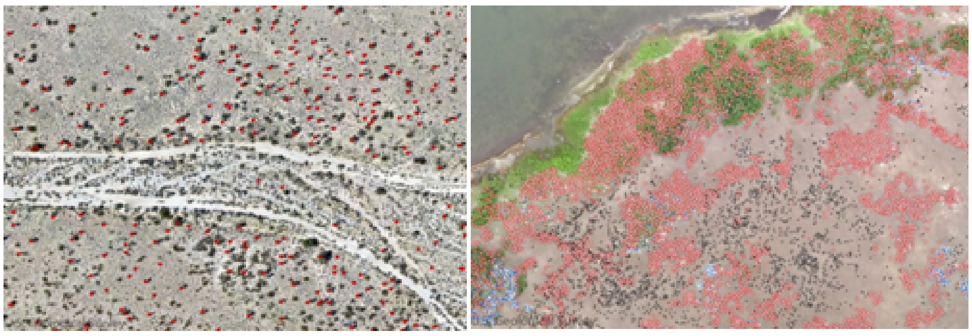


Figure 3.6: Existing point data (image from usgs.gov).

The availability of composition data is vital for the working of the algorithm. It is necessary to have composition data for every single plant type that is going to be placed by the algorithm. The only exception is when all the positions are already known by the input of existing points. In practice, this is often not the case.

PLANT TYPE PROPERTIES

Plant type properties data describe characteristics for each plant type. Three types of data fall under this category in this algorithm: shape metrics, plant spacing and plant influences. This data is normally obtained from fieldwork or previous studies. Shape metrics describe the shape of patterns of plants. Plants can grow in different patterns, e.g. certain plants grow scattered throughout an area, while others plants grow very aggregated. Figure 3.7 shows examples of different patterns of plants.

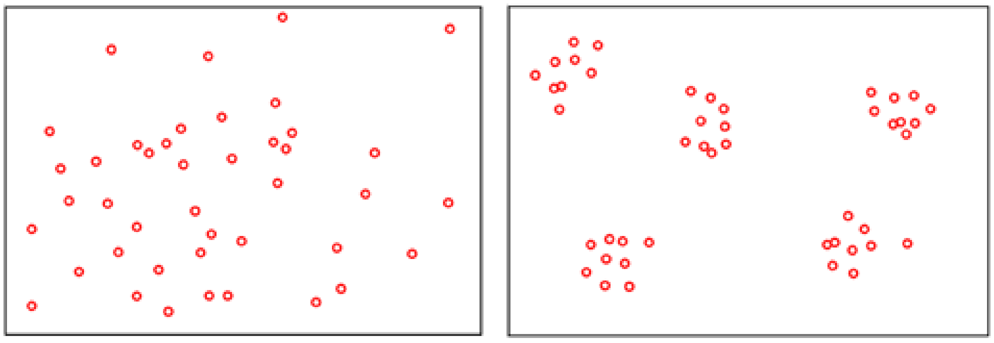


Figure 3.7: Scattered and aggregated point patterns (image from wikimedia.org).

The input of shape metrics is important for the algorithm, because they define the patterns of the plants. In the plant placement algorithm, patterns of the plants are defined in terms of size and roughness. The size of patterns is based on shape metrics that give information about the aggregation or scattering of the plants. The roughness of patterns is represented by the fractal dimension or Hurst metrics. Hurst and fractal dimension are similar and can be calculated from each other. In Figure 3.8, the principle of the Hurst metric is shown. A larger Hurst means a less rough surface, and a low Hurst a rough surface.

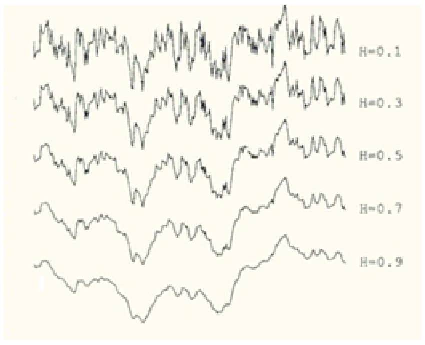


Figure 3.8: Hurst (Google Image).

The next plant property is plant spacing which defines how much space there should be between each plant. The spacing is determined from the size of the plants for each plant type. The third plant property is the influence of plant types on each other. Plant influences can be used to model interactions between different plant types. Figure 3.9 shows an example of how a plant could influences its surroundings.

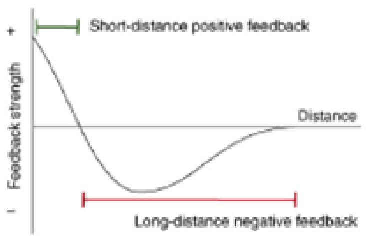


Figure 3.9: Plant interaction [43]. Depending on the distance, a plant can give positive or negative feedback to other plants. Feedback is the influence on other plants. In this example, plants are promoted to grow close to each other.

3.1.2. SYSTEM FLOW

algorithm. The aim of the algorithm is to translate several input data sources to a classified point distribution where each class in the classification represents a unique plant type.

The algorithm is divided into four components: data processing 1, point generation, data processing 2, and point classification. Figure 3.10 shows these components with including the general flow of the algorithm. Two data processing components are used, because different data input has to be processed before the point generation and classification component. In the first data processing component, data is processed to determine the presence of vegetation. In the second processing component, composition and shape metrics data is combined with the generated point distribution. Figure 3.11 shows a detailed system flow chart of all the component which visualizes the effect of each data input source on the process of the algorithm. The next paragraphs give a short introduction of each component. The next sections explains the components in more detail.

The data processing 1 component is where biotic and LCC maps are processed to determine the presence of vegetation. In addition, the component calculates the grid resolution for the input map for the point generation component based on the spacing between plants. The last step of this component is to generate a map with information about the presence of vegetation having the calculated grid resolution. More details about this process and examples are found in chapter 3.2.1.

The point generation component is where all possible plant locations are generated based on the input map generated by the first data processing component. These point locations are not assigned to any plant type in this component. The plant locations are predefined, because the location data, e.g. the coordinates, are necessary to connect information of maps and to generate information about the patterns of each plant type. The connection of this information before assigning any plant type gives the possibility to correctly map the data input to a final plant distribution. Points are generated through a method called PDD with Wang tiles. The same method to generate points is used by several global-to-local techniques discussed in 2.2. The method in this components includes several extensions to handle for example existing plant point

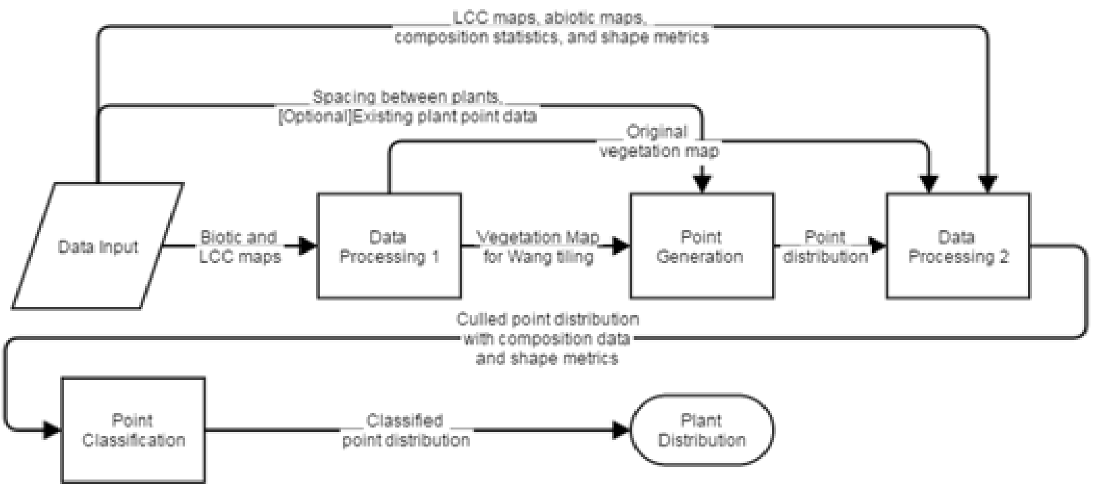

LCC maps, abiotic maps, composition statistics, and shape metrics
Spacing between plants, [Optional]Existing plant point data
Original vegetation map
Data Input
Biotic and LCC maps
Data Processing 1
Vegetation Map for Wang tiling
Point Generation
Point distribution
Data Processing 2
Culled point distribution with composition data and shape metrics
Point Classification
Classified point distribution
Plant Distribution

Figure 3.10: General system flow

data. Detailed explanation of this component work is given in chapter 3.2.2.

The second data processing component is where composition data is combined with the point distribution of the point generation component. At the end of this component each point receives a composition value for each plant type. Further, during the process of this component points are removed from the distribution if they cannot be assigned to any plant type based on the input data. The same procedure is applied to assign shape metric information to each point. The detailed procedure is explained in chapter 3.2.3.

The point classification component is where processed point distribution from the second data processing component is classified; each point is assigned a plant type or nothing. The classification is based on the method of fractal neutral landscape modelling discussed in chapter 2.2. The point classification component consists of two steps: point fractal generation and classification. The point fractal generation uses a fractal algorithm which is able to generate fractal values for points and handle shape metrics input. The classification process translates for each point its fractal value and composition data to one of the plant type or nothing. This component procedure is explained in detail in chapter 3.2.4.

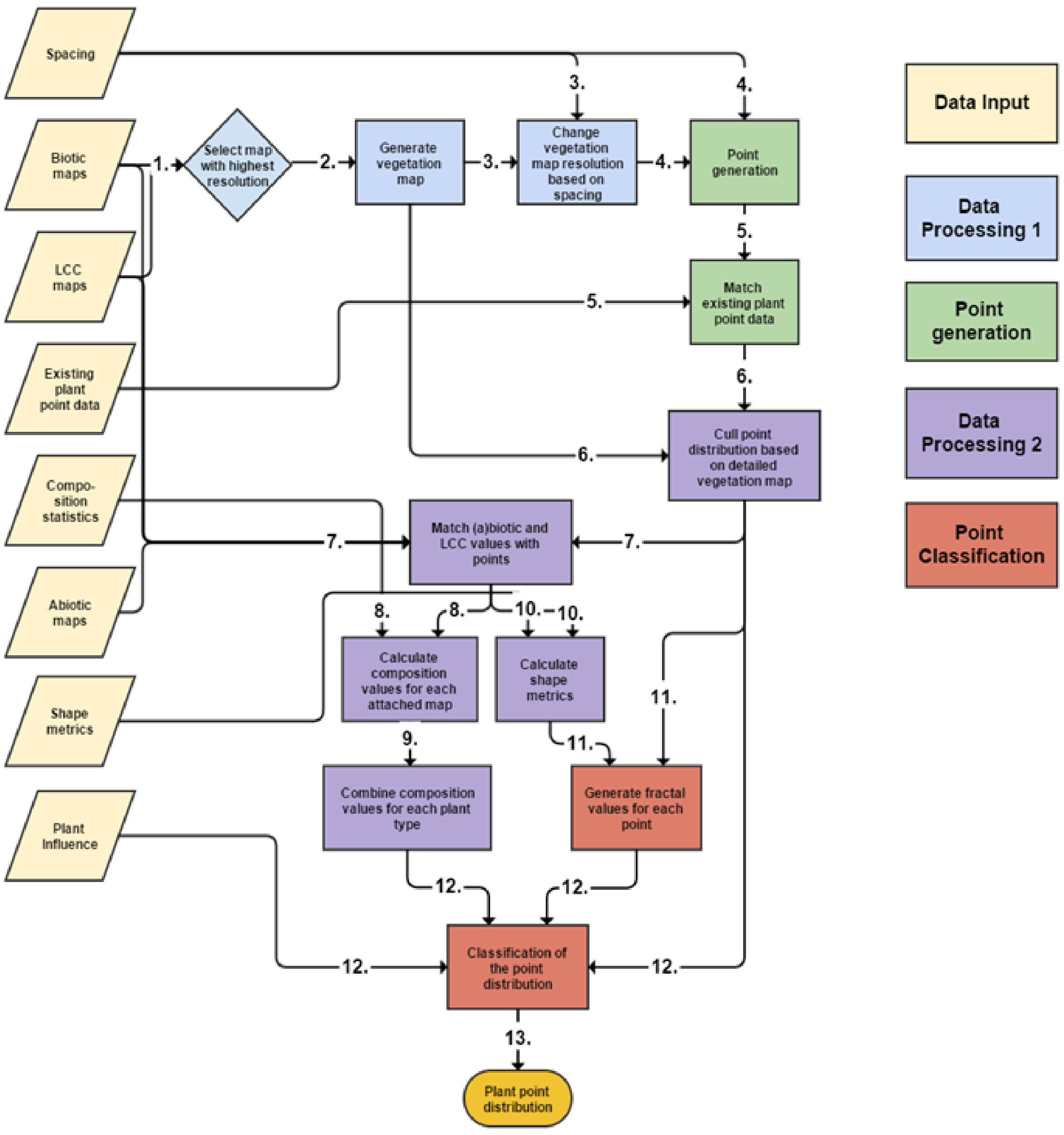

Spacing
Biotic maps
LCC maps
Existing plant point data
Compo-sition statistics
Abiotic maps
Shape metrics
Plant Influence
Select map with highest resolution
Generate vegetation map
Change vegetation map resolution based on spacing
Point generation
Match existing plant point data
Cull point distribution based on detailed vegetation map
Match (a)biotic and LCC values with points
Calculate composition values for each attached map
Calculate shape metrics
Combine composition values for each plant type
Generate fractal values for each point
Classification of the point distribution
Plant point distribution
Data Input
Data Processing 1
Point generation
Data Processing 2
Point Classification

Figure 3.11: Detailed system flow. Numbers represent the order of the system flow.

3.1.3. SYSTEM PARAMETERS

This subsection gives an overview of the parameters or important terms used in each component of the algorithm. Each parameter is shortly explained per component.

DATA PROCESSING 1

Vegetation map is a grid map that provide information of where vegetation is located. The vegetation map is used as input for the point generation component. This map is calculated from a input map with information about coverage, biomass or NDVI

Vegetation threshold is the number that is used to threshold a NDVI, biomass, coverage or LCC map to obtain a high-resolution vegetation map. A lower value results normally in a map with more vegetation. A higher value vice versa. The user sets the threshold value.

POINT GENERATION

Number of Wang corners is a number that defines the number of Wang corner tiles used during the Wang tile generation. A lower value means that Wang corners are repeated more often over the map which results in a point distribution where repetition of points is easer detectable. A higher value means that there is less repetition. Each time the number increases, the point generation becomes slower, because more Wang corners tiles have to be calculated. In the algorithm this number is set at 4.

DATA PROCESSING 2

The second data processing component does not use parameters specific for this component.

POINT CLASSIFICATION

Lacunarity is a number that defines the new frequency value for each higher octave in the fractal algorithm. This value is set at 1.87.

Octave is a Simplex noise layer. The number of octaves that is used to calculate fractal noise is calculated by using the Lacunarity and frequency value.

Frequency is a number that defines the correlation of the Simplex noise field. A low value gives a high correlation and a high gives a low correlation. A new frequency value is calculated for each octave used to generate fractal noise. The start frequency value is defined by the fractal area shape metric input value.

Amplitude is a number that defines the roughness of the fractal value. The amplitude is doubled for each new octave used to generate a single fractal noise field. The start amplitude value is defined by the roughness shape metric input value.

Scale factor is a number that is multiplied with the start frequency value at the beginning of generating fractal noise. The value of this number is calculated by using the average number of points per tile.

UNIVERSAL PARAMETERS

Plant level is a group of plant types that have approximately the same size. In the algorithm plant levels are processed sequentially starting with the largest plant level. The number of plant levels depends on the user.

Plant spacing is a number that defines the amount of space that each point should have from each other. Each plant level has its own plant spacing value. The spacing value depends on the size of the plant type in the target plant level.

Composition/coverage are often intertwined in this thesis report. They are a value that defines the amount of occupation for each plant type on the map. These numbers are calculated from spatial data sources in combination with statistical composition data.

Fractal area (shape metric) is a value that defines the size for each plant type. Each plant type receives its own value. These number are based on the input data from the user.

Roughness (shape metric) is a value that the roughness of the patterns for each plant type. Each plant types receives its own roughness value. These values can be dependent on spatial data sources

3.2. SYSTEM COMPONENTS

3.2.1. DATA PROCESSING 1

The purpose of the first data processing component is to process the input data for the point generation component. During this component two maps are generated from the input data: high-resolution presence-based vegetation map and low-resolution presence-based vegetation map. A presence-based vegetation map is a grid map that indicates which grid cell contains vegetation and which does not. From now on the presence-based vegetation map is referred to as vegetation map. The low-resolution vegetation map is used as input for the point generation component. The high-resolution vegetation map is input for the second data processing component. Two different resolution are necessary, because the point generation component requires a vegetation map where the size of the tiles are at least three time larger than the minimum distance necessary between the largest plants of the environment. Otherwise, the Wang tiling in the point generation will not work correctly, because overlaps will occur during the Wang tiling phase. If the high-resolution map already meets the required distance than the low-resolution map has the same resolution as the high-resolution map. This data processing component has the following input, and output:

Input:

- Biotic map
- LCC map
- Point spacing

Output:

- High-resolution presence-based vegetation map
- Low-resolution presence-based vegetation map

HIGH-RESOLUTION VEGETATION MAP

First, the high-resolution vegetation map is generated in this component. This map is created based on the input collection of biotic and LCC maps. At least one map belonging to these categories should be available. When multiple maps are given as input then the map is selected with the resolution, because it contains the most details. For each tile on the map the threshold parameter is applied, which means that all values above the threshold are marked as tiles that contains vegetation. Tiles that do not meet the threshold are marked as a tile that does not contain vegetation. The threshold is a continuous number in the case of a biotic map or in the case of LCC data the threshold is the absence of a category for a tile. This process is shown in pseudocode 1 and a visual example is given in figure 3.12.

Algorithm 1 Process of generating a high-resolution vegetation map

```
1: procedure GENERATEVEGETATIONGRID(grid, threshold)
2:     $vegetationGrid$
3:     for all $tile$ in $grid$ do
4:         if $tile > thresh$ then
5:             $vegetationGrid[tile] \leftarrow 1$ ‘
6:         else
7:             $vegetationGrid[tile] \leftarrow 0$
       return vegetationGrid
```

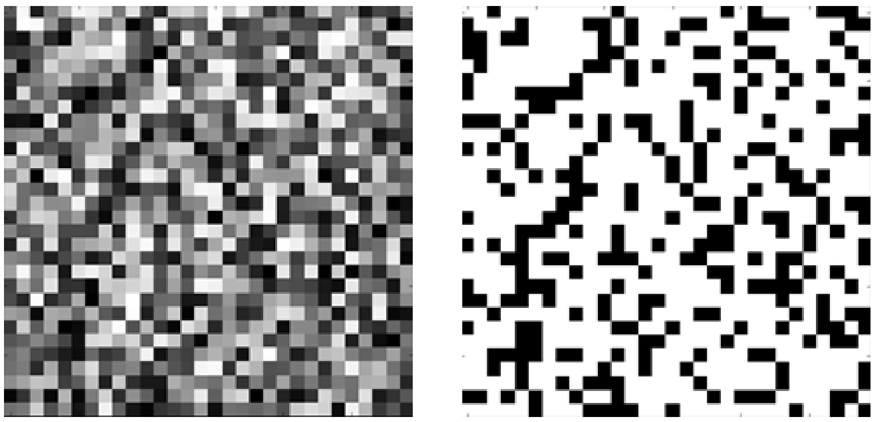


Figure 3.12: The image on the left represents hypothetical NDVI values ranging from 0 to 1. The threshold is set at .3. After thresholding the image on the right is obtained where the white pixels represent a tile with vegetation and black pixels tiles with no vegetation

LOW-RESOLUTION VEGETATION MAP

The low-resolution vegetation map is generated from the high-resolution map. The resolution of low-resolution map depends on required minimum distance between the largest plants. The input parameter "point spacing" represents this number. The resolution should be at least three higher than the "point spacing" number. If the high-resolution map meets this condition then resolution for both the high-resolution and low resolution map is the same. When the condition is not met, the resolution of the high-resolution map is doubled until the condition does. Doubling the resolution means that a group of four tiles are merged into a single tile. The last step is to calculate a value for each merged tile in the low-resolution map. Each tile is assigned as containing vegetation when at least one of the tiles from the high-resolution map belonging to the merged tile is marked as having vegetation. Otherwise, the merged tile is assigned a value indicating no presence of vegetation. This process is shown in figure 3.13.

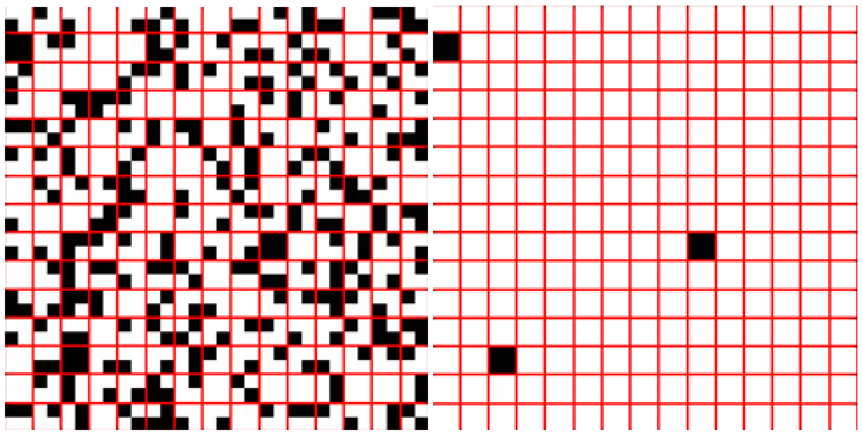


Figure 3.13: The image on the left shows the detailed vegetation map and the red lines show the resolution of the low-resolution vegetation map. The low-resolution vegetation map is shown on the right. A tile is marked black in the low-resolution vegetation map when all tiles from the high-resolution vegetation map are marked black. Black is no vegetation.

3.2.2. POINT GENERATION

The point generation component is where all possible plant locations are generated. The purpose of this component is to generate points on the low-resolution vegetation map so that a point distribution is obtained that represents all possible plant point locations by taking into account the point spacing input parameter. Points are generated by using the PDD with Wang tiling method as discussed in chapter 2. To generate plant positions properly, the PDD with Wang tiling method is modified in three ways. First, Wang corner tiling is used instead of Wang border tiling to avoid the corner problem [31]. Second, non-biased multiclass sampling is introduced instead of giving points a fixed size. Third, the use of existing plant point data is included. In this subsection, each of these modifications is discussed separately. This subsections starts with explaining PDD generation for this component. Next, the three modifications are discussed. This component uses the following input and output:

Input:

- Low-resolution presence-based vegetation map
- Point spacing
- Existing point data

Output:

- Point distribution

PDD GENERATION

PDD generation is used to generate points in this algorithm, because it places points with at least a minimum distance from each other. This distance is defined in the plant placement algorithm by the "point spacing" input. The process of creating a PDD point distribution is shown in pseudocode 2. The output of the method is shown in figure 3.14.

Algorithm 2

```
1:  function GENERATEPDD(spacing, area, existingPoints)
2:      $pointsPDD$
3:      if $isEmpty(existingPointa)$ then
4:          $pointsPDD[0] \leftarrow randomPoint(area.x, area.y)$
5:      else
6:          $pointsPDD \leftarrow existingPoints$
7:      $randomPointsList \leftarrow generateRandomPoints(area)$
8:      for $rp$ in $RandomPointsList$ do
9:          $distances \leftarrow calculateDistances(rp, pointsPDD)$
10:         if $min(distances) > spacing$ then
11:             $pointsPDD.add(rp)$
12:     return $pointsPDD$
```

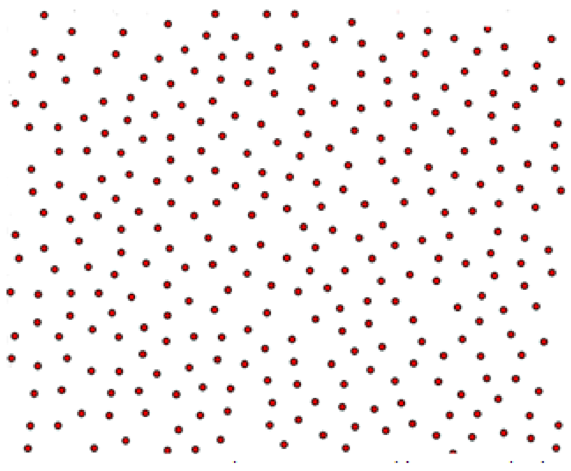


Figure 3.14: An example PDD generated by the method of this research

CORNER-BASED WANG TILING

Generating a point distribution with only the PDD method is not feasible, because it can become very slow. Therefore, the Wang tiling method is adopted which generates PDDs for a fixed amount of tiles. The generated PDDs are repeated over the map and stitched together to form one large seamless point distribution. The obtained point distribution has the same properties as the smaller PDDs. To create a Wang tiling in this component, the corner-based Wang tiling method is adopted [31]. This method starts with assigning random colors to each corner. The number of colors used is defined in the algorithm by the parameter "Number of corners". An example result is shown in Figure 3.15 overlaid on the low-resolution vegetation map. The gray blocks represent borders between each tile.

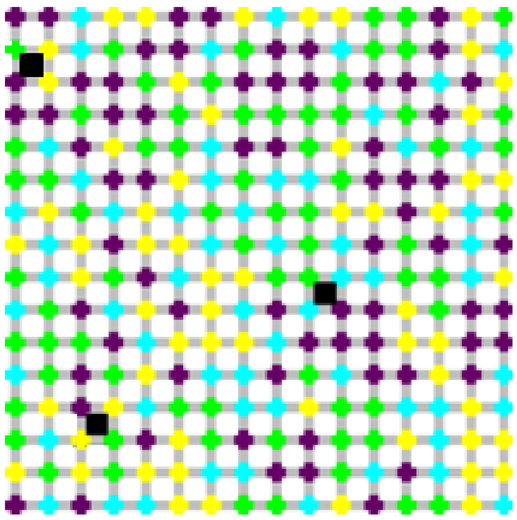


Figure 3.15: An example PDD generated by the method of this research

In the example corner Wang tiling of the figure there are four unique Wang corners (e.g. unique colors). A border is the area between two Wang corners, e.g. the gray area between two corners. The number of unique borders can be calculated from formula 3.2 represented with the variable $nBorders$. $nCorners$ is the number of Wang corners.

$$nBorders = nCorners^2 \tag{3.2}$$

A Wang tile is the area between four Wang Corners. This means that the number of unique Wang tiles is dependent on the number of Wang corners. This number can be calculated from formula 3.3. $nTiles$ is the number of unique Wang tiles.

$$nTiles = nCorners^4 \tag{3.3}$$

To generate a seamless point distribution with PDD properties, first each unique Wang corner is filled points using the PDD generation algorithm. Figure 3.16 shows two corners filled with points. The next step is to fill each unique border with points. This is done by taking into account the neighboring Wang corners of each unique border. This is shown in the second example of figure 3.16. This makes the borders and corners seamlessly to each other.

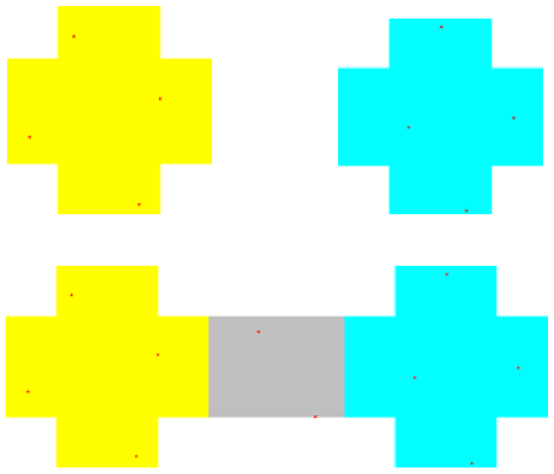


Figure 3.16: Each colored corner tile is filled by the PDD generation method with points. Each unique border is filled with points by the PDD generation method by taking into account its neighboring corners.

