# 2D Voronoi Diagram Applications: Branching Structures and Crack Patterns Synthesis

Bachelor thesis - Mathematics and Computer Science

Ecole Polytechnique


Maria Bărbulescu

Supervisor: Pooran Memari

April 2020

# Contents


**1 Introduction** **2**

**2 Overview** **3**

**3 Preliminaries** **4**
- 3.1 Delaunay Triangulation . . . . . 4
- 3.2 Voronoi Diagram . . . . . 5
- 3.3 Medial Axis . . . . . 5

**4 Curve Reconstruction and Medial Axis Approximation** **6**
- 4.1 Curve Reconstruction: 2D Crust Algorithm . . . . . 6
- 4.2 Medial Axis Approximation . . . . . 7

**5 Medial Axis Approximation Refinement** **9**
- 5.1 Point Filtering . . . . . 9
- 5.2 Edge Filtering . . . . . 11
- 5.3 Results Comparison . . . . . 13

**6 Medial Axis Applications** **15**
- 6.1 Maple Leaf . . . . . 15
- 6.2 Other Natural Occurring Patterns . . . . . 16

**7 Maze Generation** **17**
- 7.1 Voronoi Diagram Maze . . . . . 17
- 7.2 Delaunay Triangulation Maze . . . . . 18

**8 Crack Patterns** **19**
- 8.1 Uniform Distribution . . . . . 19
- 8.2 Nested Curves . . . . . 20
- 8.3 Gaussian and Laplace Distributions . . . . . 20

**9 Conclusions and Future Work** **22**

**10 References** **23**

# Section 1

# Introduction

Creating 2D branching structures is a common requirement in computer graphics, as they can model different types of structures in many application fields. These branched structures could, for example, represent crack patterns, river or sewage networks, neurons or blood vessels in the human body. For the purpose of this thesis, the crack patterns are treated separately from branching structures due to the different methods used to synthesize them.

When creating crack patterns, scientists use physics-dependent methods that rely on external stress and forces that can be applied in both two and three dimension, such as the ones proposed in [6]. The new method presented in this thesis is less complex, but also less specific, making it easier to adapt to different two-dimensional applications. A new algorithm is also presented for the purpose of synthesizing branching structures. Similarly to [5], the algorithm relies on implementations of Voronoi diagrams to approximate the medial axis of a curve, extracting both the boundary and the skeleton. However, it is based on a modification of the well-known crust algorithm [2] rather than the one-step method the paper proposes, and its principal aim is the generation of different patterns. With certain point clouds, the algorithm presented seems to yield more accurate results in two dimensions compared to previous attempts.

The medial axis of a shape is often used in the field of visual computing for different purposes, such as animation and transformation of 2D and 3D shapes. This thesis presents a less common use of the medial axis, namely the generation of patterns itself. Because of its sensitivity to the small changes in the curve (such as noise or zigzag patterns), the generated medial axis can present branches which ultimately lead to the synthesis of tree-like shapes and patterns.

Because the medial axis computation is highly dependent on a data set's density and distribution, the outputs can significantly vary for similar inputs and can hardly be predicted or guided. However, this leads to the synthesis of spontaneous patterns that can prove to be more interesting in the field of visual computing. Iterations of the medial axis algorithm are also used in order to form nested curves, creating more intricate patterns that can be used for complex maze generation.

# Section 2

# Overview

The algorithm used in order to synthesize the tree-like structures is based on graph duality, relying on Voronoi diagrams and their respective Delaunay triangulations. Given a sample set of points, the algorithm computes the curve they describe (the crust) and approximates its medial axis. Because of the nature of the crust algorithm and the fact that the medial axis computation is reasonably unstable and particularly sensitive to boundary perturbations [4], the initial approximation is not accurate enough to proceed with the study of different applications. Thus, two different refinement methods are implemented in order to improve the accuracy of said approximation and reduce noise and error propagation.

The curve reconstruction algorithm used is based on the Delaunay-oriented approach described in [2]. The medial axis is then approximated using the convergence of the Voronoi vertices [3]. The first attempt at refining the medial axis approximation is done by filtering Voronoi vertices using a well-known algorithm that has proven to work in 3D [1], namely using the vertices known as poles which converge to the medial axis. The second one is done through a new algorithm relying on edge filtering, only keeping the medial axis edges that belong to the initial Voronoi diagram. The latter proves to be efficient enough for the scope of this thesis, providing a final algorithm that works sufficiently well on both smooth curves and curves that contain sharp edges and that can synthesize different branching structures.