The third step is to fill each unique Wang tile. This process is similar as for filling the borders. To fill each Wang tile, the neighboring corners and borders are included to obtain a seamless transition between the tile, border and corners.

The final step is to fill in the Wang tiling created. The result is shown in figure 3.15. The tiles match with the tiles of the low-resolution vegetation map. Non-vegetated tiles are ignored by the algorithm. Thus, no points are placed in these tiles.

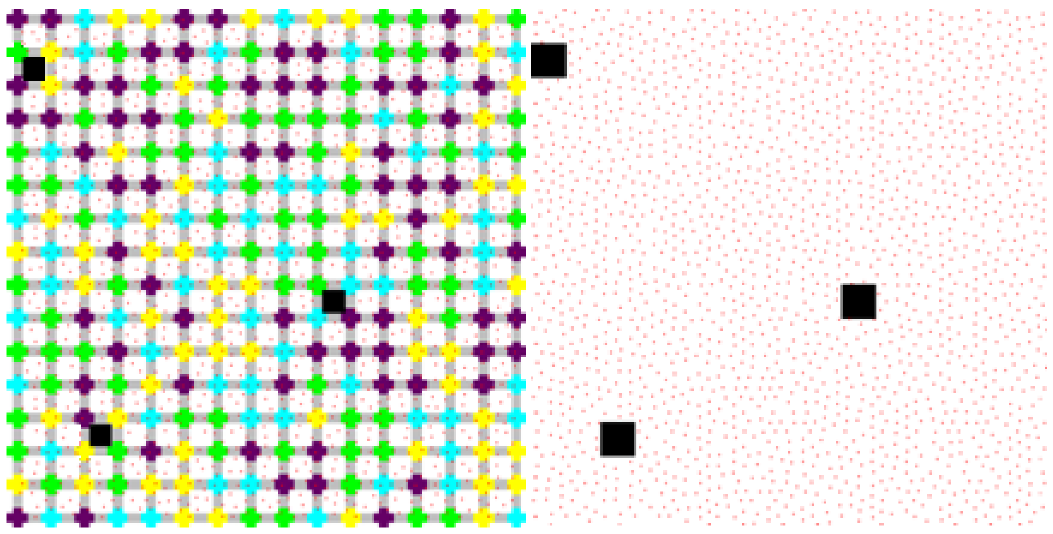


Figure 3.17: Seamless point distribution is obtained by putting all the unique corners, borders, and tiles together based on the Wang tiling. On the right the Wang tiling colors are removed and a seamless point distribution is visible. Places that are marked as no vegetation do not have points included.

NON-BIASED MULTICLASS POINTS

In the previous paragraphs PDDs were generated where all points have the same minimum distance from each other, e.g. they have the same "point spacing". This is often not true in real environments, because plants can have different sizes. It is possible to assign different sizes to the point and therefore have different spacing between points, but this gives a bias to a certain plant type for each point. The bias is created, because assigning a size to each random would be random, because in this component no information is available about the composition or patterns of each plant type. This means that points with certain size could be generated on locations where it is not possible according to the data.

To solve this issue in the placement algorithm, plant types are grouped into classes based on the size of the plants. Each class contains plant types that are approximately the same size. In the algorithm each class is also called a different plant level. The creation of the point distribution starts with the generation of the point locations for the plant types in the highest plant level, e.g. the plant types that have the largest size or spacing. This step is followed with the generation of points for the second highest plant level until all plant levels have generated their points. The same Wang tiling is used for each plant level. Corners, borders and tiles are reused. During the point generation, each plant level, except the highest plant level, has to generate points in an existing point distribution. The existing point distribution consists of points generated by the higher plant levels than the current plant level. Points generated in the current plant level have the same spacing to the existing points as to each other. Figure 3.18 shows the resulting non-biased multiclass distribution. There are two plant levels: blue and red. The two plant levels have different spacing, but the spacing between the plant levels is equal to the spacing of the blue plant level. This means that if plants belonging to the red plant level cannot grow at certain pre-defined locations, it is still possible to use these positions for other plant levels. In the case of the fixed size method, this would not be possible, because the other possible point locations are placed further away based on the original size of the point. Basically, this results in isolated plants.

The non-biased multiclass point generation method allows to generate point locations without creating a bias for a plant type based on the size of the point. A bias would influence the process in the point classification component which means that data is not correctly mapped.

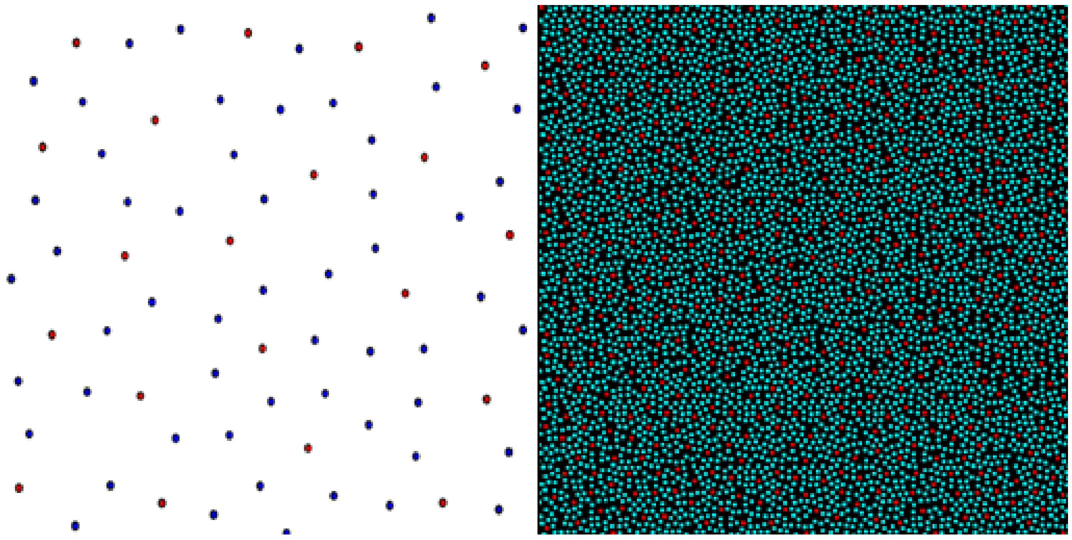


Figure 3.18: The image on the left shows a point distribution based on the non-biased multiclass point generation where on the left the red points all have a distance of .3 to each other and the blue points .1 distance to each other. Also, the non-bias principle is visible where that red points also have .1 distance to the blue points. In the image on the right a seamless tiling is created with the Wang corner tiling method.

EXISTING POINT DATA

The last option in the point generation component is to use existing point data. The usage of existing point data in combination with Wang tiling is not possible. This is because Wang tiling repeats tiles and when existing point data is added then each tile could be different from the other, e.g. no repetition of tiles is possible. Therefore, first a point distribution is generated based on Wang tiling and the existing point data is incorporated after. There are two possibilities to join the existing points with the generated points. Either remove generated points based on the position of the existing points or match the existing points with the generated points. The algorithm uses the second possibility, because if points are removed of the Wang tiling method that are too close to the existing data then the final point distribution will get holes. The disadvantage of point matching is that the positions of existing point are altered. Though, because the point distribution is dense the change in location is small for each point. In addition, the goal is not to exactly reconstruct plant positions, instead the goal of this algorithm is to create a realistic environment that is highly similar in composition and patterns.

There are different possibilities to match the existing points with points of the generated PDD distribution. The point are matched through a greedy approach, because it was the simplest implementation for the amount of time left for this research. The greedy point matching algorithm works by going through the existing point list one by one. For each existing point, it finds the closest point from the PDD distribution. After the closest point is found, the existing point is placed on the position of the point generated by the PDD distribution. Other points can no longer match with that position. Figure 3.17 shows the greedy matching process .

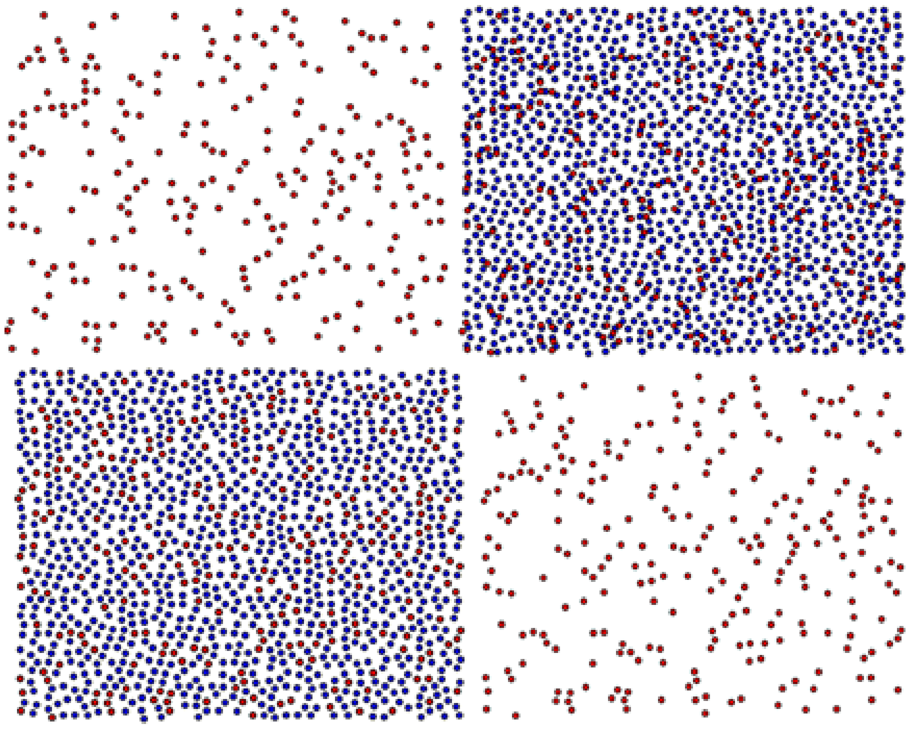


Figure 3.19: Point matching. The top left image is the existing point dataset. In the top right image red points are put over the blue points. For each red point the closest blue point is found bottom left. The last image shows the updated positions for each red point.

3.2.3. DATA PROCESSING 2

The second data processing component is where composition and shape metrics data is combined with the point distribution of the point generation component. In addition, the received point distribution is updated with the high-resolution vegetation map generated in the first data processing component. First, the point distribution is updated with the high-resolution map in this component by removing points. Second, each remaining point receives a composition and shape value for each plant type based on the data input. The component has the following input and output:

Input:

- Point distribution
- High-resolution vegetation map
- LCC data
- Composition statistics
- (A)biotic maps
- Shape metrics

Output:

- Combined composition data attached to each point
- Shape metrics attached to each point
- Updated point distribution

CULLING POINT DISTRIBUTION

The point distribution created in the point generation component was made with the low-resolution vegetation map. This means that the details from high-resolution vegetation map are missing in the point distribution, e.g. there are likely points that are placed on non-vegetated tiles of the high-resolution vegetation map. These details are added by putting the high-resolution map over the point distribution. Points that that belong to a non-vegetated tile are removed. This process is shown in figure 3.20 and explained in pseudocode 3.

Algorithm 3

```
1: function REMOVENONVEGETATIONPOINTS(points, vegetationGrid)
2:     for all $p$ in $points$ do
3:         for all $tile$ in $vegetationGrid$ do
4:             if $tile == 1$ AND $tile.contains(p)$ then
5:                 $points.remove(p)$
6:                 break
7:     return $points$
```

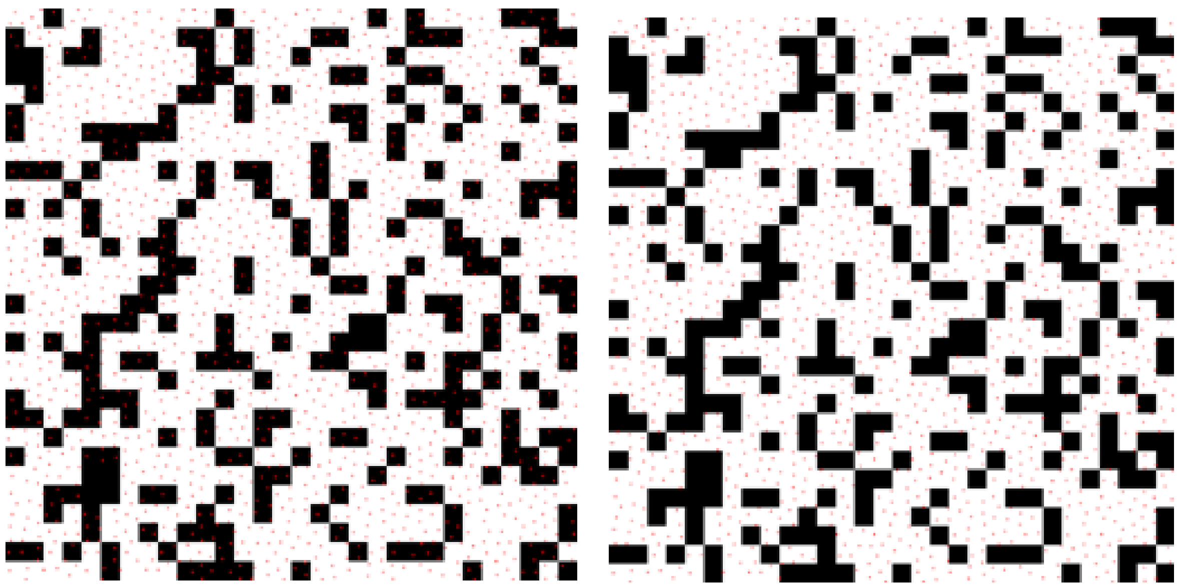


Figure 3.20: The points received from the point generation components, do not match with the high-resolution vegetation map. Therefore, it is necessary to throw away the point are in the black pixels of the detailed vegetation map. The left image overlays the point distribution with the high-resolution vegetation map and on the right all points that were in the black pixels are removed.

COMBINING COMPOSITION AND SHAPE METRIC DATA

The next step in this component is to combine the composition data of the different data sources to generate a single composition value for each plant type for each point. Composition values can be obtained from continuous raster maps that either have coverage values for certain plant types, or (a)biotic values that have a relation with statistical composition values. The second possibility is to have discrete or continuous LCC type maps that give information of where groups of plant types are located. The last possibility is the input from discrete object maps, like road data. Statistical compositions values are related to object maps by the distance factor, for example the distance to a highway.

To combine all composition data, first for each point the information from the (a)biotic or composition maps is retrieved. Each point gets (a)biotic values assigned equal to the number of input maps. In the case of a height map, discrete LCC map and object map, each point receives a height value, LCC type and distance value to the closest object on the object map. The values are based on the coordinates of each point. The second step is to replace these values for statistical composition values for each point using the input statistical composition relations. In the example, this means that each point receives three composition values for each plant type. The last step is to combine all composition values per plant type for each point to a single composition value. these three composition values have to be combined to one composition value for each plant type. This is achieved by taking the minimum composition value for each plant type. This is based on the assumption that the lowest composition value is the limiting factor on the composition of that plant type. A regression function could also be used if the relationship is known between the two factors that represent the composition values. The same procedure is also applied for attaching shape metric information to each point.

3.2.4. POINT CLASSIFICATION

The point classification component is where the points received from the second data processing part are included. Composition and shape metrics values for each plant are connected to the received points. Classification is based on the methods of fractal neutral landscape modelling, but several modifications were introduced to be able to generate patterns that are more complex, handle complex composition data, and multiple plant levels. This component starts with generating fractal values for each point per plant type to be able to generate different patterns for each plant type. The generation of fractal values if based on the shape metric values attached to each point and the position information of the points itself. The second step of this component is to assign a plant type to each point based on the calculated fractal values and composition values. Point classification component has the following input, and output:

Input:

- Point distribution with composition and shape metrics values
- Plant influences

Output:

- Classified point distribution

FRACTAL ALGORITHM

The fractal algorithm in this component generates fractal values for each point per plant type. Fractal values are necessary to be able to generate different patterns for each plant type. The choice for a fractal algorithm was made, because these algorithms generate random correlated or uncorrelated values depending on the input. Using normal random values is not sufficient, because that result in random patterns. Random patterns are not realistic [33]. Fractal values are calculated by using the "fractal Brownian motion" approach with Perlin Simplex noise [44]. Fractal Brownian motion is often applied in the automatic generation of terrain [16]. Detailed explanation of the generation of Simplex noise can found in Gustavson [45]. Simplex noise itself can be controlled by the frequency parameter. A low value means correlated noise and a high value uncorrelated/random noise. Figure 3.21 shows Simplex noise with different frequency parameters.

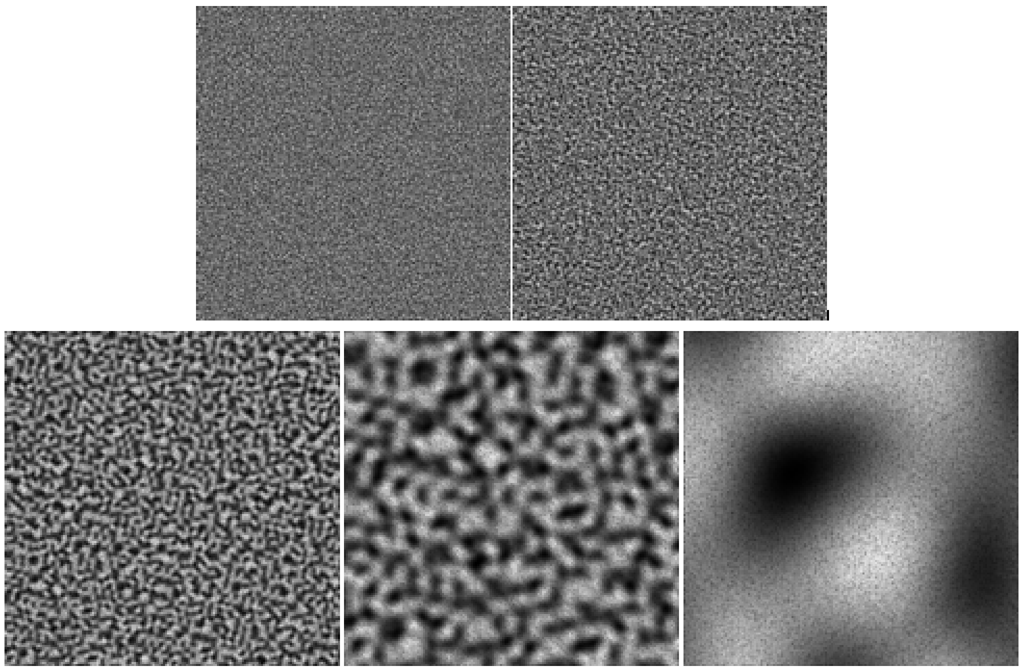


Figure 3.21: Simplex noise with different freq = .1, .3, .5, .7, .9.

Fractal Brownian noise is created by adding layers of Simplex noise with different frequencies to each other. First, the base noise layer is defined with a certain frequency. Subsequent noise layers are added to this map with higher frequency than the previous layers. The value of the frequency of each subsequent layer is controlled by the static parameter called lacunarity. Lacunarity is multiplied with the frequency of the previous noise layer. Further, the contribution of each layer to the final noise map is weighted. The amplitude parameter determines the weight of each layer. Each layer is an octave. The number of octaves (nOctaves) that should be added is calculated with formula 3.4. The idea behind the formula is that fewer octaves are necessary when the base frequency is high, because noise maps with high frequencies are very uncorrelated/random. Summing uncorrelated/random maps do not provide extra details. Figure 4 shows Fractal Brownian noise using the base noise layers from figure 3.21. The general process is described in pseudocode 4.

$$nOctaves = \frac{log(\frac{1}{frequency})}{log(lacunarity)+1} \tag{3.4}$$

Algorithm 4

```
1:  function FRACTALNOISE(x,y)
2:      $lacunarity \leftarrow 1.87$
3:      $totalNoise \leftarrow 0$
4:      $maxAmplitude \leftarrow 0$
5:      $amplitude \leftarrow 1$
6:      $i \leftarrow 0$
7:      while $i < nOctaves$ do
8:          $noise \leftarrow getSimplexNoise(x * frequency, y * frequency) * amplitude$
9:          $totalNoise \leftarrow totalNoise + noise$
10:         $frequency \leftarrow frequency * lacunarity$
11:         $maxAmplitude \leftarrow maxAmplitude + amplitude$
12:         $amplitude \leftarrow lacunarity^{-}i$
13:         $i \leftarrow i+1$
14:     $totalNoise \leftarrow totalNoise/maxAmplitude$
15:     return $totalNoise$
```

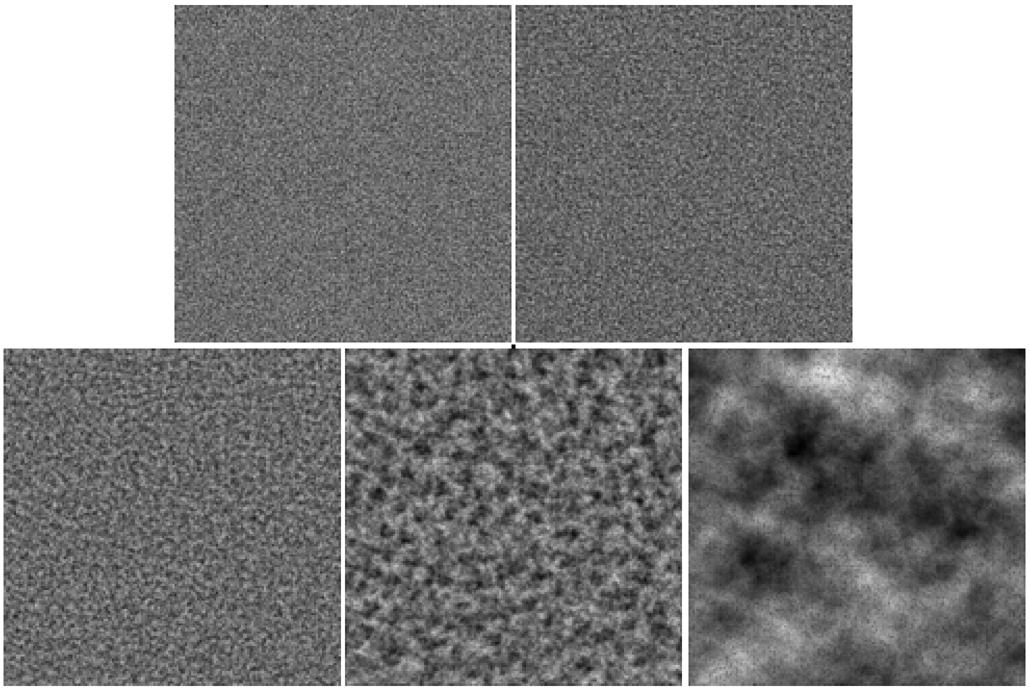


Figure 3.22: Fractal Brownian noise.

The fractal algorithm in described in the pseudocode is the general fractal Brownian noise generator. This method is used in this component, but with one main modification that it can be used in combination with shape metrics data. The shape metric data input consists of two parameters: Hurst and fractal area. Hurst describes the roughness/complexity of the patterns. Fractal area describes the initial size of the patterns. The first parameter that is connected with the fractal algorithm is the Hurst. The Hurst parameter is connected to the amplitude parameter. A high Hurst means that the patterns are not rough/complex, which means that the contribution of each sequential layer is small. On the other hand, a low Hurst indicates that the patterns are rough/complex, so each sequential layer has a large contribution. Formula 3.5 calculates the weights for each layer with the Hurst parameter.

$$amplitude = lacunarity^{-Hurst} \tag{3.5}$$

Figure 3.23 shows the impact of the Hurst parameter on the complexity of the fractal patterns. The first image shows the Hurst values used for each point. The second image shows that the complexity changes following the Hurst values from the first image.

The fractal area is the second parameter that is connected to the fractal algorithm. The fractal area replaces the frequency parameter, because the frequency parameter controls the correlation of the noise and thus the size of the patterns.

For each point, the fractal area is multiplied with a scale factor in formula 3.6. The reason for the scale factor is that fractal noise does not scale with larger number of points for the same area. Basically, it means that patterns become large when more points are added in an area.

$$scaleFactor = \sqrt{\frac{totalPoints}{nTiles}} * 0.5 \tag{3.6}$$

It is possible that the fractal area values differs per point. Thus, the size of the patterns of a plant type is different depending on a certain factor. It is not possible to give these different fractal area values per point directly to the algorithm, because that results in artefacts in the noise map. Therefore, it is necessary to approximate differences in the fractal area parameter per point. This is done by first finding the maximum fractal area value. That value is used to calculate the number of octaves. Then, for each octave the frequency is calculated. For each octave a separate fractal map is generated using the calculated frequency as start frequency. Thus, if there are 8 octaves then 8 fractal maps will be generated. The next step is to use the

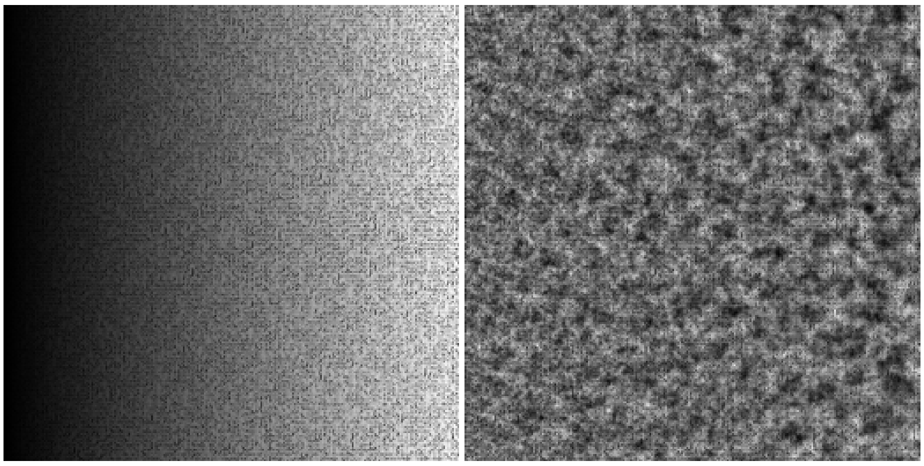


Figure 3.23: Brownian noise with changing Hurst and on the left the gradient field of Hurst. The gradient field starts at .33 and ends with .8613 from left to right. In the image on the right it can be seen that patterns going from complex/rough to less complex from left to right.

fractal area value of each point and calculate the difference with the base frequency of each fractal map. So, the difference between the fractal area value and the base frequency of each fractal map. These differences are used as weights to calculate the fractal value for each point. The fractal value is calculated by adding all the fractal values of each fractal map with the weight value. This means that the fractal maps with a base frequency value that have a small difference with the fractal area value of point, that they are given more weight than the other fractal maps. Thus, these maps contribute more to the final fractal value. This is shown in figure 3.24.

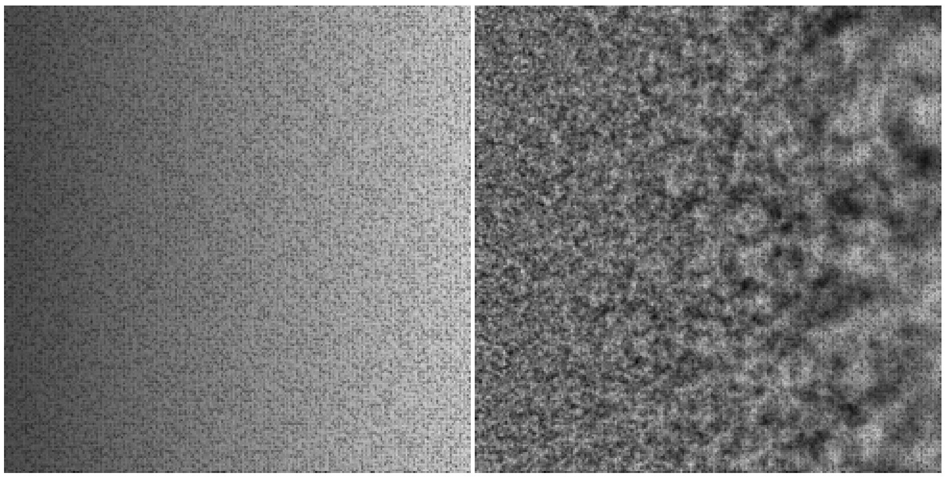


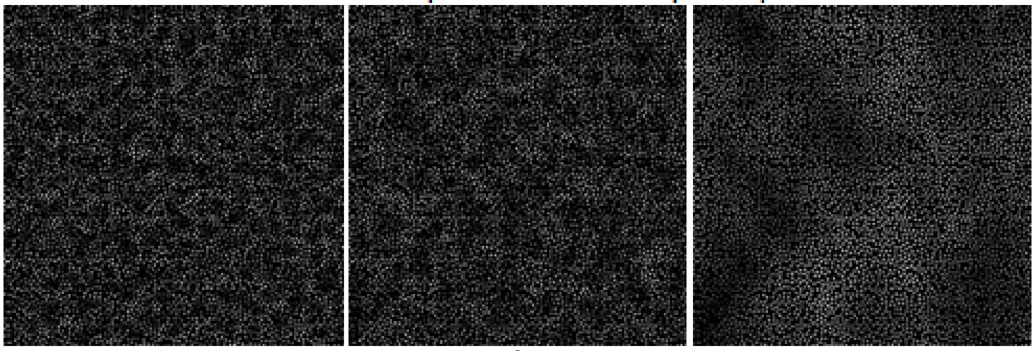
Figure 3.24: Fractal Brownian noise with changing frequency. On the left the image that indicates the values that correspond with the size of the patterns. On the right, the patterns have changing size: noise is more correlated on the right and gradually becomes uncorrelated towards the left side.

Classification process

In this part, the fractal and composition values that are available for each point are used. In addition, plant influences input can be used if plant interaction needs to be modelled. The fractal algorithm generates fractal values for each point equal to the number of plant type in the environment. This means that for each plant type a separate fractal point field is generated based on the shape metric data attached to each point for every plant type and the position coordinates of each point. The classification process is discussed in three parts: static, non-static composition values, and multiple plant levels. In the static composition value part, the composition values for a single plant type are the same for every point location. In the non-static part, compositions values are different for a single plant type. The third part discusses how to classify multiple plant levels.

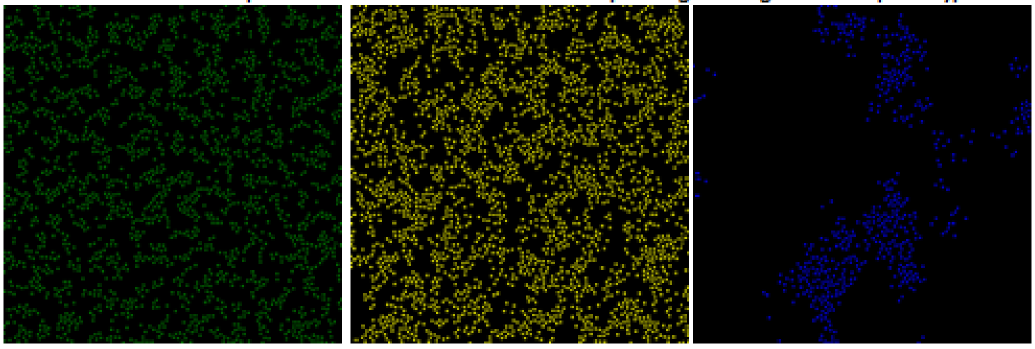
Static composition values mean that the coverage values are the same for a single plant type for the whole environment. Thus, each point has the same coverage value for a plant type. The classification process starts with sorting all fractal values from high to low for each plant type. In the next step, a fractal value is chosen from the sorted values for each plant type. The fractal value is chosen based coverage value of the plant type. The coverage value is used a percentile. This means that a coverage value of 60 percent results in the percentile of .6 of the sorted fractal values. The value that is obtained is used as threshold value for all fractal values. The points that have a fractal value higher than the threshold value are selected. This process is done separately for each plant type. This results in X maps with points selected where X is the number of plant types. For each point position, a check is made to determine how many plant types have been selected for that point. If only one plant type has been selected that plant type is assigned to that point. If more than one plant type has been selected for the point, than the plant type is assigned which has the highest fractal value at that point position. The previous steps do not classify all the point yet, because normally there are conflicting points. To classify the rest of the points, first the composition values of the remaining points have to be updated. New composition values are assigned to the remaining points based on the missing composition. The remaining points are then classified per plant type. The order of plant types is based on the average shape metrics per plant type. The plant type with largest deviation in shape metric values is assigned first. The plant type with the lowest deviation is assigned last. This process is visually expressed in figure 3.25.

Fractal values are calculated per point for each plant type (frequency is respectively .4, .55, .8 for each plant type). Thus, this step results in three fractal point maps



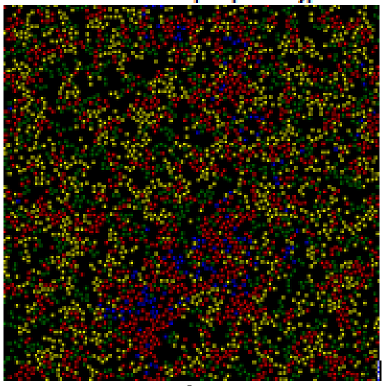
↓

Based on the coverage values (40%, 50%, 10%) of each plant type a threshold is applied on each point. This results in three maps that each matches with the corresponding coverage for each plant type

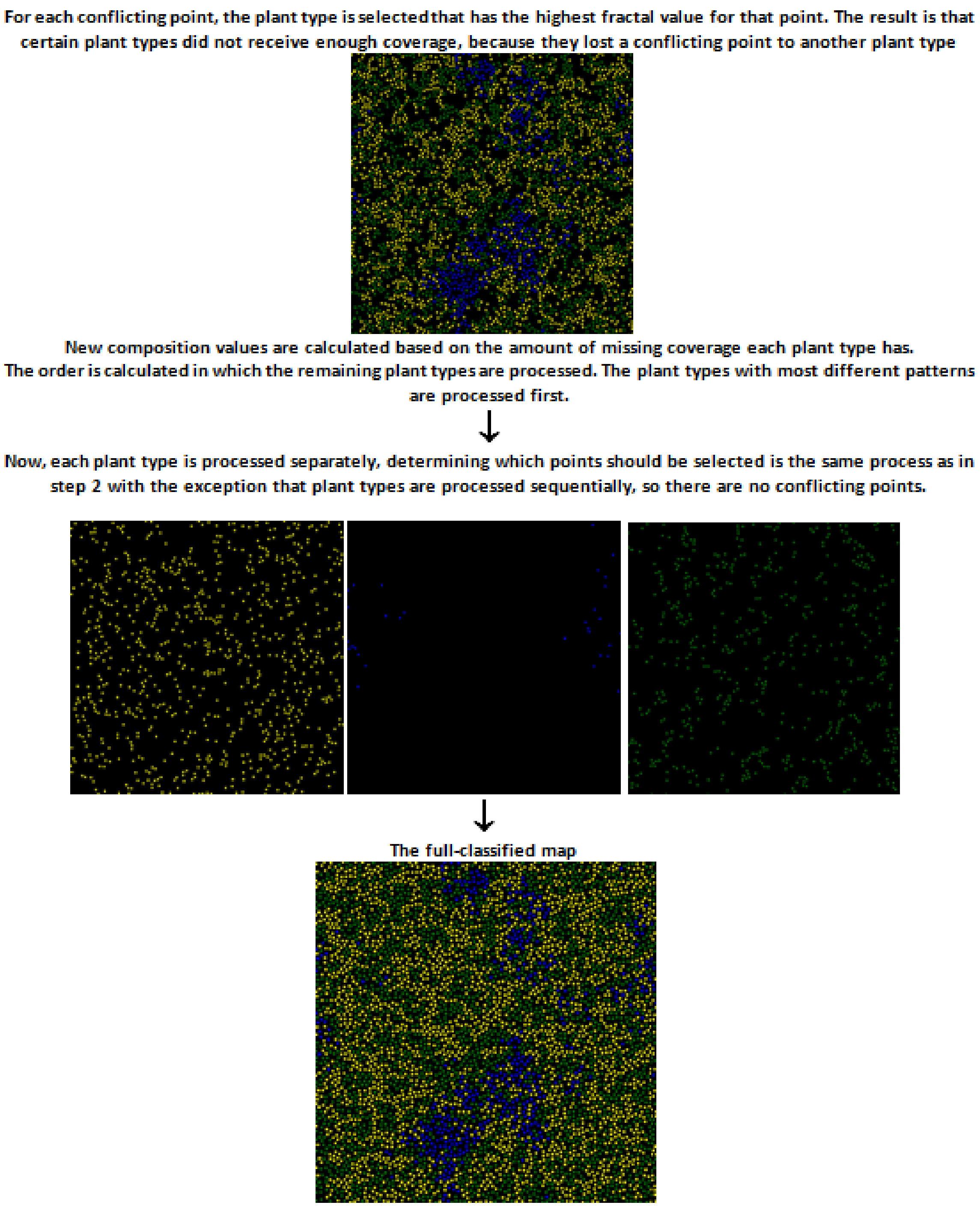


↓

Maps are put on each other and points that are conflicting are marked red. These are the points that are classified by multiple plant types



↓


For each conflicting point, the plant type is selected that has the highest fractal value for that point. The result is that certain plant types did not receive enough coverage, because they lost a conflicting point to another plant type

New composition values are calculated based on the amount of missing coverage each plant type has.
The order is calculated in which the remaining plant types are processed. The plant types with most different patterns are processed first.

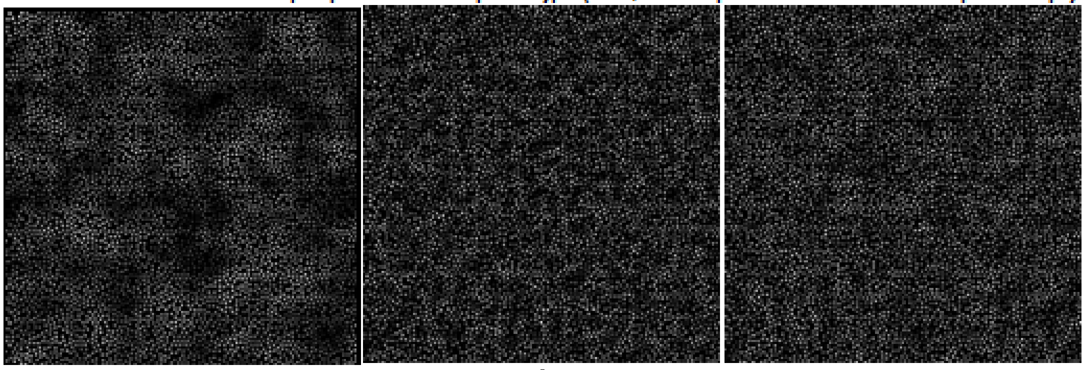
Now, each plant type is processed separately, determining which points should be selected is the same process as in step 2 with the exception that plant types are processed sequentially, so there are no conflicting points.

The full-classified map

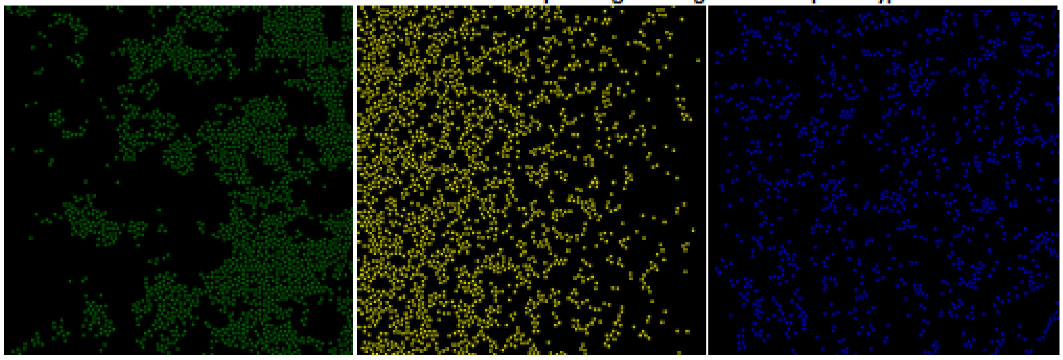
Figure 3.25: Process of classification with static composition values.

Non-static composition values mean that a single plant type has different coverage values in the environment. The process for static composition needs to be extended to be able to support this. Two extensions are necessary. The first extension is during the thresholding phase: instead of selecting a global threshold value, a threshold value must be selected for each point. Each point selects its threshold value based on the coverage it has assigned for that plant type. The remaining threshold process is not changed. Thus, a point is selected when its fractal value is higher than the threshold. A second extension is necessary after the first points are classified. After the first classification, the composition values of the remaining points are updated. For the dynamic case, this is not a simple process, because compositions values are different over the area, which means that it is possible that at a certain location a lot of coverage is missing for a plant type, but somewhere else in the area that plant types received enough coverage. Thus, the remaining points that lay in the area where there is a lot of coverage missing should be updated with a higher coverage value to get classified, while the points in the other area should get a low coverage value. This is achieved by creating sample points for each (a)biotic factor that has a relation to the composition values. Each sample points represents a value of the (a)biotic factor it represents. For each of these sample point the weighted missing coverage is calculated. For example, if there is an area that has height values ranging from 10 to 20 meters then it is possible to generate sample points for 10m, 15m, and 20m. All points in the area that placed at 20m height will calculate the missing coverage for the 20m sample point. If a point position has a height that is somewhere in between two sample points then it will give a weighted distribution to each sample point. Thus a point at 17m will contribute 60 percent to the sample point at 15m and 40 percent to the sample point at 20m. In the end, the amount of missing coverage is known for each sample point of every (a)biotic factor. The next step is to calculate the missing coverage for each remaining point that needs to be classified. This is achieved by doing an interpolation for each remaining point with the two closest sample points. This results in a missing coverage value of each (a)biotic value for every plant type. These values needs to merged into one coverage value for each plant type and this is done by selecting the lowest coverage values for each point. An example process with non-static composition values is visualized in figure 3.26

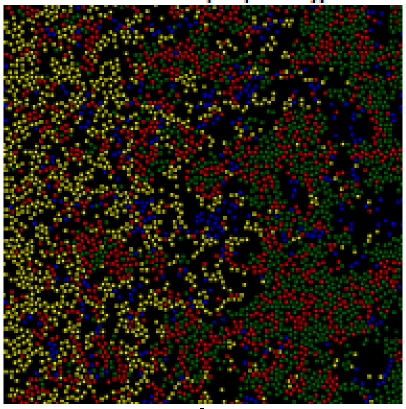
Fractal values are calculated per point for each plant type (Thus, this step results in three fractal point maps)



Based on the coverage values of each plant type a threshold is applied on each point. This results in three maps that each matches with the corresponding coverage for each plant type

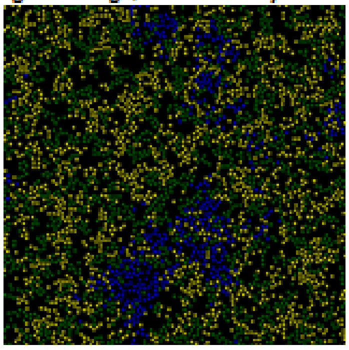


Maps are put on each other and points that are conflicting are marked red. These are the points that are classified by multiple plant types



For each conflicting point, the plant type is selected that has the highest fractal value for that point. The result is that certain plant types did not receive enough coverage, because they lost a conflicting point to another plant type

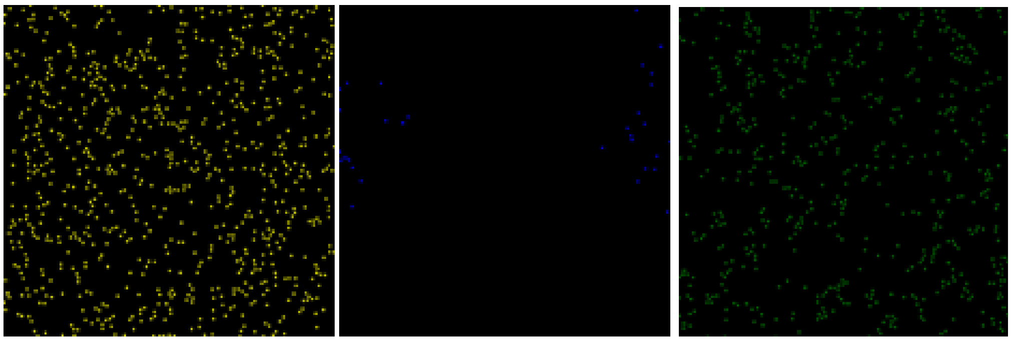
For each conflicting point, the plant type is selected that has the highest fractal value for that point. The result is that certain plant types did not receive enough coverage, because they lost a conflicting point to another plant type



New composition values are calculated based on the amount of missing coverage each plant type has.
The order is calculated in which the remaining plant types are processed. The plant types with most different patterns are processed first.

↓

Now, each plant type is processed separately, determining which points should be selected is the same process as in step 2 with the exception that plant types are processed sequentially, so there are no conflicting points.



↓

The full-classified map

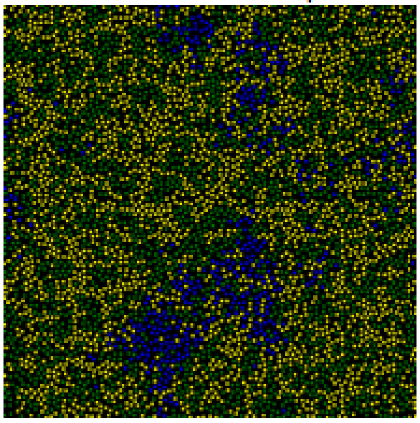
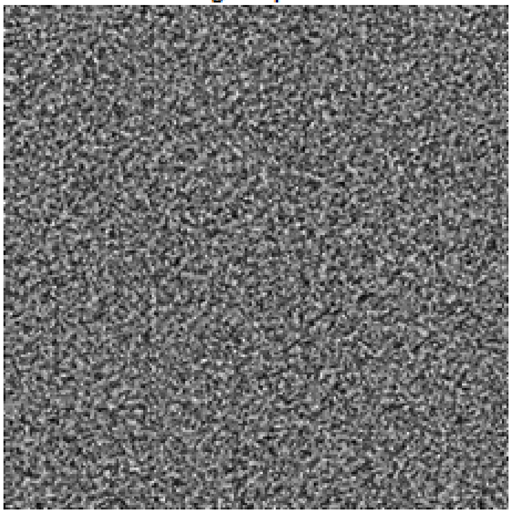


Figure 3.26: Process of classification with non-static composition values.

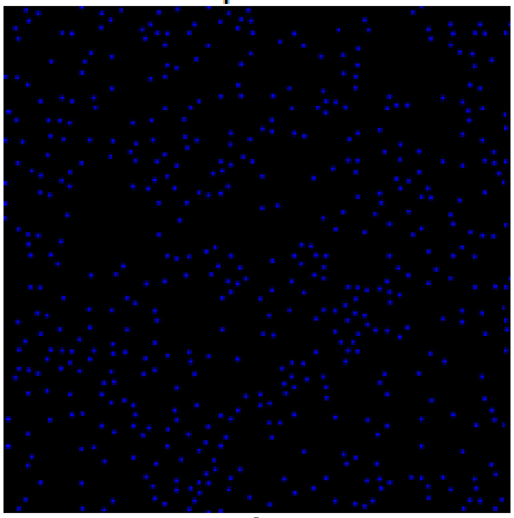
Multiclass classification Multiclass classification is used when multiple plant levels have to be classified. This classification is only possible if during the point generation component a multiclass point distribution is generated. The classification process stars with classifying the plant types in the highest plant level. For this classification, only the point locations are used that belong to the highest plant level. The remaining classification process for that plant level is not different from the classification processes discussed in the previous paragraphs. After the highest plant level is classified, it is possible that the assigned plants have a certain influence on the remaining plant types in the lower plant levels. For each classified the neighbouring points are retrieved and the composition values of these points are changed based on the influence of the classified plant. After that the classification of the highest plant level is finished and this whole process is repeated for the second highest plant level until all plant levels have been processed. Points that did not get classified in a certain plant level are used in the classification process of the remaining plant levels. In figure 3.27 the process of multiclass classification is visually expressed. In this example there are three plant types with two plant levels.

Fractal map of the blue plant type, which is processed first, because it is the only plant type group in the highest plant level



↓

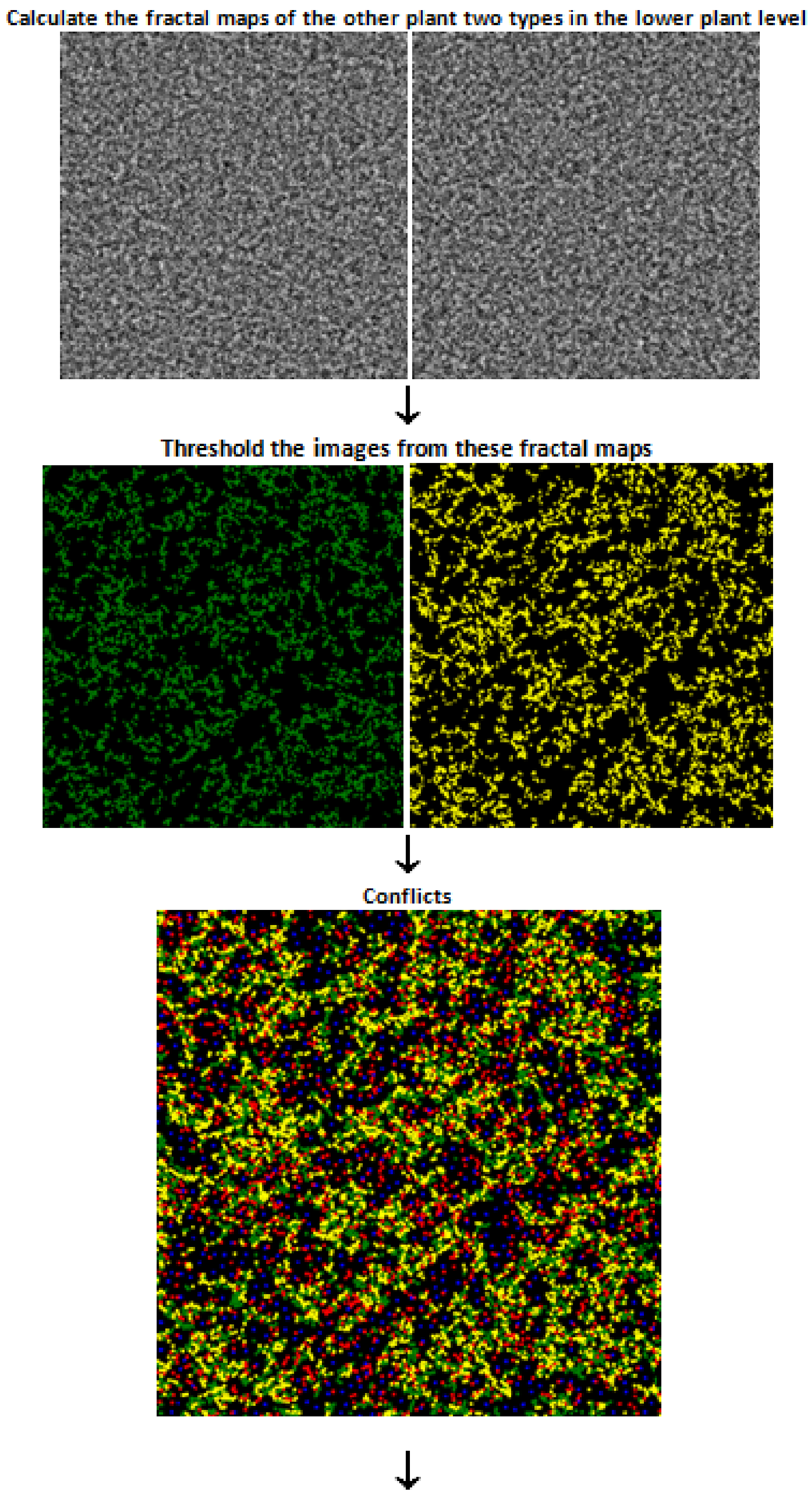
Classified blue points, since there is not competition this also the end results for the blue points



↓

This step gives the possibility to apply a plant influence on the neighbouring points based on the positions of the classified blue points. In this case, a negative effect is applied as will be visible later.

↓


Calculate the fractal maps of the other plant two types in the lower plant level
Threshold the images from these fractal maps
Conflicts

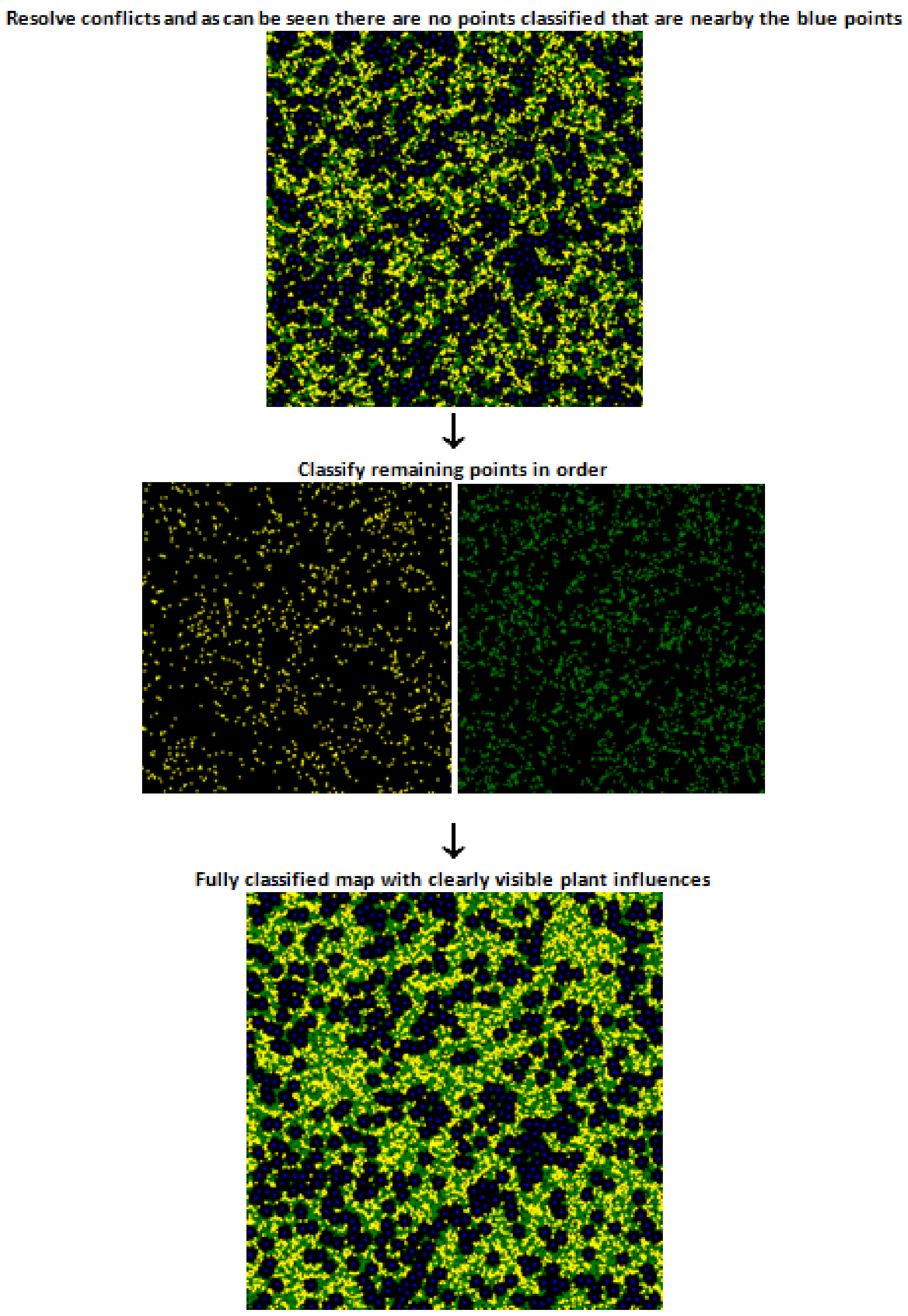

Resolve conflicts and as can be seen there are no points classified that are nearby the blue points
Classify remaining points in order
Fully classified map with clearly visible plant influences

Figure 3.27: Process of classification with different plant levels.