Starting with the Voronoi diagram of random sets of points, an algorithm is implemented to generate mazes using Depth-first search. By applying the algorithm on point sets with different distributions, these mazes simulate crack patterns in a simple yet efficient way. This new method of synthesizing crack patterns proves suitable for different 2D applications. The creation of nested curves allows for more intricate mazes and patterns, and it is done through iteratively applying the refined medial axis algorithm over an expanding data set.

In order to account for a wider range of sample points, a MATLAB app is developed to capture and save the coordinates of mouse-clicks on a two-dimensional Cartesian coordinate system. Additionally, different image analysis methods are used to extract points from images.

# Section 3

# Preliminaries

This section gives a description of the notions and definitions that future algorithms are based on, i.e. Delaunay triangulation and its respective Voronoi diagram, as well as the medial axis.

## 3.1 Delaunay Triangulation

The Delaunay triangulation is a canonical triangulation such that for every triangle, its circumscribed circle does not contain any additional points from the data set (Figure 3.1). Adapting the formal definition given in [7] to a two-dimensional space, if $P$ is a set of points in $\mathbb{R}^2$ and $p_i \in P$ with $p_i = (x, y)$, let $h$ be a map from $\mathbb{R}^2$ to $\mathbb{R}^3$ such that $h(p_i) = (p_i, x^2 + y^2)$. Then, the orthogonal projection of the lower envelope of $h$ produces the Delaunay triangulation.

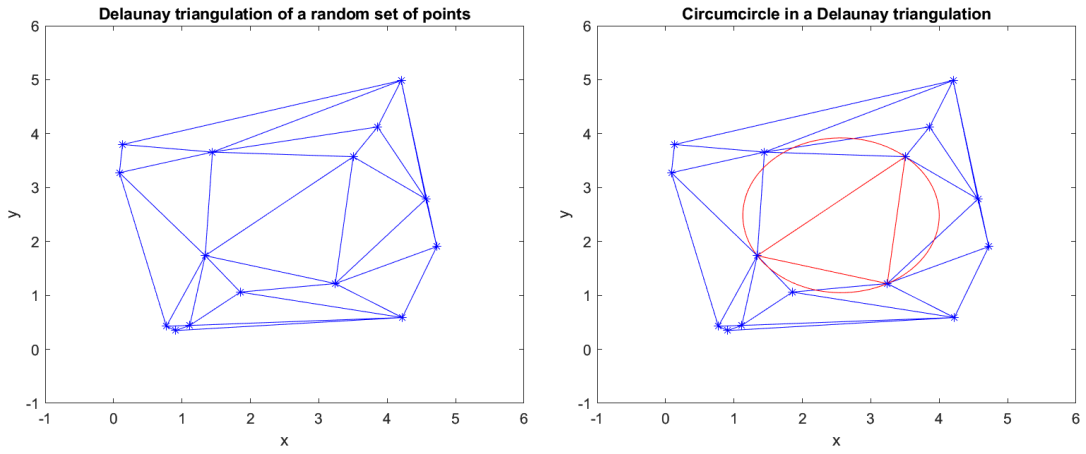


Figure 3.1: Delaunay triangulation

However, the most common way of defining the Delaunay triangulation is as the geometric dual of the Voronoi diagram. Specifically, each circumcenter of a Delaunay triangle is a Voronoi vertex. This duality comes in useful in many of the future

algorithms presented in this paper. It is also worth noting that, unlike other triangulations, the Delaunay method maximizes the smallest angle of a triangle. The Delaunay triangulation also contains the minimum spanning tree and the nearest neighbor graph.

## 3.2 Voronoi Diagram

The Voronoi diagram is the partition of a plane into convex polygons, or cells, such that each cell contains only one point of the initial data set, called the generating point. Additionally, every point in each polygon is closer to its generating point than any other point in the data set. In two dimensions, this can be described mathematically as follows.