3.3. PRELIMINARY RESULTS

This section gives the preliminary results of this research, which is the development of the plant placement algorithm. In the next chapter, it will be investigated whether the results of the algorithm are realistic and correct. The algorithm uses a four-step process to calculate a plant distribution from spatial data. First, it is calculated where vegetation grows in the area in the data processing 1 component based biotic map data. This information is used in the point generation component to generate all possible plant locations. During the point generation, it is possible to include existing plant point datasets. In the second data processing component, the remaining data input is used to calculate for each possible plant location composition and shape metric values of every plant type. In the point classification component all possible plant locations are classified using the composition values and fractal values that are calculated from the shape metric values. Spatial has a large influence on where possible plant locations are generated and how they are classified. First, spatial can determine where vegetation grows in area which makes it possible to define the global patterns of the whole vegetation distribution. Second, during the point generation existing point can already specify which possible plant locations should be classified, because a plant is located there. Third, the data influences the composition and shape metric values. Each data source can limit the locations where certain plant types grow, because during the classification the lowest composition values are selected for each plant type per point. The algorithm meets the criteria to work with spatial data to derive information of where to place plants and plant types. Also, a wide variety of spatial data input is possible. To conclude this chapter, an overview of the process of this algorithm is described in pseudocode 5

Algorithm 5

```
procedure PLANT PLACEMENT ALGORITHM(data)
    $highVegetationMap \leftarrow generateVegetationMap(data[biotic, coverage])$
    $lowVegetationMap \leftarrow createLowResMap(highVegetatioMap, data[pointSpacing])$

    $pointDistribution \leftarrow generatePoints(lowVegetationMap, data[pointSpacing])$

    $pointDistribution \leftarrow cullPointDistribution(pointDistribution, highVegetationMap)$
    $compositionPoints \leftarrow calculateCompositionData(pointDistribution, data[(a)biotic])$
    $shapemetricPoints \leftarrow calculateShapemetricData(pointDistribution, data[(a)biotic])$

    $fractalPoints\ getscalculateFractalValues(pointdistribution, shapemetricPoints)$
    $plantDistribution\ getsclassifyPoints(pointdistribution, fractalPoints, compositionPoints)$

    return $plantDistribution$
```

4
Implementation and Testing

This chapter discusses and shows results of the plant placement algorithm applied to real data sources. The results are plant distributions generated for an existing salt marsh area in the Netherlands called the Paulinapolder, and a future salt marsh area represented with an ecological model. The algorithm that generates these results is implemented in Python, code can be found in Appendix C and D. QuantumGIS has processed some of the input data for the algorithm. This chapter starts explaining salt marshes in general and then introduces each test area. In the next section, the results of the test areas are shown. This is followed by a section that validates these results with statistical and expert validation. Finally, the validation results are discussed.

4.1. Salt marshes

The plant placement algorithm is tested on salt marshes, which are a type of natural environment. This graduation project is done in collaboration with NIOZ, which is a research institute that is interested in generation point positions for plants of salt marshes. In the end, they want to use these point position for the 3D visualization of salt marshes. Salt marshes are coastal wetlands that are flooded and drained by water brought in by the tides. The interactions of the flooding and drainage of water with the settled plants on the marsh creates complex feather-shaped creek networks [46]. The process and the resulting patterns in the network is also called ecological self-organization. Figure 4.1 shows an aerial image of a salt marsh where these complex networks are visualized.

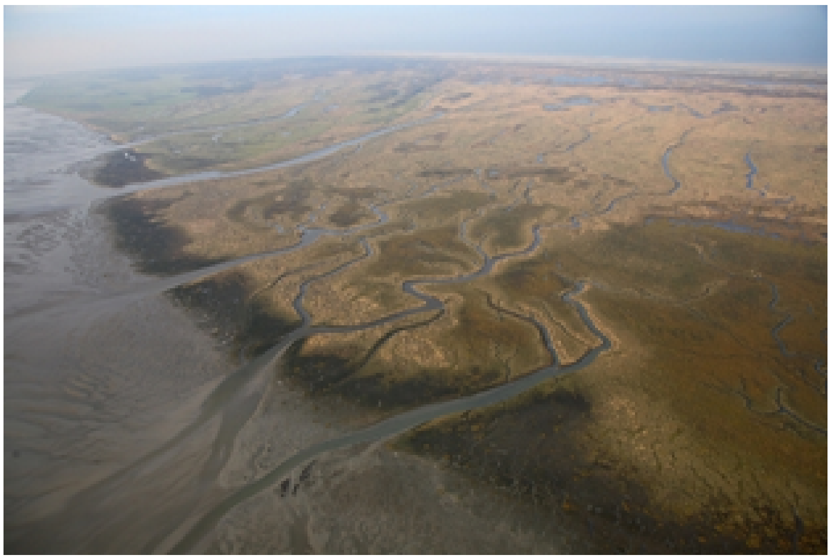
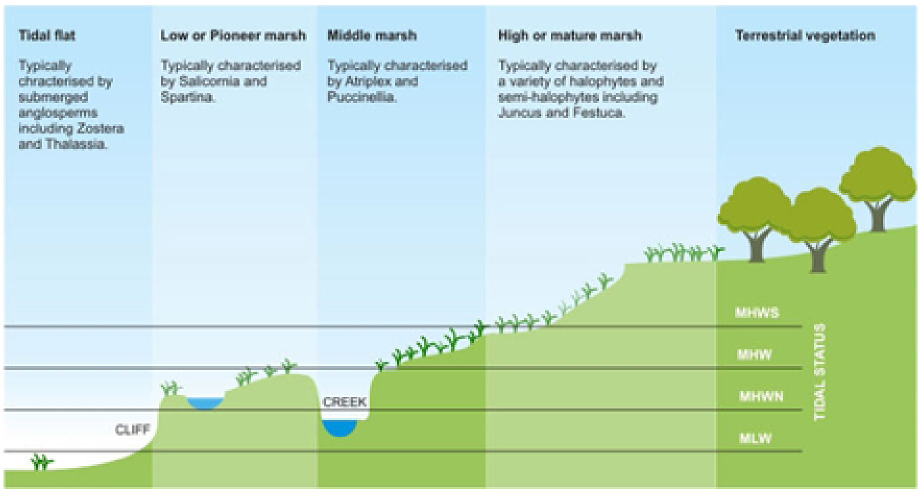


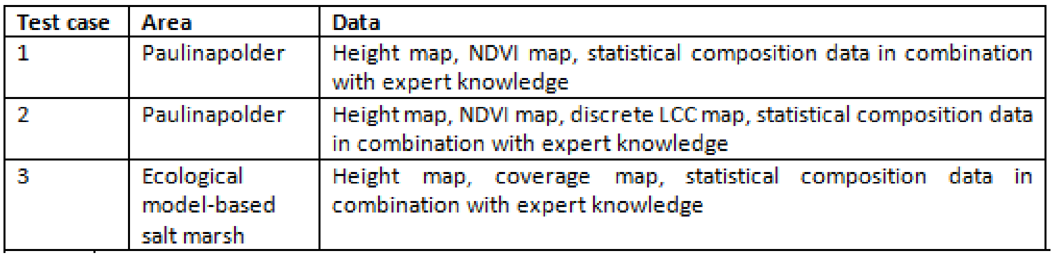
Figure 4.1: Aerial image of a salt marsh (Google Image).

Salt marshes consist of different plant zones. In each zone, different types of plants grow. The main zones are the pioneer, low, middle, and high marsh. In the pioneer and the low marsh zones plants settle on the bare mud flats which facilitate settlement of other plants from the middle and high marsh zone in the future. Figure 4.2 gives an overview of the different zones of a salt marsh.


Tidal flat
Typically chracterised by submerged anglosperms including Zostera and Thalassia.
Low or Pioneer marsh
Typically characterised by Salicornia and Spartina.
Middle marsh
Typically characterised by Atriplex and Puccinellia.
High or mature marsh
Typically characterised by a variety of halophytes and semi-halophytes including Juncus and Festuca.
Terrestrial vegetation
CLIFF
CREEK
MHWS
MHW
MHWN
MLW
TIDAL STATUS
Figure 4.2: Overview of the plant zones in a salt marsh (GoogleImage).

Initially, it was planned to create a point distribution for the salt marsh that is going to grow in the Hedwigepolder. Unfortunately, this was not possible, because data about this environment could not be made available. Instead, the plant placement algorithm is tested with three other salt marsh cases. The first two cases use data of an existing salt marsh called the Paulinapolder. The third case uses data of a future salt marsh generated by an ecological model. Two test cases are performed for the Paulinapolder, because different input data used: one where discrete LCC data is included and one where it is not. This is to test the influence of LCC data on the output of the algorithm. Other data input used for the Paulinapolder cases are NDVI, height, and statistical composition data. The third case uses height and coverage maps generated by an ecological model, and statistical composition data. Table 4.1 gives an overview of the input data for each case. In the next two subsection the available data of each area (Paulinapolder and ecological model-based salt marsh) is discussed in more detail.

Table 4.1: The test case areas with their input data



Test case	Area	Data
1	Paulinapolder	Height map, NDVI map, statistical composition data in combination with expert knowledge
2	Paulinapolder	Height map, NDVI map, discrete LCC map, statistical composition data in combination with expert knowledge
3	Ecological model-based salt marsh	Height map, coverage map, statistical composition data in combination with expert knowledge

4.1.1. PAULINAPOLDER

The Paulinapolder is a salt marsh area located along the Westerschelde in the Netherlands. In figure 4.3 an aerial image of the Paulinapolder salt marsh is shown.

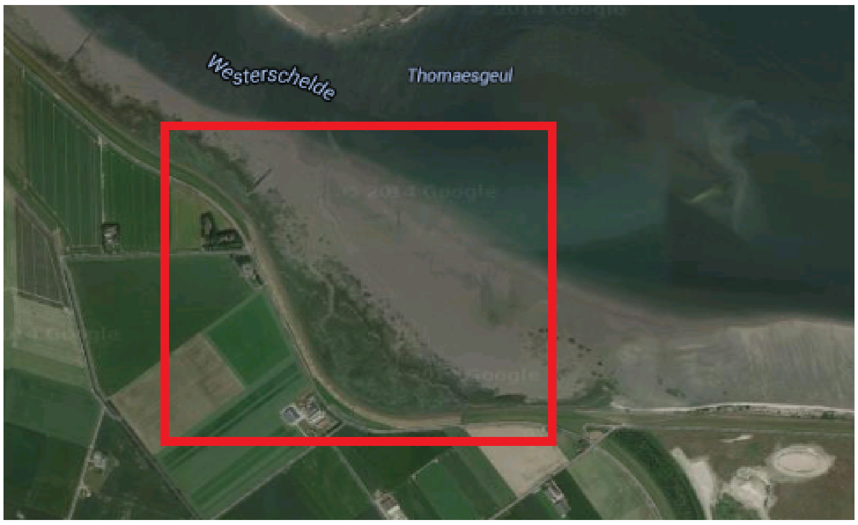


Figure 4.3: Aerial photo of the Paulinapolder by GoogleEarth. Paulinapolder is indicated by the red bounding box. The salt marsh is connected to the Westerschelde.

There are two test cases for the paulinapolder. For the first case a plant distribution is generated based on height, NDVI, and statistical composition data. The statistical composition data is related to the height of the marsh. The second case also includes a discrete LCC map. The plant types that are used for both cases are the same. The following plant types are selected for the tests:

- Spartina Anglica
- Aster Tripolium
- Limonium Vulgare
- Elymus Athericus
- Atriplex Portulacoides
- Artemisia Maritima
- Salicornia Europaea

Each input data source is now discussed shortly. The first data source discussed is the height map. The height map falls under abiotic data input category of the plant placement algorithm. The input map is shown in figure 4.4. The resolution of the map is 1m. In the algorithm, the statistical composition values are connected to the height values of the height map.

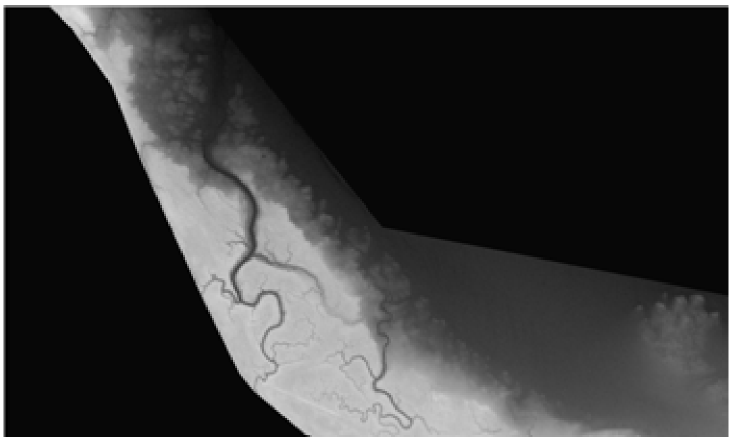


Figure 4.4: Heightmap Paulinapolder.

The second data source is the NDVI map as seen in figure 4.5. The NDVI map falls under the biotic map category of the plant placement algorithm. The NDVI map is used to define where vegetation grows in general. The threshold is set at 0.02, which means that tiles with NDVI values above the threshold are marked as tiles with vegetation. In addition, NDVI values are used to adjust the composition value for each point to differentiate between the location of Salicornia and the other plant types. This is done, because Salicornia should not bet mixed with the other plant types. Therefore, in the end points that have NDVI values lower than .08 only have the possibility to be assigned to Salicornia or nothing. Points with a larger value cannot be assigned to Salicornia. The resolution of the NDVI map is 1m.

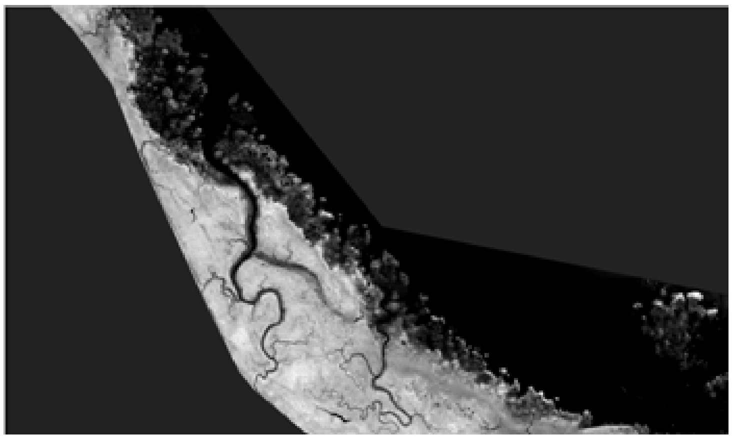


Figure 4.5: NDVI map Paulinapolder

The third data source is the discrete LCC map, as shown in figure 4.6. This map is only used as input for the second test case. The LCC contains information about where each plant zone is located in the marsh. The categories of the LCC map are thus the different plant zones: pioneer, low, middle, high marsh. In collaboration with NIOZ the following plant types are assigned to each zone:

- Pioneer marsh: Spartina, Salicornia
- Low marsh: Spartina, Aster
- Middle marsh: Aster, Atriplex, Limonium, Artemisia
- High marsh: Elymus

The LCC map was rasterized from vector data to a grid with 1m resolution. However, the vector LCC map is created based on false color aerial images with 5m resolution in combination with field observations [47]. This means that the actual accuracy/resolution of the data is lower than the 1m resolution.

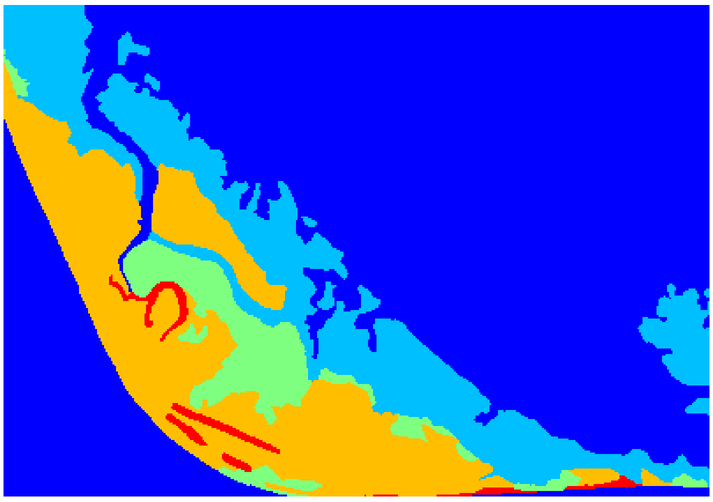
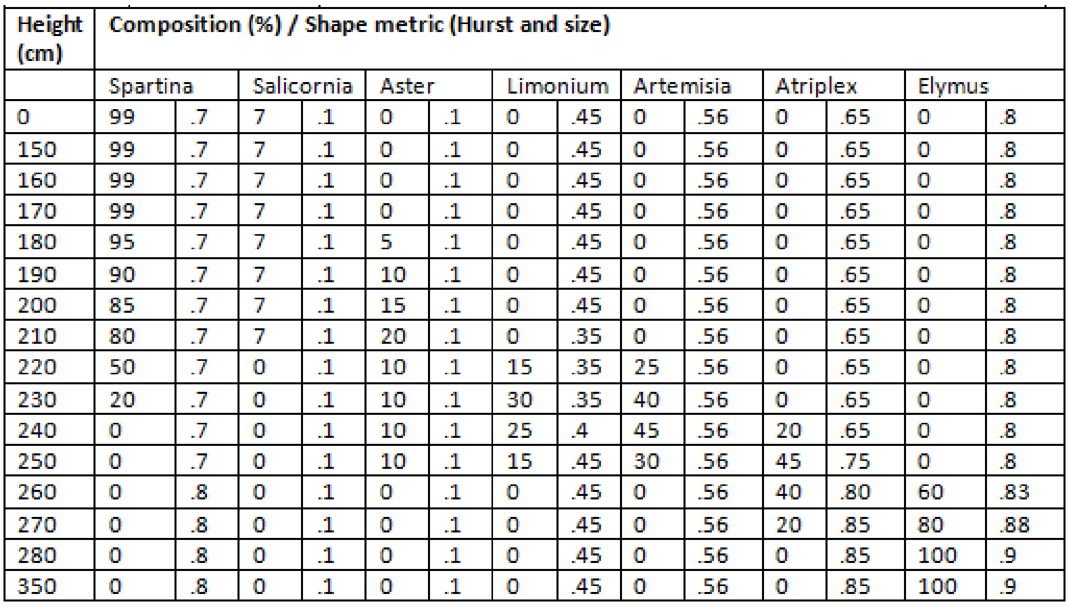


Figure 4.6: LCC map Paulinapolder where dark blue is no vegetation, light blue is pioneer marsh, green is low marsh, orange is middle marsh, red is high marsh.

The last data source is statistical composition data and shape metric data. The composition data is related to the height of the marsh. This means that a coverage value for a plant type is returned based on a height value. Initially, statistical composition data was used from de Leeuw [42]. However, because this data was obtained from a different salt marsh in the Oosterschelde, applying this data gave non-realistic results for the location of the plant types. For example, the Elymus plant type was placed in the pioneer zone, while it only grows on the highest parts of a marsh. Therefore, the composition values were changed to match it with the Paulinapolder salt marsh. The changes were made based on expert knowledge from NIOZ. The new composition values are shown in table 4.2. This table also includes the shape metric values for each plant type based on the height of the marsh.

Table 4.2: Composition and Shape metric statistics



Height (cm)	Composition (%) / Shape metric (Hurst and size)													
	Spartina		Salicornia		Aster		Limonium		Artemisia		Atriplex		Elymus	
0	99	.7	7	.1	0	.1	0	.45	0	.56	0	.65	0	.8
150	99	.7	7	.1	0	.1	0	.45	0	.56	0	.65	0	.8
160	99	.7	7	.1	0	.1	0	.45	0	.56	0	.65	0	.8
170	99	.7	7	.1	0	.1	0	.45	0	.56	0	.65	0	.8
180	95	.7	7	.1	5	.1	0	.45	0	.56	0	.65	0	.8
190	90	.7	7	.1	10	.1	0	.45	0	.56	0	.65	0	.8
200	85	.7	7	.1	15	.1	0	.45	0	.56	0	.65	0	.8
210	80	.7	7	.1	20	.1	0	.35	0	.56	0	.65	0	.8
220	50	.7	0	.1	10	.1	15	.35	25	.56	0	.65	0	.8
230	20	.7	0	.1	10	.1	30	.35	40	.56	0	.65	0	.8
240	0	.7	0	.1	10	.1	25	.4	45	.56	20	.65	0	.8
250	0	.7	0	.1	10	.1	15	.45	30	.56	45	.75	0	.8
260	0	.8	0	.1	0	.1	0	.45	0	.56	40	.80	60	.83
270	0	.8	0	.1	0	.1	0	.45	0	.56	20	.85	80	.88
280	0	.8	0	.1	0	.1	0	.45	0	.56	0	.85	100	.9
350	0	.8	0	.1	0	.1	0	.45	0	.56	0	.85	100	.9

4.1.2. ECOLOGICAL MODEL-BASED SALT MARSH

The ecological model-based salt marsh is generated with a dynamic ecological model developed by Schwarz [48] based on the work of Temmerman [38]. The model of Schwarz generates a height map of the marsh and a continuous coverage map for the Spartina plant type. For the plant distribution in this test case, not only Spartina should be placed. Other plant types are also included. Composition data of these plant types are included with statistical composition data. Further, the output of the model does generate each plant zone of the salt marsh, but only the pioneer and low marsh zones are generated. This means that the following plant types were selected, because the generated data is restricted to these two plant zones:

- Spartina
- Salicornia
- Aster

The first map data source discussed is the height map. The height map is used to connect the statistical composition values, which is used to connect the statistical composition values. The height values of the map were scaled between 0 and 100, to easier connect the composition values for the user. The model itself generated values between 0 and -1. The height map generated by the ecological model is shown in figure 4.7. The resolution of the height map is 6m.

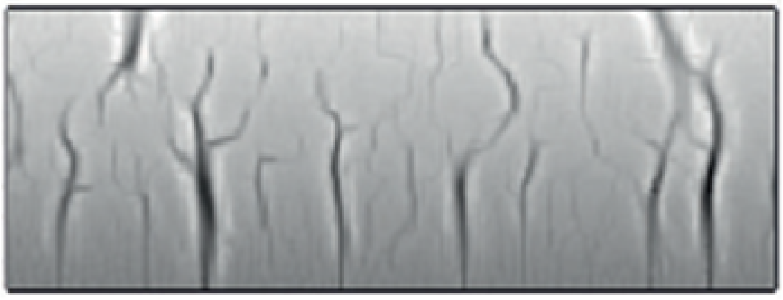


Figure 4.7: Map generated by the ecological model of Schwarz [48]

The second data source is the coverage map for Spartina. This map is not used directly for the composition values of Spartina, because additional plant types are added. Instead, it is used to determine where vegetation grows, like the NDVI map in the Paulinapolder cases. Tiles with coverage values that have a value larger than 0 are marked as tiles where vegetation grows. Also, the coverage map is used to again differentiate between Salicornia and the other two plant types. Saliconira is placed in the tiles with values lower than .01 and the other plant types in tiles with values higher than .01. The coverage map is shown in figure 4.8

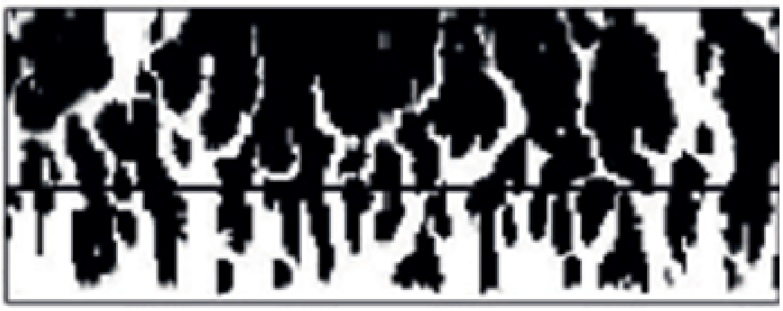


Figure 4.8: Continuous coverage map for Spartina [48]

The last data source is the statistical composition values and shape metric data. Composition values are based on the height of the marsh. An overview of the composition and shape metrics are given in table 4.3.

Table 4.3: Composition and Shape metric statistics

Height (cm)	Composition in % / shape metrics (Hurst and size)					
	Spartina		Salicornia		Aster	
50	0	.8	0	.1	0	.1
60	99	8	5	.1	0	.1
70	99	8	5	.1	0	.1
80	95	8	5	.1	5	.1
90	90	8	0	.1	10	.1
100	90	8	0	.1	10	.1
200	90	8	0	.1	10	.1

4.2. RESULTS

This sections shows the results of using the plant placement algorithm based on the data of each test case discussed in the previous section. The results are a visualization of the point distribution of each test case. First, the point distribution are shown for each test case of the Paulinapolder. This is followed with the point distribution of the test case of the ecological-based salt marsh.

4.2.1. PAULINAPOLDER WITHOUT LCC DATA

The point distribution generated by the plant placement algorithm for the first test case is shown in figure 4.9.

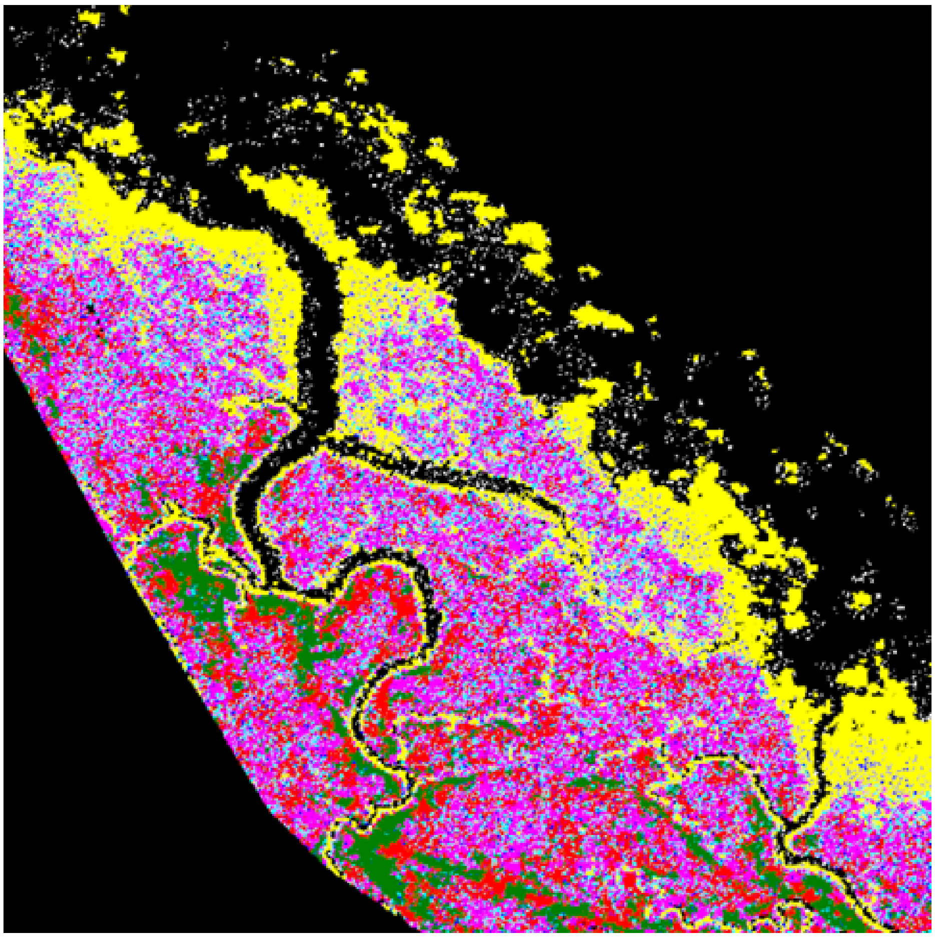


Figure 4.9: Result Paulinapolder without LCC data with the plant types Spartina (yellow), Elymus (green), Aster (blue), Salicornia (white), Limonium (teal), Artemisa (pink), and Atriplex (red)

4.2.2. Paulinapolder with LCC data

The point distribution generated by the plant placement algorithm for the second test case is shown in figure 4.10.

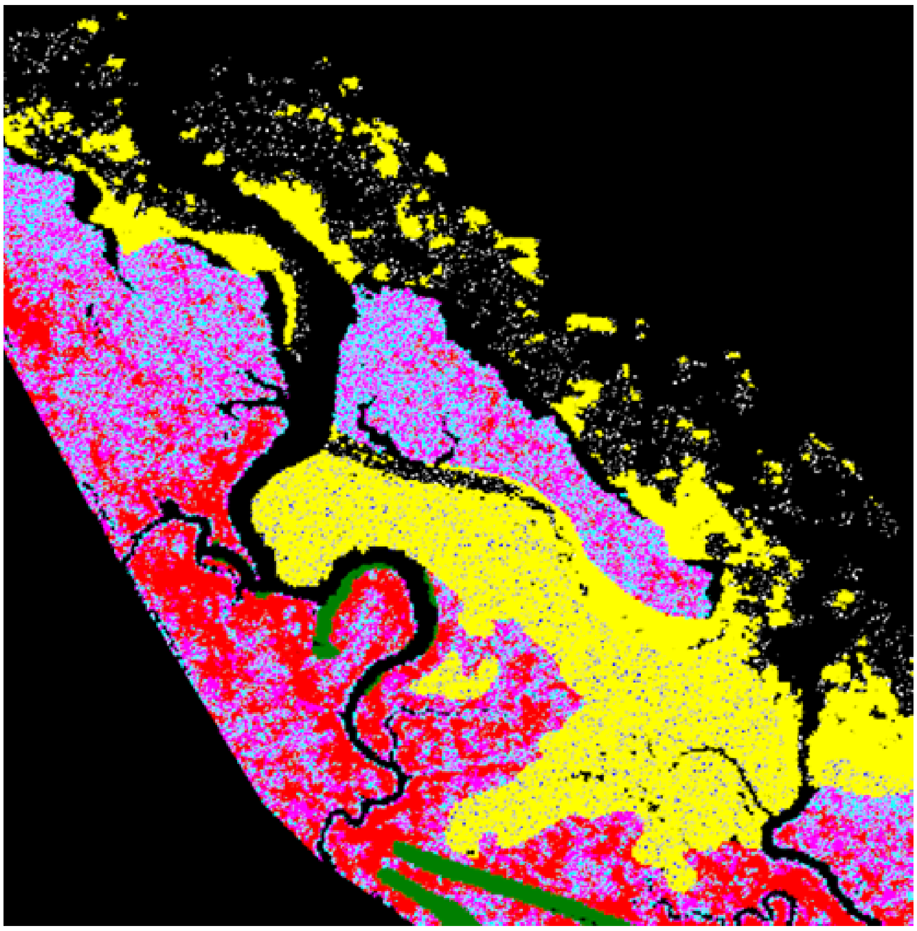


Figure 4.10: Result Paulinapolder with LCC data with the plant types Spartina (yellow), Elymus (green), Aster (blue), Salicornia (white), Limonium (teal), Artemisa (pink), and Atriplex (red)

4.2.3. Ecological model-based salt marsh

The point distribution generated by the plant placement algorithm for the third test case is shown in figure 4.11.

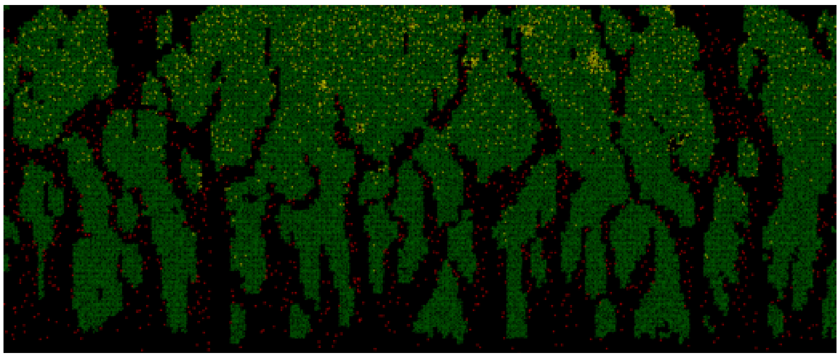


Figure 4.11: Plant point distribution based on data provided by an dynamic cellular ecological model in combination with statistical composition values with the plant types Spartina (green), Aster (yellow), and Salicornia (red).

4.3. VALIDATION

In this section, the results generated by the plant placement algorithm are validated for each test case are validated. Two types of validation are used: statistical and expert validation. Statistical validation calculates whether the algorithm correctly maintains the composition of the different plant types. This means that if a plant type has X percent coverage assigned then the final result should have X percent coverage of that plant type. The statistical validation is performed on artificial datasets discussed in the next subsection and each plant distribution discussed in the previous section. Expert validation is performed to determine the realism of the plant distribution. The realism of the distribution are discussed with an expert of NIOZ. This sections starts with discussing the statistical validation, which is followed with the expert validation.

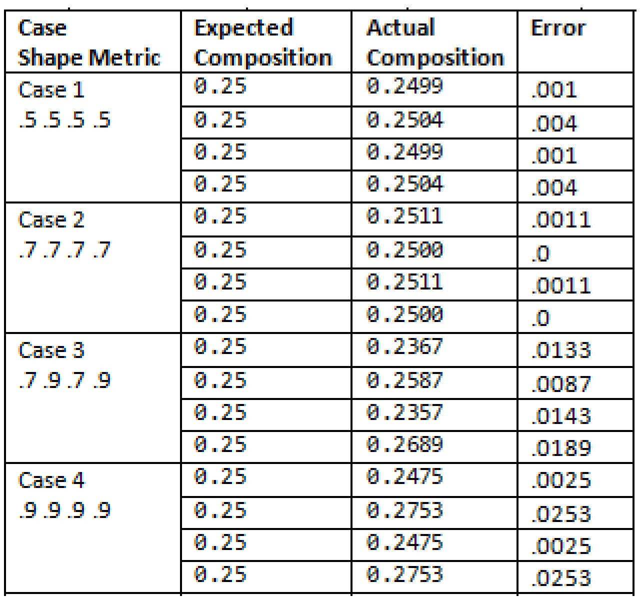
4.3.1. STATISTICAL VALIDATION

Statistical validation is used to determine whether the input composition values based on spatial data sources are correctly mapped the output plant distribution. If plant type A has X percent coverage at a certain height then the plant distribution should contain X percent coverage at specific height. First statistical validation is applied on plant distribution based on artificial datasets to investigate which factors influences the final coverage per plant type. Finally, the composition statistics are calculated for each generated plant distribution of the previous section.

ARTIFICIAL DATA

The first tests performed are with artificial data of four plant types with equal composition values of 25 percent. Four cases with the equal composition values are tested. The factor that changes each case is the shape metric input for each plant type. The shape metric input and results are shown in table 4.4.

Table 4.4: Composition results for a test set with four plant types with equal composition (.e.g. .25 for every point and plant type).

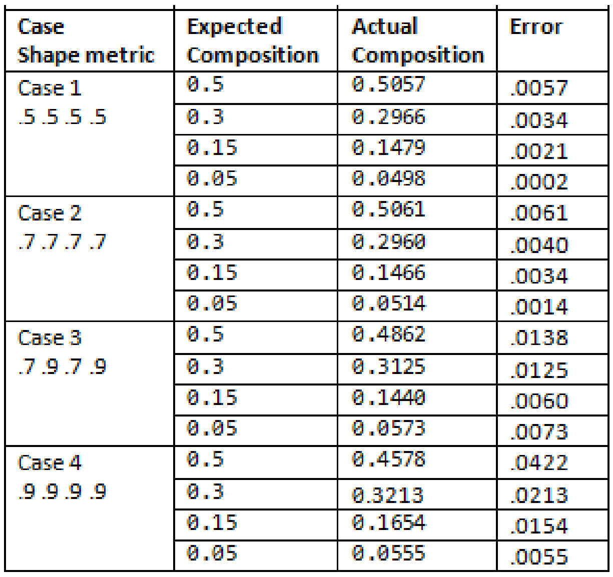


Case Shape Metric	Expected Composition	Actual Composition	Error
Case 1	0.25	0.2499	.001
.5 .5 .5 .5	0.25	0.2504	.004
	0.25	0.2499	.001
	0.25	0.2504	.004
Case 2	0.25	0.2511	.0011
.7 .7 .7 .7	0.25	0.2500	.0
	0.25	0.2511	.0011
	0.25	0.2500	.0
Case 3	0.25	0.2367	.0133
.7 .9 .7 .9	0.25	0.2587	.0087
	0.25	0.2357	.0143
	0.25	0.2689	.0189
Case 4	0.25	0.2475	.0025
.9 .9 .9 .9	0.25	0.2753	.0253
	0.25	0.2475	.0025
	0.25	0.2753	.0253

These results show that when the shape metric value becomes larger (e.g. the patterns become larger), the error become larger in the actual composition. We can see in the results that when the shape sizes becomes larger that the error becomes larger in the composition values that assigned. However, the error values are low for all cases.

The second test with artificial data uses four plant types again, but these plant types do not have the same composition values. The shape metric value are equal to the shape metric input of the previous artificial test case. Table 4.5 gives an overview of the input and the results.

Table 4.5: Composition results for a test set with four plant types with unequal composition.

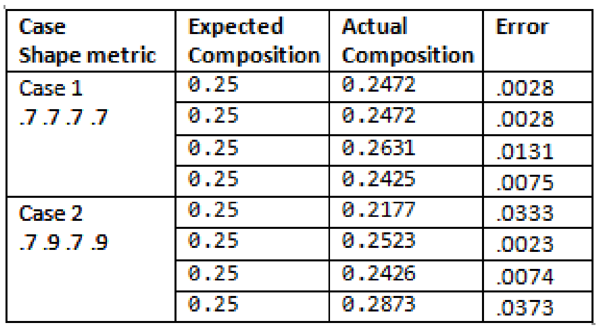


Case Shape metric	Expected Composition	Actual Composition	Error
Case 1 .5 .5 .5 .5	0.5	0.5057	.0057
	0.3	0.2966	.0034
	0.15	0.1479	.0021
	0.05	0.0498	.0002
Case 2 .7 .7 .7 .7	0.5	0.5061	.0061
	0.3	0.2960	.0040
	0.15	0.1466	.0034
	0.05	0.0514	.0014
Case 3 .7 .9 .7 .9	0.5	0.4862	.0138
	0.3	0.3125	.0125
	0.15	0.1440	.0060
	0.05	0.0573	.0073
Case 4 .9 .9 .9 .9	0.5	0.4578	.0422
	0.3	0.3213	.0213
	0.15	0.1654	.0154
	0.05	0.0555	.0055

The results of the second artificial test case also have a larger error when the shape metrics become larger. Some errors are slightly larger than the errors of the first test case, but it is not significant.

The third test case with artificial data uses 4 plant types as well. This time the coverage values for a single plant type are not equal for the whole area. The coverage values for each plant type follow a gradient. In this case, two plant types grow more at the left side of the map and gradually appear less on the right side of the map. For the other two plant types, this is vice versa. The results are shown in Table 4.6.

Table 4.6: Composition results for a test set with four plant types with non-static composition. Two plant types (1 and 2) have a composition gradient of 0 to .5 and the others (3 and 4) have the inverse gradient.



Case Shape metric	Expected Composition	Actual Composition	Error
Case 1 .7 .7 .7 .7	0.25	0.2472	.0028
	0.25	0.2472	.0028
	0.25	0.2631	.0131
	0.25	0.2425	.0075
Case 2 .7 .9 .7 .9	0.25	0.2177	.0333
	0.25	0.2523	.0023
	0.25	0.2426	.0074
	0.25	0.2873	.0373

The result from this table also shows the same trend that larger shape metric values introduce a larger error. To explain this trend, the plant distributions of the third test case are visualized in figure 4.12.

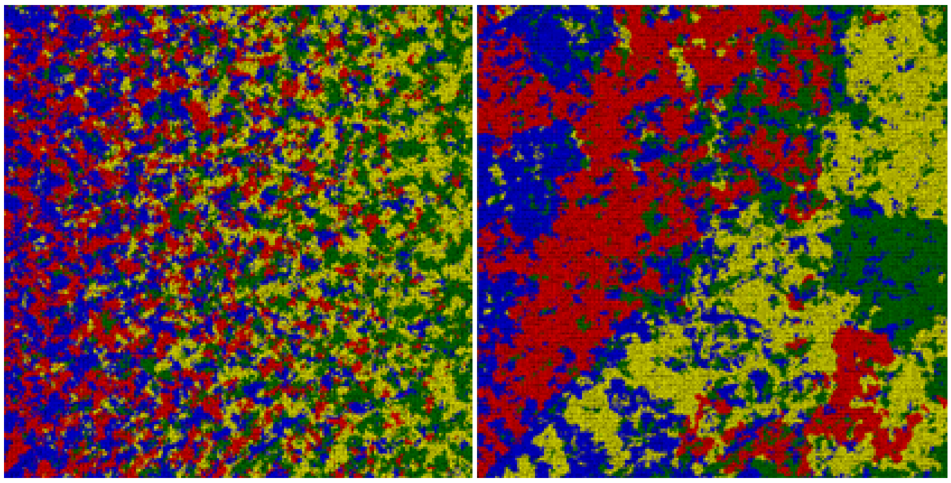


Figure 4.12: Generated results of non-static composition test set. The first image is with shape sizes of .7 and the right image with shape sizes of resp. .7 .9. .7, .9. Each color represent a different plant type. Blue is the first plant type, red the second, green the third and yellow the fourth. The number of each plant type corresponds with the order of the values of the composition and shape metrics

These plant distributions show that the patterns become very large with high shape metric values. Basically, the patterns become too large for the area, e.g. the given size of the patterns is not realistic for the environment. If, the size of area becomes larger the error becomes smaller. Table 4.7 shows that the error became smaller when the area of the map is doubled. Figure 4.13 visualizes the plant distribution with the doubled area.

Table 4.7: Composition statistics

Case Shape metric	Expected Composition	Actual Composition	Error
Case 1 .7 .9 .7 .9	0.25	0.2486	.0014
	0.25	0.2495	.0005
	0.25	0.2625	.0125
	0.25	0.2394	.0106

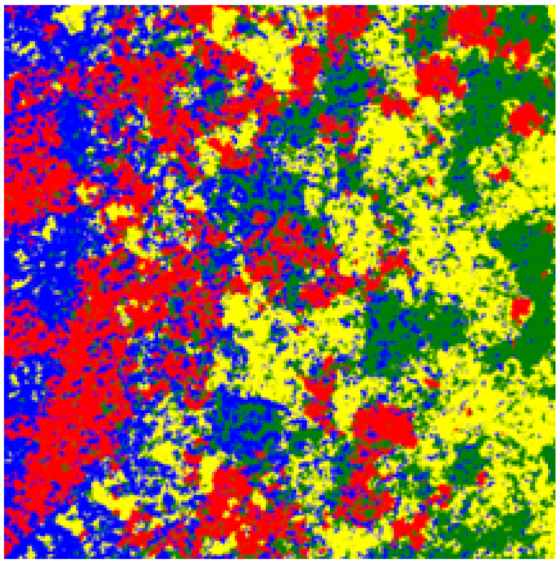
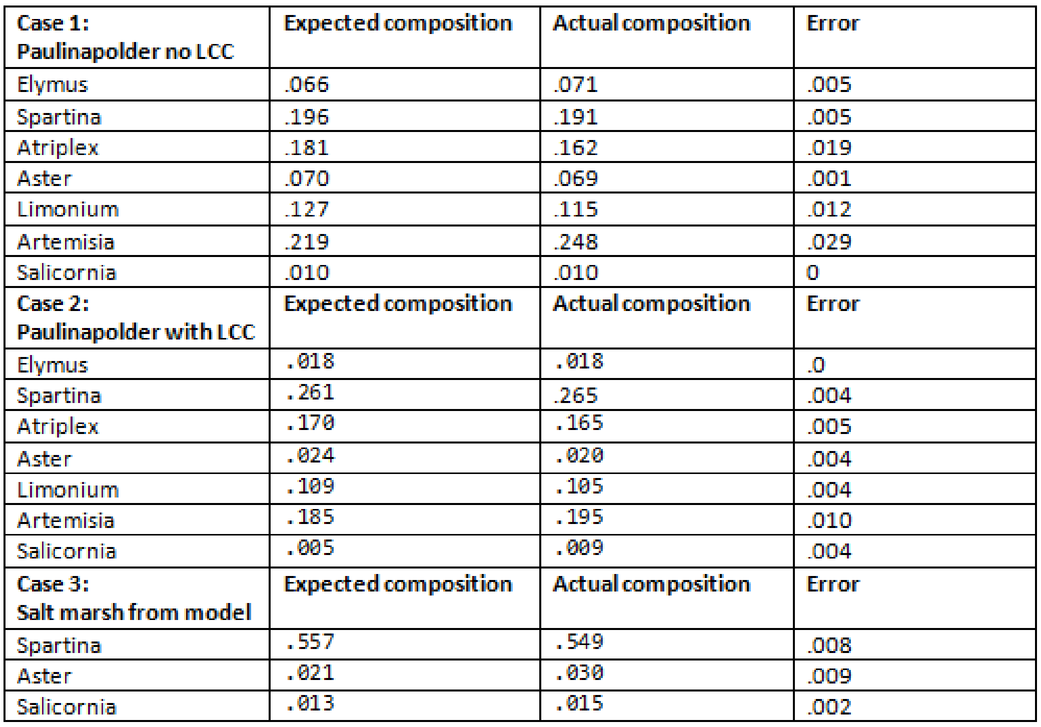


Figure 4.13: Four plant types with shape sizes of resp .7, .9, .7 , .9. with a doubled map size. Blue is the first plant type, red the second, green the third and yellow the fourth. The number of each plant type corresponds with the order of the values of the composition and shape metrics. Thus, red and yellow have a .9 shape metric value.

REAL DATASETS

Finally, statistical validation is applied on the results of the real data test cases of the previous section. The results of this validation are summarized in table 4.8.

Table 4.8: Composition statistics for real environment cases.



Case 1: Paulinapolder no LCC	Expected composition	Actual composition	Error
Elymus	.066	.071	.005
Spartina	.196	.191	.005
Atriplex	.181	.162	.019
Aster	.070	.069	.001
Limonium	.127	.115	.012
Artemisia	.219	.248	.029
Salicornia	.010	.010	0
Case 2: Paulinapolder with LCC	**Expected composition**	**Actual composition**	**Error**
Elymus	.018	.018	.0
Spartina	.261	.265	.004
Atriplex	.170	.165	.005
Aster	.024	.020	.004
Limonium	.109	.105	.004
Artemisia	.185	.195	.010
Salicornia	.005	.009	.004
Case 3: Salt marsh from model	**Expected composition**	**Actual composition**	**Error**
Spartina	.557	.549	.008
Aster	.021	.030	.009
Salicornia	.013	.015	.002

For all three cases the error is low, especially for case 2 and 3. The highest error is almost 3 percent for the Artemisia plant type in case 1 which got 2.9 percent extra coverage assigned. There is no clear reason, but this error might be due to different shape metric values. Often, Artemisia is placed together with Limonium and Atriplex. Atriplex has a high shape metric value, which might be the cause for the relatively large error in composition in comparison to the rest of the plant types for that case.

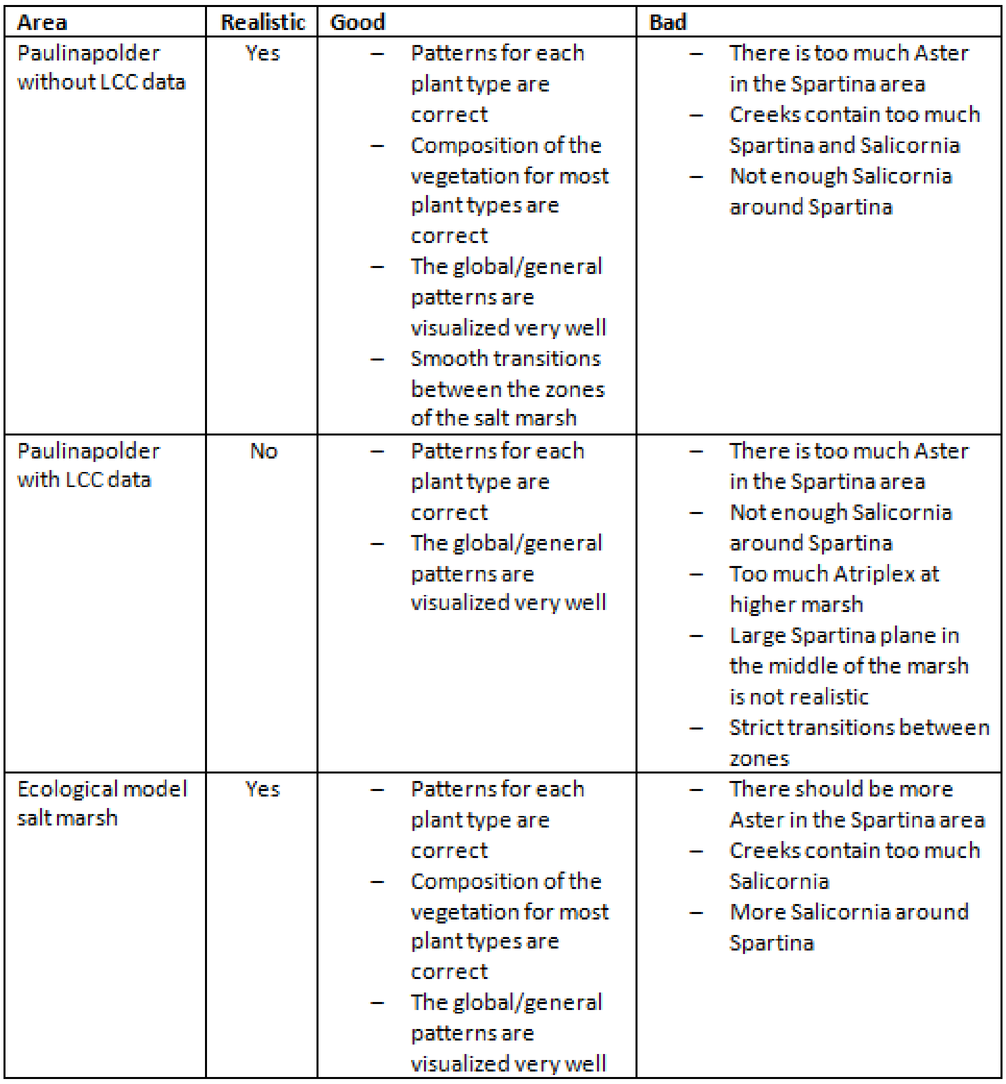
In general, the error in both real and test cases are very low. This means that algorithm is able to correctly map the input composition values to a plant distribution which compositions corresponds with the input composition. Currently, it is only tested whether we the composition values are correctly mapped. For completeness, this should also be done for the shape metrics input. Normally shape metrics are calculated based on a grid map. There is no clear method for calculating shape metrics for a point distribution. This should be further investigated in future research projects.

4.3.2. EXPERT VALIDATION

Expert validation is used to judge the realism of the results obtained with the plant placement algorithm. These results are discussed with an ecologist Johan van de Koppel from NIOZ. He is involved in creating dynamic ecological models to predict future areas. The ecological model that was used in the third test case was inspired on a model that Johan has worked on. In addition, he is familiar with the Paulinapolder. The purpose of the discussion with Johan was to validate whether the generated point distribution is realistic with respect to the location of each plant of every plant type and with respect to the patterns exhibited by each plant type. The main questions that were asked during the discussion and a summary of the answers are included in Appendix B. The most important conclusions for each test case are summarized in table 4.9.

The Paulina polder without using LCC data and the salt marsh based on an ecological model were judged as realistic. However, the point distribution for the Paulinapolder with discrete LCC data was judges as not realistic.

Table 4.9: Overview of expert validation of the test results



Area	Realistic	Good	Bad
Paulinapolder without LCC data	Yes	– Patterns for each plant type are correct – Composition of the vegetation for most plant types are correct – The global/general patterns are visualized very well – Smooth transitions between the zones of the salt marsh	– There is too much Aster in the Spartina area – Creeks contain too much Spartina and Salicornia – Not enough Salicornia around Spartina
Paulinapolder with LCC data	No	– Patterns for each plant type are correct – The global/general patterns are visualized very well	– There is too much Aster in the Spartina area – Not enough Salicornia around Spartina – Too much Atriplex at higher marsh – Large Spartina plane in the middle of the marsh is not realistic – Strict transitions between zones
Ecological model salt marsh	Yes	– Patterns for each plant type are correct – Composition of the vegetation for most plant types are correct – The global/general patterns are visualized very well	– There should be more Aster in the Spartina area – Creeks contain too much Salicornia – More Salicornia around Spartina

PAULINAPOLDER WITH LCC DATA

The main reason that the plant distribution of the Paulinapolder was judged not realistic is because of the LCC data. The LCC data has very strict borders between the different marsh zones, and in each zone different plant types grow which due to this the results has rigid borders between the zones. The usage of this LCC map resulted in a large Spartina with Aster area in the middle of the map which is not realsitic for that area. In addition, on the higher marsh parts Atriplex received too much coverage, because of the strict categories in the LCC map. Both areas are shown in figure 4.14. These problems could possibly be partly solved if more different plant types are allowed in each category of the LCC map by adding the plant types of the neighboring zones with a certain probability. Another possibility would be to look at how the plant types are divided in the different LCC groups, and how that influences the composition statistics at the different areas.

The strict transitions between each zone could also be improved upon by adding an extra processing step in the planet placement algorithm for discrete LCC data. A possibility would be to use a neighborhood kernel on the LCC data [49] to remove the strict transitions between each zone. A neighborhood kernel smooths the transitions by averaging the values near the borders of the zones. However, in this case it might have limited effect, because each zone is very large. Values near the borders will be averages, but values deeper in each zone are not changed. The zones are large, because the LCC map is derived from low-resolution aerial images. This means that the accuracy of the data is low in comparison with the other data that is provided, which can explain the issues with the LCC data. The main conclusion from the test case with LCC data set is that extra preprocessing steps are necessary for LCC data. Thus, the discrete LCC data should be processed to a

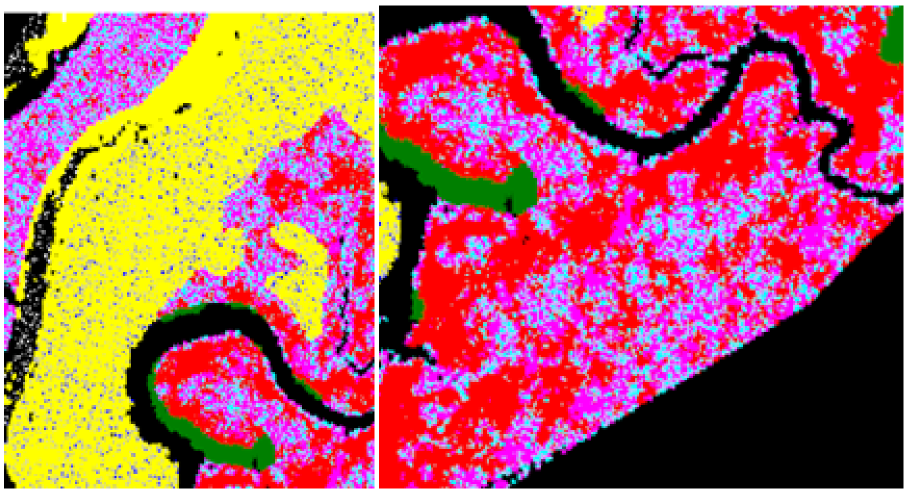


Figure 4.14: Picture of the large Spartina (yellow) area and picture where Atriplex (red) took over too much of the area. Both examples from our point distribution of the LCC case. Both images show non-realistic results. Other plants in the images are Elymus (green), Aster (blue), Salicornia (white), Limonium (teal), Artemisa (pink).

continuous LCC map for smoother transitions of zones. This is something that should be further investigated in the future. Further, an LCC map with improved quality would probably improve the results as well.

Paulinapolder without LCC data

The plant distribution for the Paulinapolder without LCC data was judged realistic by the expert. In general, the patterns of the complete vegetation population and the patterns of each individual were considered as realistic. Figure 4.15 shows a photo of some global patterns of the marsh. The patterns of the generated plant distribution appear to match with the patterns of the photo.

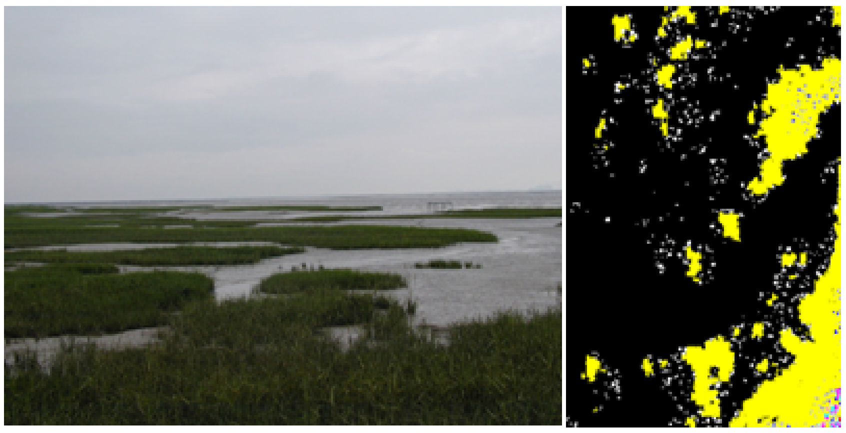


Figure 4.15: Picture of global and local patterns of Spartina, and next to it patterns generated of Spartina (yellow) and some Salicornia (white) by the algorithm

The location in the marsh for most plant types was judges as correct. There were only a few minor issues. There was too much Aster placed between Spartina in the Paulinapolder, and the another issue was that Spartina and Salicornia were placed too far land inwards in the creeks or were placed in the middle of the creeks. Both issues are visualized in figure 4.16.

The cause of both issues is the statistical composition data input. The composition data input used was mainly dependent on a height map. For example, Spartina and Salicornia are both placed at lower grounds. The plant placement algorithm follows composition statistics correctly as we have seen in the previous section. The height of creeks meet height requirement for both Spartina and Salicornia which means that Spartina and Salicornia is placed in the creeks. Thus, to make sure the plants are placed correctly, additional data is necessary to enable constraints on the creeks, so that no, or less, plants can be placed there. Possible

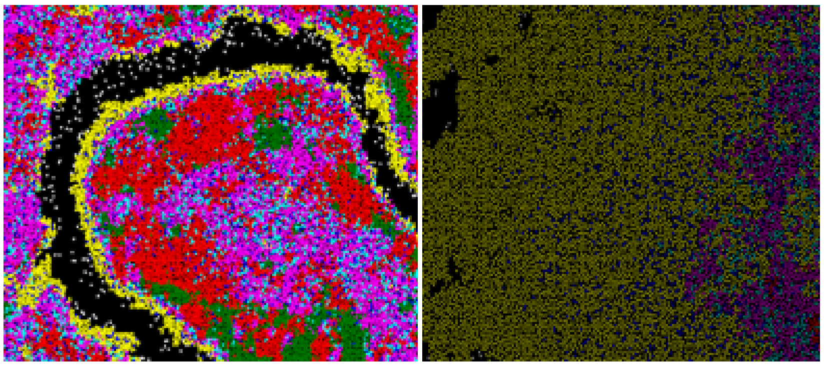


Figure 4.16: Issues in the plant distribution: creeks and Aster (blue). Creeks contain too much Spartina (yellow) and Salicornia (white) in the left image. On the right image there is too much Aster (blue) between Spartina (yellow).

data sources could be water speed or length of the creeks. The Aster case is also, caused by the height statistic relation. In the Paulinapolder, Aster is not often found together with Spartina, because the area where Spartina grows is very salt. Locations of Aster can be better predicted when salt data is combined with the coverage data of Aster.

To show that in general the locations of each plant type are correct, figure 4.17 is included. This figure shows a photo where Spartina patches are growing at lower ground with Salicornia (small white/brown plants in the front) around it, on higher ground Aster (white flowers) is growing. This is also visible in the plant distribution generated by the plant placement algorithm.

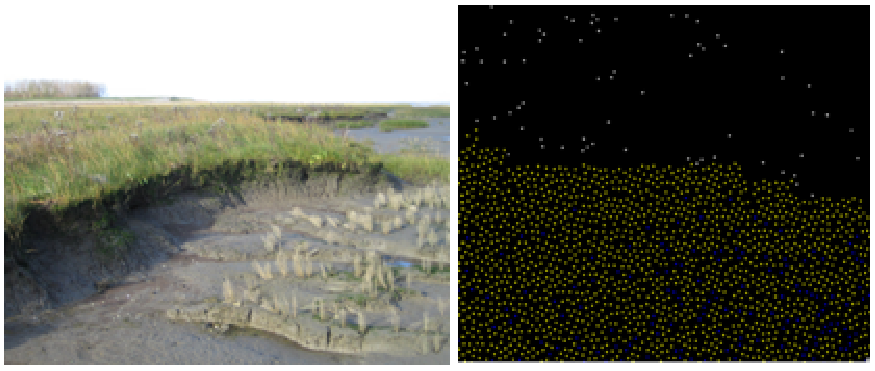


Figure 4.17: Spartina (yellow), Aster (blue), Salicornia (white) and example picture. Similar patterns and composition

The transitions between the zones were judged as realistic as well. Each zone of the salt marsh could be clearly identified by the expert. Figure 4.18 shows an example of the transitions between the zones. In the back, the large Spartina patches are visible, followed by Aster (white large flowers), and then Aster is mixed with plant types growing higher on the marsh.

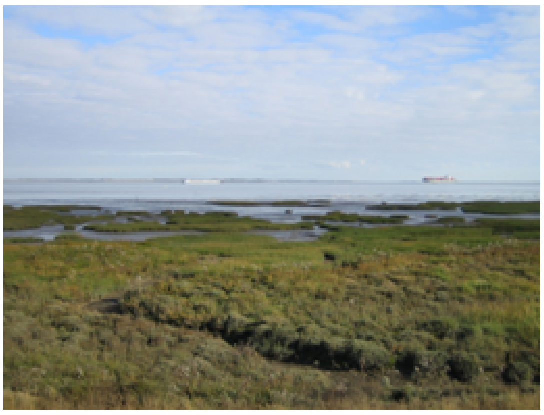


Figure 4.18: A photo that shows the transitions between the different zones

Ecological model-based salt marsh

The plant distribution based on the output of an ecological model was judged realistic. Comments that were made for the Paulinapolder without LCC data also apply to this area. There is one difference, because more Aster should be growing together with Spartina in this area. There should be more Aster placed in the plant distribution, because the area that the ecological model represents, is not salt, which allows Aster to grow much more in the area with Spartina. The issue of Salicornia placement in creeks also applies to this area. Figure 4.19 shows an area of the point distribution where Aster is placed. In that area, more Aster should be placed so that about 50 percent of that area is covered with Aster.

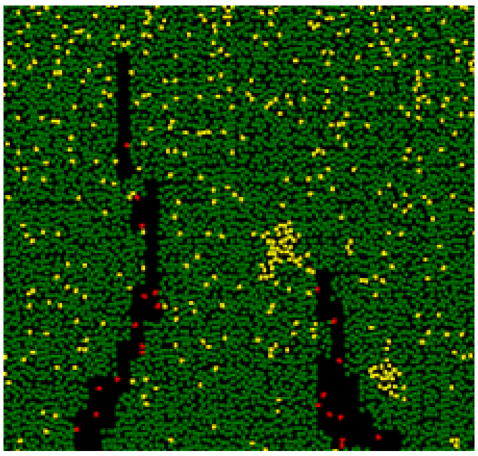


Figure 4.19: Aster placement in the ecological model case. There should be more Aster placed in that area, around 50 percent Aster and 50 percent Spartina is more realistic. In this example, Aster is yellow. Spartina is green, and Salicornia is red.

General comments

The expert also made some general comments about the plant distribution. The first one was related to having different sizes of plants. Figure 4.17 shows that Salicornia is much smaller than Spartina and Aster. This was not taken into account during the generation of the results. It is possible with the current plant placement algorithm to assign Salicornia a lower plant size by a separate plant level for Salicornia. This also would result more Salicornia around the Spartina patches if the composition values remain the same, because more smaller plant positions are available for Salicornia. The expert also mentioned that there should be more Salicornia around the Spartina patches. It is important to note the sizing of the points depends on how you want represent a plant in for example a 3D visualization. Depending on the plant type, a single plant positions could be used for a patch of plants or a single large plant. This means that smaller plants could be given in the same size in the algorithm, by giving a patch of small plants the same size as one individual large plant.

Johan made the interesting point that the plant placement algorithm could be used by, for example, ecologist to validate their datasets, because the algorithm correctly maps the input coverage to a point distribution. Datasets of ecologist can be used as input for the algorithm and the output distribution can be used to check whether the input data gives the expected distribution. Based on this mapping, they can judge if their data is correct or that certain data is missing. This also emphasizes one of strong points the plant placement algorithm: it is logical, the data that is given as input matches with the output. This makes it easy to check the influence of the different data sources on for example the composition.

Finally, the results were investigated by the author as well. It was noticed that sometimes the global patterns of the distributions can have a square outlook. This is most visible in the point distribution based on the ecological model data. Points are generated in a tile when it is known that vegetation grows there. This means that sudden changes between vegetation and no vegetation tiles occur. This is especially visible when the tiles for the point generation are relatively large. A solution for this could be to remove random points of the tiles that share a border with tiles that do not contain vegetation after the point generation. However, this is not supported by the current algorithm and it will require future research. It is recommended that additional possibilities are considered that are data-driven instead.

4.4. DISCUSSION

In this section, the plant placement algorithm is discussed using the validation results from the previous section in combination with the criteria introduced in chapter 3.1. For each criteria, it is discussed whether the algorithm fulfills it. Criteria are spatial data-driven, adaptive, and effective. Spatial-data driven means that the algorithm can handle different spatial data sources; the input information is correctly mapped in the final point distribution. Adaptive means that it can handle different plant types and different plant levels. Effective means that the results of the algorithm should be realistic. Each of these criteria is discussed separately:

Spatial data-driven: The algorithm is fully dependent on spatial data sources. The algorithm is able to work with different data sources by using the composition or pattern information already present in the map or by connecting to it by means of statistical information, e.g. connecting composition values based on a height relationship. This relationship could be based on continuous values, like values in a height map, or based on geometrical relations like distance or orientation to a certain object on a map. The compositions values of each map are combined to one value per point for each plant type. In chapter 4.3.1., we have seen that these combined composition values are correctly mapped in the output distribution. The validation of this mapping could not be done for the shape metrics input. The issues in the generated distributions of the real data examples can be explained by the data input, because the current spatial data input (e.g. a height map) was not sufficient for applying all the relevant constraints on the placement of each plant type. For example, salt data was necessary to have to correctly place the correct amount of Aster in the generated distributions. Further, we have seen the algorithm is not able to use LCC data correctly to obtain realistic results. This is demonstrated with the Paulinapolder case that uses LCC data. Plants were generated at wrong positions and and strict boundaries were generated in the marsh which should have been gradual. In future research project, this should be improved. Another possibility is to improve the plant placement algorithm, so that it takes additional processing steps for LCC data by, for example, smoothing the transitions between the different categories of the map.

Further, the algorithm is able to generate future areas based on data provided by dynamic ecological models. Besides the input of continuous or discrete maps, the algorithm can handle existing point data as is shown in chapter 3. This means that the output of the current plant detection techniques can be used as input for plant placement algorithm. The plant placement algorithm is then able to assign plant types to the detected plant positions or generate additional plant positions for groundcover plants. Currently, some data input is not supported by the algorithm, like LiDAR point clouds. This kinds of data first needs to be processed by detection methods. Due to time constraints, it was not possible to include data sources that represent information about the direction of patterns. Sometimes environmental processes can influence the direction in which patterns grow as can be seen in figure 4.20.

To sum up this criterion: any data source can be included as input to the algorithm as long as it can provide information about the composition of a plant type. It is also possible to include existing point data. Composition values are correctly used in the algorithm, which means that each map can have its own significant influence on the plant placement process. The processing of discrete LCC data needs extra processing steps for the data to be handled properly by the algorithm. Further, it is important to look at the quality of

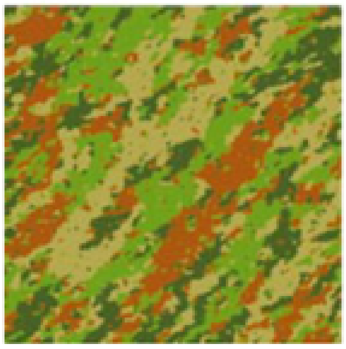


Figure 4.20: Patterns are oriented in a certain direction.

the input data when including different data sources. Low accuracy or resolution of the data can result in unrealistic generated distribution. This was the problem for the LCC data case.

Adaptive: The generated results with different data sources show that the algorithm is adaptive. However, since both test areas were salt marshes the test cases did not represent completely different environments. Chapter 3 shows that there is the possibility of including different sizes of plants for forest-like environments, though; this has not been tested with real data. It is possible to generate different kinds of patterns for each plant type, which is clearly visible in the results described in chapter 4.1. Further, testing is necessary for different environments.

Effective: An expert found two of the three test cases realistic. The discussion of the validation shows that the quality of the input data has a large effect on the realism of the generated plant distribution. There were minor issues in the realistic plant distributions. These issues could be solved by using data of better quality or by including data, that applies extra constraints on the plant distribution. The realism of the generated plant distribution can be improved by adding additional pre-processing steps to the algorithm to handle discrete LCC data better and to decrease the square-like global patterns of the vegetation distribution. These additional processing steps are a topic for future research.

5
Conclusion and Future Work

This chapter summarizes the conclusions of this research project and presents ideas for future work First, the research questions that were introduced in the first chapter are answered. Then, the conclusions are discussed, and recommendations are made for future work.

5.1. Research questions

To answer the main research questions, the following sub questions needs to be answered:

1. *What are the current techniques to determine plant positions?*

   Through a literature study, current techniques for determining plant positions are found in the areas of plant detection, procedural ecosystem generation, and dynamic ecological models. This research shows that the current plant detection techniques are not sufficient to generate a realistic plant distribution for both existing and future areas. These techniques need high-resolution aerial or LiDAR data and can only detect large plants, like trees. Small ground cover plants cannot be detected, because they cannot be distinguished from the background or other plants. Further, because the detection techniques are dependent on aerial images and/or LiDAR data they cannot be used for future areas. Procedural ecosystem generation techniques are able to generate plant positions when no prior information is available of where the plants are located. These techniques are not able to use spatial data to correctly generate plant types for each plant position. Ecological models can generate realistic point positions for future areas, but these models are not commonly available, because the creation of these models is very difficult and requires large amounts of ecological knowledge. Also, these models often focus on the position generation of just a single plant type. There are other ecological models, but these do not focus on the generation of positions of plants which limits their relevance to this research.

2. *What kind of spatial data is necessary to generate and classify plant positions of different size, for both future and existing areas, and how does the data influence the results?*

   This research shows that aerial images or LiDAR data cannot be used alone to determine where plants are placed. Small plants are not identified and neither can it be used for future areas. To obtain data for future areas the output of dynamic ecological models has to be used. These models generate maps containing information about height, coverage, and biomass. This is data similar to what is available for existing areas.

   In addition, it is demonstrated that other parameters have to be used to determine plant positions, instead of using parameters derived from images or point clouds (e.g. intensity). Two other parameters are necessary: composition and shape metrics. Composition gives information about the location of plant types and shape metrics about the patterns of each plant type. This means that spatial data have to be used that is able to represent information that is or can be related to composition or shape metric data. This could be LCC maps that already contain composition information or (a)biotic maps which are used as a factor for statistical composition functions. This requires that a relation or function is know of the (a)biotic factor with the composition or coverage of plant types.

This research demonstrates that the algorithm correctly processes the information of the spatial data. Each spatial data sources connected to composition data can have a significant influence on the result, because the input composition values are maintained during the plant distribution generation. Each map that used, restricts where one or multiple plant types can be placed. Further, discrete LCC data should be handled as a special case for the generation of plant positions. The reason for this is that LCC data creates strict boundaries between plant zones or types, which create sharp transitions between plant types in the plant distribution. Therefore, it is proposed that LCC data is handled differently than other data and that is translated beforehand to a continuous LCC map to remove the strict borders between zones. This step is necessary for all the maps that contains discrete composition information. For the current algorithm, the usage of this kind of data is not recommended unless discrete LCC data is manually processed to a continuous LCC map.

Finally, the quality of the spatial data is important for the generation of realistic plant distributions. The inclusion of a low quality data set with other data sets can make the whole distribution unrealistic. It is preferred that the maps are from equal quality, so that the algorithm can adjust its parameters automatically to match with the quality of the maps. In addition, statistical composition values from literature have to be checked and used carefully, because often such data input is not calculated from the target area, but calculated from a similar area as the target area. However, as this research the usage of such can give problems, because there can be a difference in composition for similar environments like in the salt marsh case between the Oosterschelde and Westerschelde.

3. *How can plant point positions for different plant types be generated based on spatial data?*

   The best approach for generating plant point positions in combination with spatial data is to use the PDD generation method with the Wang tiling technique. This point generation techniques was modified in three aspects to be suitable for the purposes of this research: Wang corners tiles were used instead of Wang border tiles. This solved the corner problem that the current procedural point generation techniques have for the generation of ecosystems. The corner problem is the problem that points can overlap in the area where the Wang border tiles touch each other after the point generation method is finished. The second aspect introduces the concept of non-bias multiclass point generation, which means that large points can be used seamlessly by smaller plants without having space artefacts. The third modification gives the option to integrate existing spatial data points in the point generation algorithm. Existing points are integrated by first generating a regular point distribution which are then matched with the existing point data set. This approach was chosen adding existing points directly in the PDD with Wang tiling technique, breaks the Wang tiling principle. Instead, an existing point is matched with the closest point of the generated point distribution of the algorithm. A greedy approach is used for this, which is not yet fully developed and should be improved upon in the future. Finally, spatial data also decides where possible point locations are generated. This information is derived from aerial images, biomass, coverage, or LCC maps.

4. *How can we determine where to place which plant type based on spatial data?*

   To be able to determine where a plant type should be placed based on spatial data, a method from neutral landscape modelling was used. This method was modified, so that it is able to process non-static composition values which are obtained from spatial maps. Also, that it works with point input data. The third modification is that the method is able to classify plant types from different plant levels and that these plant levels can influence each other. During the validation, it was shown that these modifications have been successfully used, because the composition values are mapped correctly and that an expert judged most of the real data cases as realistic. The main idea behind correctly mapping the composition values, was to assign a separate threshold value to each point based on their fractal and coverage value and to update coverage values separately for each point.

Next, the main research question can be answered:

- *How can plant positions be generated and classified from spatial data of existing and future areas to obtain a realistic plant distribution?*

  The research proposes a 4-step method to generate a realistic plant distribution based on spatial data. In the first step, spatial data is processed to determine where vegetation in general is located. In the second step, all possible point locations are generated with the new point generation method based on the information where vegetation grows and where it does not. In the third step, for each point the composition and shape information derived from spatial data have to be combined. This is done by first calculating the composition values of each input map for every point. The composition values connected to each point are then combined to a single composition value for each plant type based on the idea that each value is a limiting factor for that plant type. Finally, points are classified with the new classification method, which is improved version of method from neutral landscape modelling. The improved method is able to handle information from spatial data sources. Based on the validation by an expert, it can be concluded that the proposed plant placement algorithm in this research is able to generate realistic plant distributions and that it is able to correctly map spatial data input to an output plant distribution. In addition, it is possible to generate plant distributions for future areas, because ecological models are used that can generate maps of these areas.

5.2. CONCLUSIONS

The main contribution of this research is that it introduces a new method for generating plant point distributions from spatial data. This method is able to generate plant locations for both groundcover plants as well as plants in future areas. Current techniques in the geo-domain that focus on plant position generation are not able to do this. The disadvantage of the method is that it does not generate plant positions that exactly match with the real environment. Instead, it generates positions that approximately match with the real world and that plant types have similar patterns as in the real environment. This approximation is justified, because in case of for example realistic 3D visualization of natural environments exact matching points are not necessary and because exact positions cannot be determined for groundcover plants and future areas.

A second contribution of this research is that the new method uses different types of maps to generate plant positions. To obtain data for future areas, dynamic ecological models are used, which can generate the needed information for future areas. Also, the proposed method is not limited to the usage of high resolution aerial images or LiDAR point clouds. Instead, lower resolution maps can be used, that contain information or able to connect information about either composition or shape metrics for single or multiple plant types.

This research demonstrates that it is possible to generate a realistic plant distribution for real environment based on spatial data where different plant types are included. This is achieved by integrating concepts from ecology and procedural modelling. These had to be modified to work properly with spatial data and different types of plants. The procedural point generates uses a new concept of non-biased multiclass point generation and it is able to integrate existing point data. The point classification uses a modified technique from neutral landscape modelling to properly map the non-static information provided by spatial data and to work with multiple plant levels.

Another important contribution of this research is that it proposes a method that does not replace any of the current detection techniques, but can be used to improve upon them. Current state-of-the-art plant detection techniques can be used as normally to detect plant positions for existing environments, but the output plant positions can be used as input for the newly developed method. This way it is possible to enrich the data generated by the current plant detection techniques by including, for example, the positions of small groundcover plants or the plant type of each detected plant. Possible, characteristics (e.g. height or size) of the plant measured by the plant detection techniques can be used to improve the classification of each detected plant.

The final contribution of this research is that the developed method is able to validate (spatial) ecological datasets. An ecologist can give a map as input along with measured statistical composition data that is related to the input map. Then, from the results of the plant placement method the ecologist is able to see whether the input data of the target area is as expected or that there are errors in the measurements of the data. It is also possible to see whether the spatial data input captures all the information to determine realistic positions for a certain plant type. During the validation of the results, the expert was able to see that Aster was not correctly placed, because salt information was missing. This kind of validation is possible, because the proposed method is mapping input composition based on spatial data correctly to a realistic plant distribution having the same input composition values.

5.3. FUTURE WORK

Besides the contributions of this method there is still room for improvements. Recommendations for future work are divided into three categories: improvements in the algorithm itself, validation improvements, and improvements for application areas.

Algorithm

In the algorithm, it is not possible to give orientation to the patterns in an environment. Thus, that the patterns of the plants grow in a certain direction. This requires the usage of additional spatial data sources and an improvement in the fractal algorithm. It must be investigated which spatial data sources can be used to represent the orientation of patterns for different plant types. Possible sources could be data give information about water speed and the direction that water flows in that area. The fractal algorithm needs to be improved, because that algorithm controls the shape of the patterns for each plant type. It must be investigated how the information from the new spatial data sources can be connected with the fractal algorithm. Initial tests not included in this research report suggest that this must be done by adjusting the frequency parameter in the fractal generation.

Several improvements could be made in the point generation component of the algorithm. First, the current PDD generation is not optimal, because sometimes point are not optimal distributed over the area. This means that certain points have a significant larger distance to their neighboring points. This could be improved by using Lloyd relaxation method [50] which is an iterative algorithm that moves points, so that the distance between each point is in the end more or less equal. Second, the usage of existing point data in combination with the point generation can be improved. Currently, a simple greedy algorithm is applied to match the existing points with the generated points. For future research, it must be investigates what kind of point matching algorithm is preferred to obtain more accurate matching results. Another option might be to test whether a GPU-based PDD generation technique can be used [51]. Such a technique can generate in certain cases more than a million points per second. This would give the option to remove the Wang tiling method and generate points around the existing data input. For future research, it can be explored if such a technique is feasible in combination with the spatial data input and the currect extensions.

The third improvement is in the usage of discrete LCC data by the algorithm. In the validation phase of this research, it became clear that LCC data requires some additional processing steps so that it can be used to generate realistic results. A possible extra processing step is to apply a neighbourhood kernel on these LCC maps to generate smooth transition between the categories. In the future, it must be studied which kind of additional processing steps are available and necessary. In addition, it must be investigated what the exact causes are that a LCC map did not give realistic results, because the quality of the LCC map may also play an important role into this problem.

The final improvement for the algorithm is to optimize the performance and memory management of the algorithm to be able to generate larger plant distribution for larger environments. The focus of this research was not to implement a fully optimized algorithm, but this is an interesting option to investigate when larger plant distributions are required.

Validation

In this research, the results were tested through statistical validation of the composition of each generated plant distribution and through expert validation to investigate the realism of the distributions. For future studies, it must be investigated whether the plant distribution return similar shape metric values as what is given as input. This was not done in the current research, because normally the methods to calculate shape metric values assume that the patterns are on a grid and not in point format. Thus, it must also be investigated how to calculate shape metric values from a point set.

Another improvement in the validation is to perform additional tests for different types of environments. Currently, tests are performed only for salt marshes, so it must be investigated how the algorithm perform with other types of environments. Probably, the most interesting environments to test in the future, are environments where there is a combination of trees and groundcover plants. This gives the possibility to test the concept of plant levels in the algorithm.

Application areas

The main motivation for this research and the research institute NIOZ where I partly did an internship, is to use the plant distribution for the 3D visualization of natural environments. For future research, it is a possibility to generate a 3D visualization of the plant distributions in this research that were judged realistic by the expert. Also, a 3D visualization could improve the validation of the results of the algorithm, because plants are represented with an actual model instead of a point with a colour. This can make it easier to perform expert tests with ecologists.

Another application area is the plant detection domain. The conclusions of this research already showed that there is a possibility to enrich the results of these techniques, because small plants are not detected and positions often miss plant type information. Plant detection techniques are however able to generate other properties for the detected plants, like height, and crown area. An option for future research is to use these properties in the plant placement algorithm as additional an information source in the point classification part. In addition, these properties could be used for more advanced calculation such as the total amount of sunlight that each plant receives. These properties give the possibility to generate a more accurate classification of each plant position.

Additionally, for future research it must be investigated how accurate the algorithm can enrich the current plant datasets obtained through advanced plant detection techniques by performing several case studies. The outcome of these case studies can be interesting for companies, like Bluesky (bluesky-world.com), which are currently creating tree maps of whole countries. Their current maps only contain information about the position, height, and crown area of each tree. An interesting option for these maps would be to identify if the algorithm can accurately enrich them and to identify if the added data is useful for the current applications of the tree maps. In addition, it must be investigated how to apply this algorithm efficiently on tree maps, because they are very large datasets. It would be interesting to research if this data could be enriched directly in a spatial database by creating an extension that uses this algorithm.

The area of the generation of LCC maps can also benefit from this research in the future. In principle, the algorithm generates a very detailed LCC map. It is worth investigating how to generalize the generated point distribution to create a LCC map of a desired resolution and certain class level. For example, how can the results of the first test case in this research be converted to a LCC map with categories of the different plant zones of the salt marsh. An important factor for this is whether the map should be continuous or discrete. Not only could it be used for generating LCC maps, it could also be possible to update existing LCC maps with additional details by for example indicating per plant zone the composition or information about the patterns. Also, it could be possible to generate a higher resolution map or translate a discrete map to a continuous map.

Further, the algorithm has to possibility to identify data quality issues for ecologists as has been discussed in the conclusions. For a future research project a tool could be developed around the algorithm to visualize to the user which spatial data input is responsible for the placement of plant in a certain area. Such tool gives thus insight in how the spatial data is processed and in how each processed data influences the location of plants in the complete plant distribution. Instead of building a complete new tool, it could also be possible to create an extension for a GIS software so that the results can be analyzed directly.

Finally, the algorithm is able to generate plant distributions for future areas. Therefore, the usage of this algorithm should be investigated for the planning community. Planners could investigate plant distribution for future areas by providing different data or changing some parameters. For example, the algorithm has to possibility to process object maps to influence the plant distributions. A planner could provide a map of a road that is going to be build in a rural area and the algorithm could show how the plant distribution reacts on this. By changing the input with another possible future road, it could be investigated which road is the most beneficial for the nature in that area.

Bibliography

[1] C. J. Pettit, C. M. Raymond, B. A. Bryan, and H. Lewis, *Identifying strengths and weaknesses of landscape visualisation for effective communication of future alternatives,* Landscape and Urban Planning **100**, 231 (2011).

[2] R. van Lammeren, J. Houtkamp, S. Colijn, M. Hilferink, and A. Bouwman, *Affective appraisal of 3d land use visualization,* Computers, environment and urban systems **34**, 465 (2010).

[3] J. Borsboom-van Beurden, R. Van Lammeren, T. Hoogerwerf, and A. Bouwman, *Linking land use modelling and 3d visualisation,* in *Innovations in design & decision support systems in architecture and urban planning* (Springer, 2006) pp. 85–101.

[4] C. Pettit, W. Cartwright, M. Berry, *et al.*, *Geographical visualization: A participatory planning support tool for imagining landscape futures,* Applied GIS **2**, 22 (2006).

[5] M. Schaafsma, R. Brouwer, A. Gilbert, *et al.*, *Hedwigepolder: publieke waardering van natuurontwikkeling in zeeland,* H2O: Tijdschrift voor Watervoorziening en Waterbeheer **43** (2010).

[6] Z. Reljic, M. Sawada, J. Poitevin, and G. Saunders, *Integrating gis and 3d visualization for dynamic landscape representation in canada's national parks,* (2005).

[7] R. Van Lammeren, M. Hilferink, A. Bergsma, and M. Van Beek, *Google Earth based visualization of Sustainable Outlook (GESO)*, Tech. Rep. (CGI-2008-03, 2008).

[8] P. Ghadirian and I. D. Bishop, *Integration of augmented reality and gis: A new approach to realistic landscape visualisation,* Landscape and Urban Planning **86**, 226 (2008).

[9] J. Hempenius, *Exploring 3D visualization of vegetation*, Master's thesis, Wageningen University (2010).

[10] M. Weidenbach and R. de Kok, *Report on Automatic Detection of Individual Trees and Stand Openings from Quickbird Satellite Imagery of South African Plantations*, Tech. Rep. (Landconsult, 2008).

[11] M. Rahman and B. Gorte, *Individual tree detection based on densities of high points of high resolution airborne lidar,* GEOBIA , 350 (2008).

[12] J. Secord and A. Zakhor, *Tree detection in urban regions using aerial lidar and image data,* Geoscience and Remote Sensing Letters, IEEE **4**, 196 (2007).

[13] B. Ganster, *Improving Usability in Procedural Modeling,* Ph.D. thesis, Universitäts-und Landesbibliothek Bonn (2013).

[14] R. L. Saunders, *Realistic terrain synthesis using genetic algorithms*, Ph.D. thesis, Texas A&M University (2006).

[15] H. Zhou, J. Sun, G. Turk, and J. M. Rehg, *Terrain synthesis from digital elevation models,* Visualization and Computer Graphics, IEEE Transactions on **13**, 834 (2007).

[16] I. Parberry, *Designer Worlds: Procedural Generation of Infinite Terrain from USGS Elevation Data Designer Worlds: Procedural Generation of Infinite Terrain from USGS Elevation Data*, Tech. Rep. (Tech. rep., University of Northern Texas, 2013).

[17] P. Musialski, P. Wonka, D. G. Aliaga, M. Wimmer, L. Gool, and W. Purgathofer, *A survey of urban reconstruction,* in *Computer Graphics Forum*, Vol. 32 (Wiley Online Library, 2013) pp. 146–177.

[18] D. G. Aliaga, C. A. Vanegas, and B. Beneš, *Interactive example-based urban layout synthesis,* in *ACM Transactions on Graphics (TOG)*, Vol. 27 (ACM, 2008) p. 160.

[19] S. MÜLLER ARISONA, C. ZHONG, X. HUANG, and R. QIN, *Increasing detail of 3d models through combined photogrammetric and procedural modelling,* Geo-spatial Information Science **16**, 45 (2013).

[20] Y. I. Parish and P. Müller, *Procedural modeling of cities,* in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (ACM, 2001) pp. 301–308.

[21] J. Hammes, *Modeling of ecosystems as a data source for real-time terrain rendering,* in *Digital Earth Moving* (Springer, 2001) pp. 98–111.

[22] M. Hussain, D. Chen, A. Cheng, H. Wei, and D. Stanley, *Change detection from remotely sensed images: From pixel-based to object-based approaches,* ISPRS Journal of Photogrammetry and Remote Sensing **80**, 91 (2013).

[23] M. A. Alizadeh Khameneh, *Tree detection and species identification using lidar data,* (2013).

[24] O. Deussen, P. Hanrahan, B. Lintermann, R. Měch, M. Pharr, and P. Prusinkiewicz, *Realistic modeling and rendering of plant ecosystems,* in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (ACM, 1998) pp. 275–286.

[25] B. Lane, P. Prusinkiewicz, *et al.*, *Generating spatial distributions for multilevel models of plant communities,* in *Graphics Interface* (Citeseer, 2002) pp. 69–80.

[26] W. van Haevre, *Realism In Environment Sensitive Plant Models and their Animation*, Ph.D. thesis, University Hasselt (2007).

[27] B. Beneš, *A stable modeling of large plant ecosystems,* in *Proceedings of the International Conference on Computer Vision and Graphics*, pp. 94–101.

[28] B. Beneš, N. Andrysco, and O. Št'ava, *Interactive modeling of virtual ecosystems,* in *Proceedings of the Fifth Eurographics conference on Natural Phenomena* (Eurographics Association, 2009) pp. 9–16.

[29] E. Ch'ng, *An artificial life-based vegetation modelling approach for biodiversity research,* Green Technologies: Concepts, Methodologies, Tools and Applications , 417 (2010).

[30] M. Alsweis and O. Deussen, *Wang-Tiles for the simulation and visualization of plant competition* (Springer, 2006).

[31] A. Lagae, *Tile-based methods in computer graphics,* (2007).

[32] M. Weier, A. Hinkenjann, G. Demme, and P. Slusallek, *Generating and rendering large scale tiled plant populations,* Journal of Virtual Reality and Broadcasting **10** (2013).

[33] S. Saura and J. Martínez-Millán, *Landscape patterns simulation with a modified random clusters method,* Landscape ecology **15**, 661 (2000).

[34] W. W. Hargrove, F. M. Hoffman, and P. M. Schwartz, *A fractal landscape realizer for generating synthetic maps,* Conservation Ecology **6**, 2 (2002).

[35] J. Johnson, *A study of generative systems for modeling natural phenomena*, Master's thesis.

[36] J. Molofsky and J. D. Bever, *A new kind of ecology?* BioScience **54**, 440 (2004).

[37] J. Conway, *The game of life,* Scientific American **223**, 4 (1970).

[38] S. Temmerman, T. Bouma, J. Van de Koppel, D. Van der Wal, M. De Vries, and P. Herman, *Vegetation causes channel erosion in a tidal landscape,* Geology **35**, 631 (2007).

[39] Q.-X. Liu, P. M. Herman, W. M. Mooij, J. Huisman, M. Scheffer, H. Olff, and J. van de Koppel, *Pattern formation at multiple spatial scales drives the resilience of mussel bed ecosystems,* Nature communications **5** (2014).

[40] R. Tönjes, *3d reconstruction of objects from aerial images using a gis,* International Archives of Photogrammetry and Remote Sensing **32**, 140 (1997).

[41] C. Wang, T. R. Wan, and I. J. Palmer, *Automatic reconstruction of 3d environment using real terrain data and satellite images,* Intelligent Automation & Soft Computing **18**, 49 (2012).

[42] J. de Leeuw, L. P. Apon, P. M. Herman, W. de Munck, and W. G. Beeftink, *The response of salt marsh vegetation to tidal reduction caused by the Oosterschelde storm-surge barrier* (Springer, 1994).

[43] M. Rietkerk and J. Van de Koppel, *Regular pattern formation in real ecosystems,* Trends in Ecology & Evolution **23**, 169 (2008).

[44] K. Perlin, *Improving noise,* in *ACM Transactions on Graphics (TOG)*, Vol. 21 (ACM, 2002) pp. 681–682.

[45] S. Gustavson, *Simplex noise demystified,* Linköping University, Linköping, Sweden, Research Report (2005).

[46] J. van de Koppel, T. J. Bouma, and P. M. Herman, *The influence of local-and landscape-scale processes on spatial self-organization in estuarine ecosystems,* The Journal of experimental biology **215**, 962 (2012).

[47] J. Reitsma, *Toelichting op de vegetatiekartering westerschelde 2004 op basis van false colour-luchtfoto's,* (2006).

[48] C. Schwarz, *Implications of biogeomorphic feedbacks on tidal landscape development*, Ph.D. thesis, Radboud University Nijmegen (2014).

[49] W. D. Wells, *Generating enhanced natural environments and terrain for interactive combat simulations (genetics),* in *Proceedings of the ACM symposium on Virtual reality software and technology* (ACM, 2005) pp. 184–191.

[50] S. Lloyd, *Least squares quantization in pcm,* Information Theory, IEEE Transactions on **28**, 129 (1982).

[51] L.-Y. Wei, *Parallel poisson disk sampling,* in *ACM Transactions on Graphics (TOG)*, Vol. 27 (ACM, 2008) p. 20.

A
Reflection

This thesis proposes a plant placement algorithm, which can translate different data sources to generate a realistic plant distribution. A plant distribution can be generated for existing and future areas. As well, it is possible to generate both small and large plants.

This research took place over the course of 15 months. The reason that it took longer than the normal 9 months, is that I am doing two masters with a partially combined graduation project. Also, at the beginning of the thesis project I still had to finish some courses for both Masters and therefore I was only able to work part-time on my thesis project. In my initial planning, I took this into account and aimed at finishing my project last November. Eventually, it turned this was not possible due to extra time I had to spend my other Master. This however bought me some time to go over my developed algorithm again and to implement some additional improvements.

The generation of plant distribution for existing and future are getting important for current applications. Recently, companies such as Bluesky have started mapping accurate tree position for complete countries with remote sensing techniques. The research that I conducted could help these maps in the future to add information about plant types or approximations of positions of smaller plants, which cannot be detected with the current techniques. The three maps are used for various applications in the society, like tree and utility management, which means that this research can contribute to these applications.

Further, the algorithm is able to generate plant distribution for future areas by using the output of ecological models. Ecologists are very interested in such techniques, because they want to have realistic 3D visualizations of the output of their models. The first step in creating a realistic 3D visualization for future natural environments is to have an accurate plant distribution to place the plant models. These 3D visualizations can be used to promote nature to the public and to promote ecological research. Also, it can be used for planning or decision-making cases, because a 3D visualization in combination with the algorithm could improve understanding of how certain actions lead to specific changes in nature. A good example would be the Hedwigepolder that is going to be flooded. Many people think that the Hedwigepolder area is then completely lost to the water. Instead, a complete natural environment is going to grow there which in the future could be used for recreational purposes.

This research and the field of Geomatics are strongly related. Although, the program at TU Delft is focussed more on finding application of geographical data for the urban environment, the field of Geomatics and geographical data is also often applied for rural areas. For example, the related education in Wageningen is more focussed on rural areas. The field of Geomatics is about the analysis, acquisition, management, processing, and visualization of geographical data. Parts of each of these topics are included in this research. An important topic in this research is the acquisition and processing of geographical data, because this research required investigating what kind of spatial datasets could be used for generating a plant distribution and these datasets should be processed to obtain realistic plant positions with their corresponding plant type. As well as investigating how to obtain spatial datasets for future areas. These processes were possible, because of the core courses done from the Geomatics program such as Remote sensing, GIS, Python programming, data quality and 3D modelling. Also, this research combined knowledge from other fields such as Mathematics, Computer Science, and Ecology. That demonstrates the broad and interdisciplinary nature and technical depth of Geomatics.

B
Expert discussion

Discaimer: This document gives an overview/summary of the discussion.

Discussion with ecologist Johan van de Koppel from NIOZ with Benny Onrust. The discussion is about the three point distributions generated by the plant placement algorithm. Two examples are from the Paulinapolder where the first example is based on NDVI and height values, the second and the second includes a discrete LCC map. The third example is based on maps generated by an ecological model.

Paulinapolder contains the following plant types: Salicornia (white), Spartina (yellow), Aster (blue), Limonium (teal), Artemisia (pink), Atriplex (red), Elymus (green).

Ecological model contains the plant types: Salicornia (red), Spartina (green), Aster (yellow)

Each area will be discussed separately. The non-LCC and LCC cases are discussed together.

B.1. Paulinapolder area

What do you think about the general features of the landscape? Are they realistic? General features are the creeks, large-scale patterns of the whole population, and transitions of the vegetation zones within the salt marsh.

Creeks are very good visible in both the LCC and non-LCC case and seems to be realistic.
The large scale patterns are also good, because starting at the water side there are fragmented patterns when getting more land-inwards the patterns of the whole vegetation population become bigger, while the creeks are still visible.
The transitions in the LCC case are very strict which are in some cases not realistic. The large Spartina area in the middle of the map is not realistic. It should contain more species growing in the middle marsh zone. However, the transitions in zones in general are correct, starting with Salicornia and Spartina which goes from Aster to Limonium and Artemisia and Atriplex and Elymus. This is realistic done in the non-LCC case where the changes are gradually as well. For the LCC case this is not realistic, because the transitions are very strict to each zone.

What do you think about the placement of each plant type separately? Are there locations and patterns realistic?

The patterns of Spartina are realistic; clustering very tightly together. However, in the LCC case there is too much Spartina in the middle map, that area should have more other plant types. This is better in the non-LCC case where other plant types are more dominant in that area. The creeks in the non-LCC case are the only problem for Spartina, because on the sides of each creek there grows a lot of Spartina, which should not be there when the creeks get more land-inwards.
Salicornia is more randomly scattered around Spartina in the pioneers zone. This clearly visible and is realistically done in both cases. For both cases there should be probably more Salicornia around Spartina. Further,

for the non-LCC case there is too much Salicornia in the creeks just like for the Spartina. Also, salicornia is placed in the middle of creeks,which is not realistic. In the middle of the creeks, the water speed is too high to have plants settled there. This is better in the LCC case.
Aster should be randomly scattered over the area, which can be seen in both cases. In both cases Aster is too much mixed with the Spartina. That is not possible in this area, because it is too salt. If it was a non-salt marsh then it is possible and realistic. However, based on the salt the Aster should be placed on higher ground. This means that it should start growing where Limonium and Artemisia begin to grow.
Artmesia and Limonium are both placed correctly in the area with the correct patchiness. They both should start growing after patches of Spartina on higher grounds. They should appear less frequent in higher zones of the marsh.This correct for both cases.
Atriplex should be appear growing similar as Limonium and Artemisia in small patches and when going more land-inwards the patches should become larger and Atriplex should appear more frequent. In both cases this is realistically done. However, in the non-LCC case, there is too much Atriplex at higher grounds.
Elymus only grows at the higher grounds of a marsh with large patches. This can be seen in both cases. In the LCC case the patterns are very strict and Elymus should be placed more frequently like in the non-LCC class.

Are the areas realistic?
The non-LCC case is realistic based on the inspected point distribution. In general the patterns for each plant type and the transitions between them are realistic. There is one small error in the coverage of Aster, and sometimes certain creeks contain too many plants. However, these are minor issues, which do not have a large influence on the realism of the area. The LCC case is not realistic, because of the large Spartina area in the middle of the map. Also the transitions between different plant types are very strict.

Any general comments?
Points in each case have approximately the same space to each other. This is of course not always true, for example Salicornia is a very small plant, which can have a very low spacing between each point, while other have for example a larger spacing. However, because in the end the points are going to be used for visualizations, it must be investigated for each plant type how a point is translated to an actual plant model. Some types should be individually placed, others could be placed in patches.
Also judging the realism of an environment is therefore easier to do in a 3D visualization then for a point distribution.
As final note, the algorithm to place plants would be also interesting to be used as a kind of validation for the data used by ecologist, because the ecologist could input his data and then check whether his data matches his expectations.

B.2. ECOLOGICAL MODEL-BASED AREA

This area was judges after the paulinapolder, and most comments also apply for this area. These comment are not repeated.

Spartina
Correct

Aster
In this case I would expect more Aster, because this area does not contain a lot of salt, which results in a 50/50 distribution of Spartina and Aster at higher grounds

Salicornia
Again, Salicornia is placed in the middle of creeks where it cannot grow, because the water speed is too high. It should be placed more next to the Spartina.

Is it realistic?
Yes, it has the same issues as the non-LCC case, but again this are the same minor issues.

General comments
-

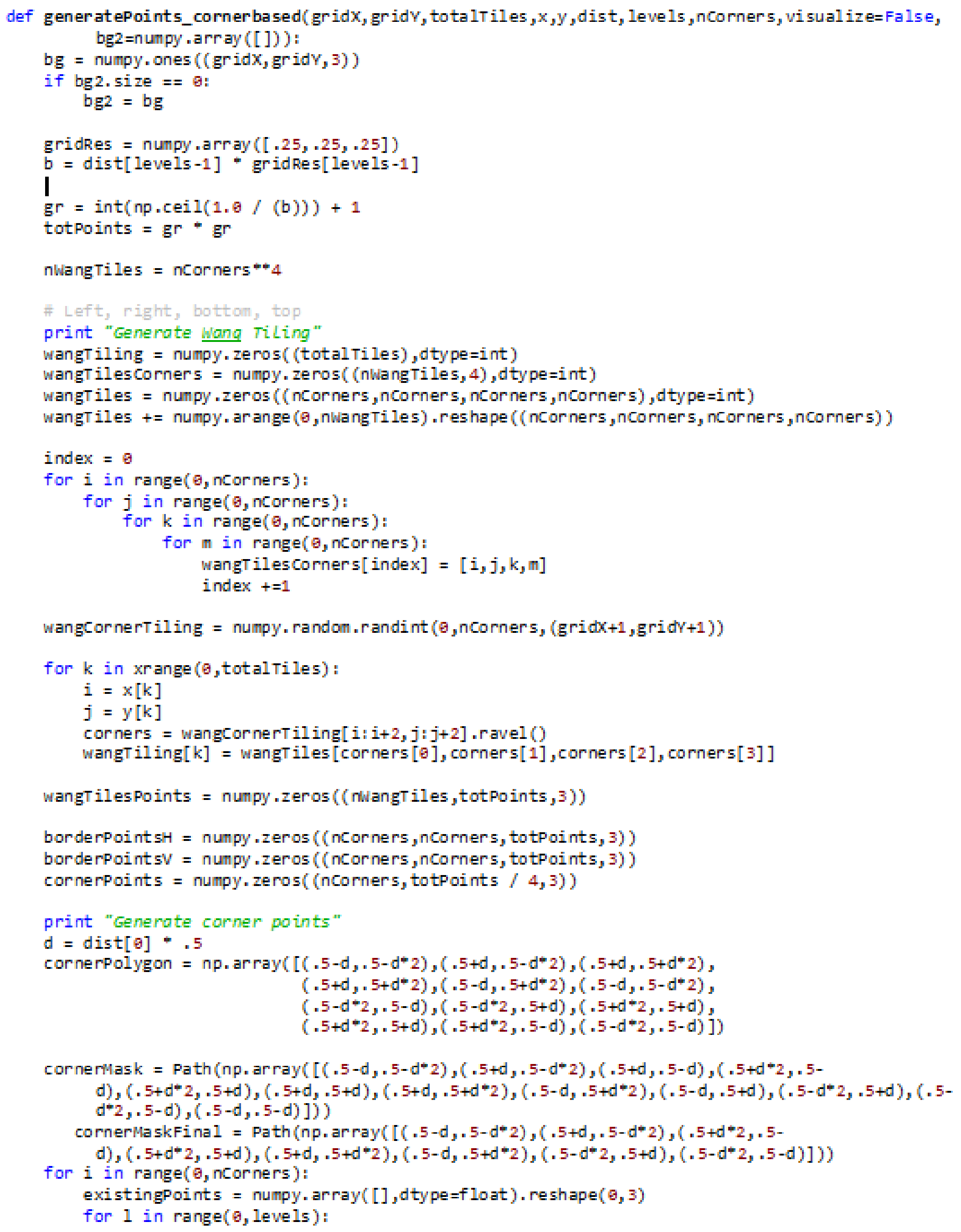
C

Data processing and classification code (parts)

```
def getFractalNoise(points,nTypes,hurst,frequencies):
    totalPoints = points.shape[0]
    frequencies = 1 - frequencies
    cNoise = numpy.zeros((nTypes,totalPoints),dtype=float)
    for i in range(0,nTypes):
        print "type",i
        # Calculate number of octaves
        h = hurst[i]
        f = frequencies[i]
        lac = 1.87
        f **= 2.2
        freq = numpy.min(f)
        offset = 0.0

        octaves = numpy.log(1/freq) / numpy.log(lac) + 1
        octavesInt = numpy.floor(octaves).astype(int)
        octavesInd = numpy.arange(0,octavesInt+1)
        exponent_array = numpy.power(lac, numpy.outer(octavesInd,-h))
        freq_array = freq * numpy.power(lac,octavesInd)
        weights = f[numpy.newaxis,:] / freq_array[:,numpy.newaxis]
        weights[weights > 1] **= -.5
        weights **= 2.0

        w =  numpy.max(weights / numpy.sum(weights,axis=0),axis=1)
        e = numpy.max(exponent_array / numpy.sum(exponent_array,axis=0),axis=1)
        mW = w < .01
        mE = e < .01
        m = numpy.logical_and(mW,mE)

        if octavesInd[m].size != 0:
            octavesInt = octavesInd[m][0]
            print octavesInt

        print "Generate Simplex noise"
        octaveNoise = numpy.zeros((octavesInt,totalPoints),dtype=float)
        for j in range(0,octavesInt):
            sNoise = ((getSimplexNoise_new(points, i, freq_array[j]) + offset) *
exponent_array[j])
            octaveNoise[0:j+1,:] += sNoise

        cumsumExponent = numpy.cumsum(exponent_array[0:octavesInt,:][::-1,:],axis=0)[::-1,:]

        remainder = octaves - octavesInt
        if (remainder > 0):
            octaveNoise += (getSimplexNoise_new(points,i, freq_array[octavesInt])+offset) *
exponent_array[octavesInt]**remainder
            cumsumExponent += exponent_array[octavesInt]**remainder
        octaveNoise /= cumsumExponent

        totWeights = numpy.sum(weights[0:octavesInt],axis=0)
        cNoise[i,:] += numpy.sum(octaveNoise[0:octavesInt] * weights[0:octavesInt],axis=0)
        cNoise[i,:] /= totWeights

    return cNoise

def getSimplexNoise(points, pType, freq):
    simplexNoise = fractalnoise.raw_noise_2d(points*freq, pType)
    simplexNoise /= numpy.sum(simplexNoise)
    simplexNoise *= simplexNoise.shape[0]
    return simplexNoise

def classifyPoints(sortedFractalNoise, fractalNoise, coveragePoints, pointIndices, pointType,
points, plantTypes, visualize):
    nTypes = coveragePoints.shape[0]
    totalPoints = pointIndices.shape[0]
    thresholdIndex = (coveragePoints * (totalPoints-1)).astype(int)

    selectedPositions = numpy.zeros((nTypes,totalPoints),dtype=bool)
    selectedValues = numpy.zeros((nTypes,totalPoints))
    for i in range(0,nTypes):
        thresholdValues = sortedFractalNoise[i,thresholdIndex[i,:]]
        thresholdValues[coveragePoints[i,:] == 0] = numpy.Inf
        selectedPositions[i,:] = fractalNoise[i,:] >= thresholdValues
        selectedValues[i,selectedPositions[i]] = fractalNoise[i,selectedPositions[i]]

    nSelections = numpy.sum(selectedPositions,axis=0)
    selectedPoints = nSelections > 0
    ind = pointIndices[selectedPoints]
    pointType[ind] =
plantTypes[numpy.argmax(selectedValues[:,selectedPoints],axis=0).astype(int)]

    return pointType

def classify_final(fractalNoise,coveragePoints,
heightPoints,ndviPoints,points,heightCoverages,noCov,unit,order,plantTypeLevels,plantInfluences=n
umpy.array([])):
    levels = numpy.unique(points[:,2]).size
    visualize = False
    tree = spatial.cKDTree(points[:,0:2])
    nTypes = fractalNoise.shape[0]
    totalPoints = fractalNoise.shape[1]
    pointType = numpy.zeros((totalPoints),dtype=int)
    plantInfluencesPoints = numpy.zeros((nTypes,totalPoints))
    pointIndices = numpy.arange(0,points.shape[0])
    sortedIndices = numpy.argsort(fractalNoise,axis=1)[:,::-1]
    fractalNoiseSorted = fractalNoise[numpy.arange(nTypes)[:, None],sortedIndices]

    heightBins = numpy.zeros((2,points.shape[0]))
    heightBins[:,:] = (heightPoints / unit)[numpy.newaxis,:]
    heightBins = numpy.trunc(heightBins)
    heightBins[1,:] += 1
    heightBins *= unit
    weights = (unit - numpy.abs(heightPoints[numpy.newaxis,:] - heightBins)).ravel()
    uq,ui = numpy.unique(heightBins,return_inverse=True)
    heightRange = uq.size

    for l in range(0,levels):
        nocoverage = noCov
        m1 = points[:,2] <= l
        pTypes = numpy.where(plantTypeLevels == l)[0]
        nTypes = pTypes.size
        orderLevel = pTypes.copy()
        for o in range(0,orderLevel.shape[0]):
            orderLevel[o] = numpy.where(pTypes[o] == order)[0]
        totCov = numpy.sum(coveragePoints[pTypes],axis=0)
        if nTypes == 0:
            break
        m2 = pointType == 0
        m = numpy.logical_and(m1,m2)
        pInd = pointIndices[m]
        fns = fractalNoiseSorted[m[sortedIndices[pTypes,:]]].reshape(nTypes,pInd.shape[0])
        fn = fractalNoise[pTypes[:,np.newaxis],pInd[np.newaxis,:]]
        cov = coveragePoints[pTypes[:,np.newaxis],pInd[np.newaxis,:]]
        pointType = classifyPoints(fns, fn, cov, pInd, pointType, points,pTypes+1, visualize)

        print "Update coverage values"
        ml = numpy.tile(m,2)
        uil = ui[ml]
        weightsl = weights[ml]
        totCovPerHeight = numpy.bincount(uil,weights=weightsl,minlength=heightRange)
        totCovPerHeight[totCovPerHeight == 0] = 1
        for i in range(0,nTypes):
            i = pTypes[i]
            maskPT = pointType == i+1
            maskPTd = numpy.tile(maskPT,2)
            totCovPerHeightType =
numpy.bincount(ui[maskPTd],weights=weights[maskPTd],minlength=heightRange)
            heightCoveragesNew = heightCoverages[i,:] - totCovPerHeightType / totCovPerHeight
            covFunction = interpolate.interp1d(uq,heightCoveragesNew,
kind='slinear',bounds_error=False,fill_value=0)
            rest = covFunction(heightPoints)
            rest = numpy.minimum(rest,ndviPoints[i])
            covLowerThan0 = rest < 0
            covHigherThan0 = rest > 0
            totMinCov = numpy.sum(rest[covLowerThan0])
            totPlusCov = numpy.sum(rest[covHigherThan0])
            totMinCov /= float(totalPoints)
            totPlusCov /= float(totalPoints)
            rest += plantInfluencesPoints[i,:]
            rest[rest < 0] = 0
            rest[rest > 1] = 1
            coveragePoints[i,:] = rest

        totCov = numpy.sum(coveragePoints[pTypes,:],axis=0)
        totCov[totCov == 0] = 100.0
        totCov += nocoverage
        nocoverage /= totCov
        coveragePoints[pTypes,:] /= totCov

        print "Classify remaining points"
        iTypes = pTypes.copy()
        for i in range(0,nTypes):
            i = orderLevel[i]
            pt = numpy.take(iTypes,[i])
            i = iTypes[i]
            remainingPoints = numpy.logical_and(m,pointType == 0)
            pInd = pointIndices[remainingPoints]
            if pInd.size == 0:
                break

            nPoints = pInd.shape[0]
            fns = fractalNoiseSorted[i,remainingPoints[sortedIndices[i,:]]].reshape(1,nPoints)
            fn = fractalNoise[i,pInd].reshape(1,nPoints)
            cov = coveragePoints[i,pInd].reshape(1,nPoints)
            pointType = classifyPoints(fns, fn, cov, pInd, pointType, points,pt+1,visualize)

            mask = pTypes != i
            pTypes = pTypes[mask]
            if pTypes.size != 0:
                totCov = numpy.sum(coveragePoints[pTypes,:],axis=0) + nocoverage
                totCov[totCov == 0] = 100.0
                coveragePoints[pTypes,:] /= totCov
                nocoverage /= totCov

        if plantInfluences.size != 0 or numpy.sum(plantInfluences) != 0:
            print "Apply plant influences"
            for i in range(0,iTypes.size):
                if numpy.sum(plantInfluences[iTypes[i]]) != 0:
                    pos = pointType == iTypes[i]+1
                    influencedPoints = tree.query_ball_point(points[pos,0:2],.5)
                    for k in range(0,numpy.sum(pos)):
                        for j in range(0,plantInfluences.shape[1]):
                            plantInfluencesPoints[j,influencedPoints[k]] +=
plantInfluences[iTypes[i],j]
            coveragePoints += plantInfluencesPoints
            coveragePoints[coveragePoints < 0] = 0

    return pointType

def main():
    print "Load data"

    sourceHeight = 'height_masked_final.asc'
    sourceBiomass = 'ndvi_masked_final.asc'

    sourceCoverageModel = 'testCov.txt'
    sourceHeightModel = 'heightModel.txt'
    sourceLCC = "LCC.asc"

    heightGrid = numpy.loadtxt(sourceHeight, skiprows=6)
    heightGrid = heightGrid[500:1050,300:750]
    ndviGrid = numpy.loadtxt(sourceBiomass, skiprows=6)
    ndviGrid = ndviGrid[500:1050,300:750]
    lccGrid = numpy.loadtxt(sourceLCC, skiprows=6)
    lccGrid = lccGrid[500:1050,300:750]
    heightModelGrid = numpy.loadtxt(sourceHeightModel)
    coverageModelGrid = numpy.loadtxt(sourceCoverageModel)

    PaulinaPolder = False
    NDVI = True
    LCC = False
    if PaulinaPolder:
        if NDVI and LCC:
            vegetationMask = ndviGrid > 0
        elif NDVI:
            vegetationMask = ndviGrid > 0.02
        elif LCC:
            vegetationMask = lccGrid > 0
        heightValues = heightGrid[vegetationMask]
        baseValues = ndviGrid[vegetationMask]
        lccValues = lccGrid[vegetationMask]
        lengthX, lengthY = heightGrid.shape
    nTypes = 7
    area = "NDVI"

    lXTemp = lengthX
    lYTemp = lengthY
    if lengthX % 2 == 1:
        lXTemp += 1
    if lengthY % 2 == 1:
        lYTemp += 1
    vegetationMaskExtended = np.zeros((lXTemp,lYTemp),dtype=bool)
    vegetationMaskExtended[0:lengthX,0:lengthY] = vegetationMask
    res = 2.0
    wangGridLengthX = np.ceil(lengthX / res)
    wangGridLengthY = np.ceil(lengthY / res)
    xWangIndices,yWangIndices = numpy.indices((wangGridLengthX,wangGridLengthY))
    blocks = view_as_blocks(vegetationMaskExtended, block_shape=(int(res),int(res)))
    blocks = blocks.reshape(wangGridLengthX,wangGridLengthY,res*res)
    blocks_summed = np.sum(blocks,axis=2)
    wangVegetationMask = blocks_summed > 0
else:
    vegetationMask = coverageModelGrid > 0
    heightValues = heightModelGrid[vegetationMask] * 1
    baseValues = coverageModelGrid[vegetationMask] * .01
    lengthX, lengthY = heightModelGrid.shape
    nTypes = 3
    heightValues += numpy.fabs(numpy.min(heightValues))
    minHeight = numpy.min(heightValues)
    maxHeight = numpy.max(heightValues)
    heightValues = (heightValues - minHeight) / (maxHeight - minHeight)
    heightValues *= 100
    area = "MODEL"

    tileIDs = numpy.arange(0,heightValues.shape[0])
    tileIDsGrid = numpy.zeros((lengthX,lengthY))
    tileIDsGrid[vegetationMask] = tileIDs

    wangRes = 1
    res = 1
    wangLengthX = lengthX * res
    wangLengthY = lengthY * res

    wangVegetationMask = numpy.zeros((wangLengthX,wangLengthY),dtype=bool)
    xWangIndices,yWangIndices = numpy.indices((wangLengthX,wangLengthY))
    xw = numpy.trunc(xWangIndices / res).astype(int)
    yw = numpy.trunc(yWangIndices / res).astype(int)
    wangVegetationMask[xWangIndices,yWangIndices] = vegetationMask[xw,yw]

nTiles = heightValues.shape[0]
plantTypeLevels = numpy.zeros((nTypes))

if PaulinaPolder:
    if LCC:
        area2 = "LCC"
    elif NDVI:
        area2 = "NDVI"
    heightMatrix = getHeightMatrix("NDVI")
    coverageMatrix = getCoverageMatrix(area2) * .01
    hurstMatrix = getHurstMatrix("NDVI")
else:
        heightMatrix = getHeightMatrix("MODEL")
        coverageMatrix = getCoverageMatrix("MODEL") * .01
        hurstMatrix = getHurstMatrix("MODEL")

    unit = 1.0
    heightBins = numpy.zeros((2,heightValues.shape[0]))
    heightBins[:,:] = (heightValues / unit)[numpy.newaxis,:]
    heightBins = numpy.trunc(heightBins)
    heightBins[1,:] += 1
    heightBins *= unit
    heightBins = numpy.unique(heightBins)
    heightCoverage = numpy.zeros((nTypes,heightBins.size))
    heightHurst = numpy.zeros((nTypes,heightBins.size))

    for i in range(0,nTypes):
        fc = interpolate.interp1d(heightMatrix,coverageMatrix[i],
kind='slinear',bounds_error=False,fill_value=0)
        fh = interpolate.interp1d(heightMatrix,hurstMatrix[i],
kind='slinear',bounds_error=False,fill_value=0)
        heightCoverage[i] = fc(heightBins)
        heightHurst[i] = fh(heightBins)

    print "Declare data"
    coveragePerTile = numpy.zeros((nTypes,nTiles),dtype=float)
    hurstPerTile = numpy.zeros((nTypes,nTiles),dtype=float)
    constraintsPerTile = numpy.zeros((nTypes,nTiles),dtype=float)
    compositionPerTile = numpy.zeros((nTypes,nTiles),dtype=float)
    ndviPerTile = numpy.ones((nTypes,nTiles),dtype=float)

    print "Get data"
    for i in range(0,nTypes):
        fc = interpolate.interp1d(heightBins,heightCoverage[i],
kind='slinear',bounds_error=False,fill_value=0)
        fh = interpolate.interp1d(heightBins,heightHurst[i],
kind='slinear',bounds_error=False,fill_value=0)
        coveragePerTile[i] = fc(heightValues)
        hurstPerTile[i] = fh(heightValues)
        if NDVI and LCC:
            ndviPerTile[i] = calculateCovNDVI(area,i,baseValues)
            lccTiles = calculateCovLCC(i,lccValues)
            constraintsPerTile[i] = numpy.minimum(ndviPerTile[i],lccTiles)
            compositionPerTile[i] = numpy.minimum(coveragePerTile[i],constraintsPerTile[i])
        elif NDVI:
            ndviPerTile[i] = calculateCovNDVI(area,i,baseValues)
            constraintsPerTile[i] = ndviPerTile[i]
            compositionPerTile[i] = numpy.minimum(coveragePerTile[i],constraintsPerTile[i])
        elif LCC:
            constraintsPerTile[i] = calculateCovLCC(i,lccValues)
            compositionPerTile[i] = numpy.minimum(coveragePerTile[i],constraintsPerTile[i])

    if LCC:
        if NDVI:
            tempCov = numpy.minimum(ndviPerTile,coveragePerTile)
        else:
            tempCov = numpy.minimum(ndviPerTile,coveragePerTile)
        lccGroups = 4
        noCoveragePerTile = 1 - numpy.sum(tempCov,axis=0)
        noCoveragePerTile[noCoveragePerTile < 0] = 0
        for i in range(0,lccGroups):
            lccMask = numpy.where(lccValues == i+1)[0]
            plantsLCC = getLCCTypes(i+1)
            tcon = constraintsPerTile[:,lccMask]
            tcon = tcon[plantsLCC,:]
            totCovlcc = numpy.sum(tcon,axis=0)
            for j in range(0,plantsLCC.shape[0]):
                compositionPerTile[plantsLCC[j],lccMask] /=
(totCovlcc+noCoveragePerTile[lccMask])
    elif NDVI:
        totCov = numpy.sum(compositionPerTile,axis=0)
        noCoveragePerTile = 1 - totCov
        noCoveragePerTile[noCoveragePerTile < 0] = 0
        compositionPerTile /= totCov+noCoveragePerTile

    xWang = xWangIndices[wangVegetationMask]
    yWang = yWangIndices[wangVegetationMask]
    nWangTiles = xWang.shape[0]
    print "Start Point Generation"
    if PaulinaPolder:
        tileIDs = numpy.arange(0,nTiles)
        tileIDsGrid = numpy.zeros((lengthX,lengthY))
        tileIDsGrid[vegetationMask] = tileIDs
        dist = numpy.array([.4]) / res
        points = wangtiles.generatePoints_cornerbased(wangGridLengthX,wangGridLengthY,
nWangTiles, xWang, yWang,dist,1,4)
        points[:,0:2] *= res
        trunckedPoints = np.trunc(points[:,0:2]).astype(int)
        cull = vegetationMaskExtended[trunckedPoints[:,0],trunckedPoints[:,1]]
        points = points[cull,:]
        trunckedPoints = trunckedPoints[cull,:]
        points[:,3] = tileIDsGrid[trunckedPoints[:,0],trunckedPoints[:,1]]
    else:
        dist = numpy.array([.33]) / wangRes
        points = wangtiles.generatePoints_cornerbased(wangLengthX,wangLengthY, nWangTiles, xWang,
yWang,dist,1,4)
        points[:,0:2] *= wangRes
        trunckedPoints = np.trunc(points[:,0:2] / (res*wangRes)).astype(int)
        points[:,3] = tileIDsGrid[trunckedPoints[:,0],trunckedPoints[:,1]]

    totalPoints = points.shape[0]
    print totalPoints

    hurstPoints = hurstPerTile[:,points[:,3].astype(int)]
    coveragePoints = compositionPerTile[:,points[:,3].astype(int)]
    heightPoints = heightValues[points[:,3].astype(int)]
    nocoveragePoints = noCoveragePerTile[points[:,3].astype(int)]
    basePoints = constraintsPerTile[:,points[:,3].astype(int)]


    print "Generate MultiFractal Noise"
    scaleFactor = math.sqrt(totalPoints / (nTiles * 1.0)) * 0.5
    cNoisePoints = getFractalNoise_new(points[:,0:2]*scaleFactor,nTypes,hurstPoints,hurstPoints)

    avgFreqT = numpy.average(hurstPoints, axis=1)
    meanFreq = numpy.average(avgFreqT)
    sd = numpy.fabs(avgFreqT - meanFreq)
    order = numpy.argsort(sd)[::-1]

    print "Start classification process"




    pointType = classify_final(cNoisePoints,coveragePoints.copy(), heightPoints,
basePoints,points,heightCoverage,nocoveragePoints.copy(),unit,order,plantTypeLevels)
```

D

Point generation code (parts)



```
def generatePoints_cornerbased(gridX,gridY,totalTiles,x,y,dist,levels,nCorners,visualize=False,
        bg2=numpy.array([])):
    bg = numpy.ones((gridX,gridY,3))
    if bg2.size == 0:
        bg2 = bg

    gridRes = numpy.array([.25,.25,.25])
    b = dist[levels-1] * gridRes[levels-1]

    gr = int(np.ceil(1.0 / (b))) + 1
    totPoints = gr * gr

    nWangTiles = nCorners**4

    # Left, right, bottom, top
    print "Generate Wang Tiling"
    wangTiling = numpy.zeros((totalTiles),dtype=int)
    wangTilesCorners = numpy.zeros((nWangTiles,4),dtype=int)
    wangTiles = numpy.zeros((nCorners,nCorners,nCorners,nCorners),dtype=int)
    wangTiles += numpy.arange(0,nWangTiles).reshape((nCorners,nCorners,nCorners,nCorners))

    index = 0
    for i in range(0,nCorners):
        for j in range(0,nCorners):
            for k in range(0,nCorners):
                for m in range(0,nCorners):
                    wangTilesCorners[index] = [i,j,k,m]
                    index +=1

    wangCornerTiling = numpy.random.randint(0,nCorners,(gridX+1,gridY+1))

    for k in xrange(0,totalTiles):
        i = x[k]
        j = y[k]
        corners = wangCornerTiling[i:i+2,j:j+2].ravel()
        wangTiling[k] = wangTiles[corners[0],corners[1],corners[2],corners[3]]

    wangTilesPoints = numpy.zeros((nWangTiles,totPoints,3))

    borderPointsH = numpy.zeros((nCorners,nCorners,totPoints,3))
    borderPointsV = numpy.zeros((nCorners,nCorners,totPoints,3))
    cornerPoints = numpy.zeros((nCorners,totPoints / 4,3))

    print "Generate corner points"
    d = dist[0] * .5
    cornerPolygon = np.array([(.5-d,.5-d*2),(.5+d,.5-d*2),(.5+d,.5+d*2),
                              (.5+d,.5+d*2),(.5-d,.5+d*2),(.5-d,.5-d*2),
                              (.5-d*2,.5-d),(.5-d*2,.5+d),(.5+d*2,.5+d),
                              (.5+d*2,.5+d),(.5+d*2,.5-d),(.5-d*2,.5-d)])

    cornerMask = Path(np.array([(.5-d,.5-d*2),(.5+d,.5-d*2),(.5+d,.5-d),(.5+d*2,.5-
        d),(.5+d*2,.5+d),(.5+d,.5+d),(.5+d,.5+d*2),(.5-d,.5+d*2),(.5-d,.5+d),(.5-d*2,.5+d),(.5-
        d*2,.5-d),(.5-d,.5-d)]))
       cornerMaskFinal = Path(np.array([(.5-d,.5-d*2),(.5+d,.5-d*2),(.5+d*2,.5-
        d),(.5+d*2,.5+d),(.5+d,.5+d*2),(.5-d,.5+d*2),(.5-d*2,.5+d),(.5-d*2,.5-d)]))
    for i in range(0,nCorners):
        existingPoints = numpy.array([],dtype=float).reshape(0,3)
        for l in range(0,levels):
            points = generatePDD (gridRes[l],dist,l,cornerMask,cornerMaskFinal,minBoundX=.5-
        d*2,minBoundY=.5-
        d*2,maxBoundX=.5+d*2,maxBoundY=.5+d*2,existingPoints=existingPoints,itr=50)
            existingPoints = numpy.concatenate((existingPoints,points),axis=0)
        cornerPoints[i,0:points.shape[0]] = points

print "Generate border points"
d = dist[0]
borderVMask = Path(np.array([(.5 - d*.5,d),(.5 + d*.5,d),(.5 + d*.5,1-d),(.5 - d*.5,1-d)]))
borderVPolygon = np.array([(.5 - d*.5,d),(.5 + d*.5,d),(.5 + d*.5,1-d),
                           (.5 + d*.5,1-d),(.5 - d*.5,1-d),(.5 - d*.5,d)])
for i in range(0,nCorners):
    ct = nozero(cornerPoints[i,:])
    ct[:,1] += .5
    for j in range(0,nCorners):
        cb = nozero(cornerPoints[j,:])
        cb[:,1] -= .5
        borderPoints = numpy.concatenate((ct,cb),axis=0)
        existingPoints = borderPoints
        for l in range(0,levels):
            points = generatePDD (gridRes[l],dist,l,borderVMask,borderVMask,minBoundX=(.5 -
        d*.5),minBoundY=d,maxBoundX=(.5 + d*.5),maxBoundY=(1 -
        d),existingPoints=existingPoints,itr=50)
            existingPoints = numpy.concatenate((borderPoints,points),axis=0)
        borderPointsV[i,j,0:points.shape[0]] = points

d = dist[0]
borderHMask = Path(np.array([(d,.5 + d*.5),(d,.5 - d*.5),(1-d,.5 - d*.5),(1-d,.5 + d*.5)]))
borderHPolygon = np.array([(d,.5 + d*.5),(d,.5 - d*.5),(1-d,.5 - d*.5),
                           (1-d,.5 - d*.5),(1-d,.5 + d*.5),(d,.5 + d*.5)])

for i in range(0,nCorners):
    cr = nozero(cornerPoints[i,:])
    cr[:,0] += .5
    for j in range(0,nCorners):
        cl = nozero(cornerPoints[j,:])
        cl[:,0] -= .5

        borderPoints = numpy.concatenate((cr,cl),axis=0)
        existingPoints = borderPoints
        for l in range(0,levels):
            points = generatePDD
        (gridRes[l],dist,l,borderHMask,borderHMask,minBoundX=d,minBoundY=(.5 - d*.5),maxBoundX=(1 -
        d),maxBoundY=(.5 + d*.5),existingPoints=existingPoints,itr=50)
            existingPoints = numpy.concatenate((borderPoints,points),axis=0)
        borderPointsH[i,j,0:points.shape[0]] = points

d = dist[0] * .5

tileMask = Path(np.array([(d*2,d),(1-d*2,d),(1-d,d*2),(1-d,1-d*2),(1-d*2,1-d),(d*2,1-d),(d,1-
        d*2),(d,d*2)]))
tileMask = Path(np.array([(d,d),(1-d,d),(1-d,1-d),(d,1-d)]))
tileMaskFinal = Path(np.array([(0,0),(1,0),(1,1),(0,1)]))
print "Generate wang tile points"

for i in range(0,nWangTiles):
    wt = wangTilesCorners[i,:]
    leftTop = wt[1]
    rightTop = wt[3]
```

```
        leftBot = wt[0]
        rightBot = wt[2]
        left = nozero(borderPointsV[leftTop,leftBot])
        left[:,0] -= .5
        right = nozero(borderPointsV[rightTop,rightBot])
        right[:,0] += .5
        bottom = nozero(borderPointsH[rightBot,leftBot])
        bottom[:,1] -= .5
        top = nozero(borderPointsH[rightTop,leftTop])
        top[:,1] += .5

        ltPoints = nozero(cornerPoints[leftTop])
        ltPoints[:,0] -= .5
        ltPoints[:,1] += .5
        rtPoints = nozero(cornerPoints[rightTop])
        rtPoints[:,0] += .5
        rtPoints[:,1] += .5
        lbPoints = nozero(cornerPoints[leftBot])
        lbPoints[:,0] -= .5
        lbPoints[:,1] -= .5
        rbPoints = nozero(cornerPoints[rightBot])
        rbPoints[:,0] += .5
        rbPoints[:,1] -= .5

        borderCornerPoints =
          numpy.concatenate((left,right,bottom,top,ltPoints,rtPoints,lbPoints,rbPoints),axis=0)
        existingPoints = borderCornerPoints

        for l in range(0,levels):
            points =
          generatePDD(gridRes[l],dist,l,tileMask,tileMaskFinal,minBoundX=d,minBoundY=d,maxBoundX=1-
          d,maxBoundY=1-d,existingPoints=existingPoints,itr=50)
            existingPoints = numpy.concatenate((borderCornerPoints,points),axis=0)
        wangTilesPoints[i,0:points.shape[0],:] = points

    cornerPolygon -= .5
    vertCorner = cornerPolygon.shape[0]
    vertBorder = borderHPolygon.shape[0]
    totCorners = (gridX+1)*(gridY+1)
    totBorderH = gridX*(gridY+1)
    totBorderV = (gridX+1)*gridY
    vertices = numpy.zeros((vertCorner*totCorners+totBorderH*vertBorder+totBorderV*vertBorder,2))
    colors = numpy.zeros((vertCorner*totCorners+totBorderH*vertBorder+totBorderV*vertBorder,3))
    cornerTypes = wangCornerTiling.ravel()
    xci,yci = numpy.indices((gridX+1,gridY+1))
    xci = xci.ravel()
    yci = yci.ravel()
    xbhi,ybhi = numpy.indices((gridX,gridY+1))
    xbhi = xbhi.ravel()
    ybhi = ybhi.ravel()
    xbvi,ybvi = numpy.indices((gridX+1,gridY))
    xbvi = xbvi.ravel()
    ybvi = ybvi.ravel()
    for i in range(0,int(totCorners)):
        temp = cornerPolygon.copy()
        xc = xci[i]
        yc = yci[i]
        temp[:,0] += xc
        temp[:,1] += yc
        vertices[i*vertCorner:i*vertCorner+vertCorner,:] = temp
        colors[i*vertCorner:i*vertCorner+vertCorner] = getColor(int(cornerTypes[i]+1))

    borderHPolygon[:,1] -= .5
    borderVPolygon[:,0] -= .5
    start = totCorners*vertCorner
    gray = numpy.ones((vertBorder,3))
    gray *= .75
    for i in range(0,int(totBorderH)):
        tempH = borderHPolygon.copy()
        xbh = xbhi[i]
        ybh = ybhi[i]
        tempH[:,0] += xbh
        tempH[:,1] += ybh
        vertices[start+i*vertBorder:start+i*vertBorder+vertBorder,:] = tempH
        colors[start+i*vertBorder:start+i*vertBorder+vertBorder] = gray

    start = totCorners*vertCorner+totBorderH*vertBorder
    for i in range(0,int(totBorderV)):
        tempV = borderVPolygon.copy()
        xbv = xbvi[i]
        ybv = ybvi[i]
        tempV[:,0] += xbv
        tempV[:,1] += ybv
        vertices[start+i*vertBorder:start+i*vertBorder+vertBorder,:] = tempV
        colors[start+i*vertBorder:start+i*vertBorder+vertBorder] = gray

    print "Fill Wang Tiling"
    tileIDs = numpy.arange(0,totalTiles)
    allPoints = wangTilesPoints[wangTiling]
    mask = numpy.all(allPoints[:,:,0:2] > 0,axis=2)
    allPoints[:,:,0] += x[:,numpy.newaxis]
    allPoints[:,:,1] += y[:,numpy.newaxis]
    test = numpy.zeros((allPoints.shape[0],allPoints.shape[1]))
    test += tileIDs[:,numpy.newaxis]
    test = test[mask]
    allPoints = allPoints[mask]
    pointDistribution = numpy.zeros((allPoints.shape[0],4))
    pointDistribution[:,0:3] = allPoints
    pointDistribution[:,3] = test

    return pointDistribution

def generatePDD(gridResolution, min_dist_array,level,mask,finalMask,minBoundX = 0.0, minBoundY =
        0.0, maxBoundX = 1.0, maxBoundY = 1.0, existingPoints=np.array([]),itr=20):

    min_dist = min_dist_array[level]
    totLevels = min_dist_array.shape[0]
    b = min_dist * .5
    nGroups = int(np.ceil(1.0 / (b)))
    nPoints = (nGroups+1)*(nGroups+1) + existingPoints.shape[0]
    points = np.zeros((nPoints,3))
    currentNPoints = 0

    if existingPoints.shape[0] == 0:
        rPoint =  np.random.uniform(0,1,(1,2))
        points[currentNPoints,0] = rPoint[0,0] * (maxBoundX - minBoundX) + minBoundX
        points[currentNPoints,1] = rPoint[0,1] * (maxBoundY - minBoundY) + minBoundY
        points[currentNPoints,2] = level
    currentNPoints += 1
else:
    points[0:existingPoints.shape[0],:] = existingPoints
    currentNPoints += existingPoints.shape[0]

totalItr = nGroups*nGroups
x,y = np.indices((nGroups,nGroups)) * (1/nGroups)
x = x.ravel()
y = y.ravel()
randomPointsGrid = np.random.uniform(0,1,(totalItr,totalItr,2))
randomPointsGrid[:,:,0] *= (maxBoundX - minBoundX)
randomPointsGrid[:,:,1] *= (maxBoundY - minBoundY)
randomPointsGrid[:,:,0] += minBoundX
randomPointsGrid[:,:,1] += minBoundY
t1 = mask.contains_points(randomPointsGrid.reshape(totalItr*totalItr,2))
withInBounds = t1.reshape(totalItr,totalItr).astype(int)
for i in xrange(0,totalItr):
    dist = spatial.distance.cdist(randomPointsGrid[i,:,:], points[0:currentNPoints,0:2])
    distances = 0
    for l in range(level,totLevels):
        mask = points[0:currentNPoints,2] <= l
        min_dist = min_dist_array[l]
        distL = dist[:,mask]
        d = np.min(distL,axis=1)
        distances += d
        score = withInBounds[i,:]
        score += (d > min_dist).astype(int)
    inds = np.where(score == (1 + totLevels - level))[0]
    if  score[inds].size > 0:
        minDistID = np.argmin(distances[inds])
        k = inds[minDistID]
        points[currentNPoints,0:2] = randomPointsGrid[i,k]
        points[currentNPoints,2] = level
        currentNPoints += 1
m = finalMask.contains_points(points[:currentNPoints,0:2])
return points[m,:]
```