Let $P = \{p_1, p_2, ..., p_n\}$ be a finite set of points in $\mathbb{R}^2$. Then, a Voronoi cell of a point $p_i$ is $V(p_i) = \{x \mid \|x - p_i\| \leq \|x - p_j\|, \forall j \in \{1, ..., n\}, j \neq i, \forall x \in \mathbb{R}^2\}$ and the Voronoi diagram is $Vor(P) = \{V(p_i) \forall i \in \{1, ..., n\}\}$. The Delaunay triangulation can then be defined as $Del(P) =$ the nerve of $Vor(P)$ or as the set of simplices with an empty circumscribing circle.

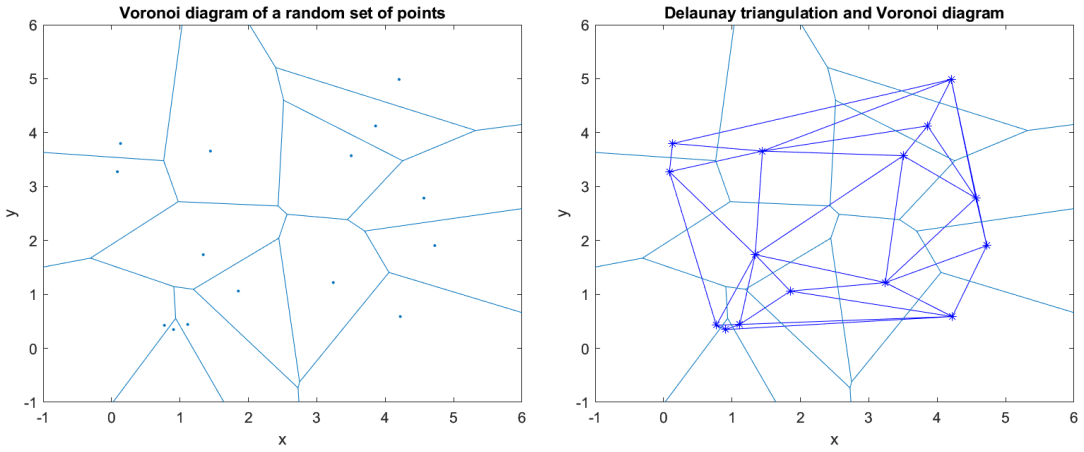


Figure 3.2: Voronoi diagram and geometric duality

## 3.3 Medial Axis

The medial axis, also known as the topological skeleton, of an open set $O$ is the set of points $x \in O$ such that $x$ has more than one closest point on the boundary of $O$. While this definition concerns continuous shapes, the Voronoi approximation is used for discrete settings. From the mathematical definition of the Voronoi diagram, it is easily deduced that each Voronoi edge is situated exactly halfway between two initial points. Furthermore, it has been proven that, for a smooth surface, the Voronoi vertices converge towards the medial axis ([4], [3]). A closed curve has both an internal and an external medial axis, which can be observed in Figure 5.4.

# Section 4

# Curve Reconstruction and Medial Axis Approximation

The purpose of this section is to summarise the algorithm behind curve reconstruction given a set of points as well as the approximation of the curve's medial axis.

## 4.1 Curve Reconstruction: 2D Crust Algorithm

The preliminary notions are now used for reconstructing a curve given a point cloud. The algorithm used for the reconstruction starts with the Delaunay triangulation of the point cloud, followed by its respective Voronoi diagram. The resulting Voronoi vertices are then added to the set of sample points, resulting in a secondary point cloud which is then triangulated according to the Delaunay transformation. In the final step, only the edges of the triangulation that are between initial sample points are kept, and the result is the reconstructed curve.

The mathematical model describing this algorithm is as follows. Let $P$ be the initial set of points and $V$ the set of its respective Voronoi vertices. Then, $D = P \cup V$ is the set of points that is to be Delaunay-triangulated. An edge of this triangulation belongs to the crust if and only if both of its end-points are in $P$. As proven in [2], the crust is a subset of the Delaunay triangulation of $P$, fact that can be observed in Figure 4.1.

While this algorithm yields an accurate reconstruction of the curve described by the point cloud in Figure 4.1, there are shortcomings when it is used on certain point sets. As described in [5], these shortcomings are to be expected when working on an insufficient sample: when two points are too far apart, they might not be connected through an edge in the second triangulation, which will ultimately result in a disruption of the curve and a poor approximation of the crust. Conversely, an overly dense point cloud can also yield an unsatisfactory result, since triangle edges can form between points that are not connected in the curve during the second

triangulation, thus affecting the overall output. To avoid error propagation, only dense enough point clouds will be considered.

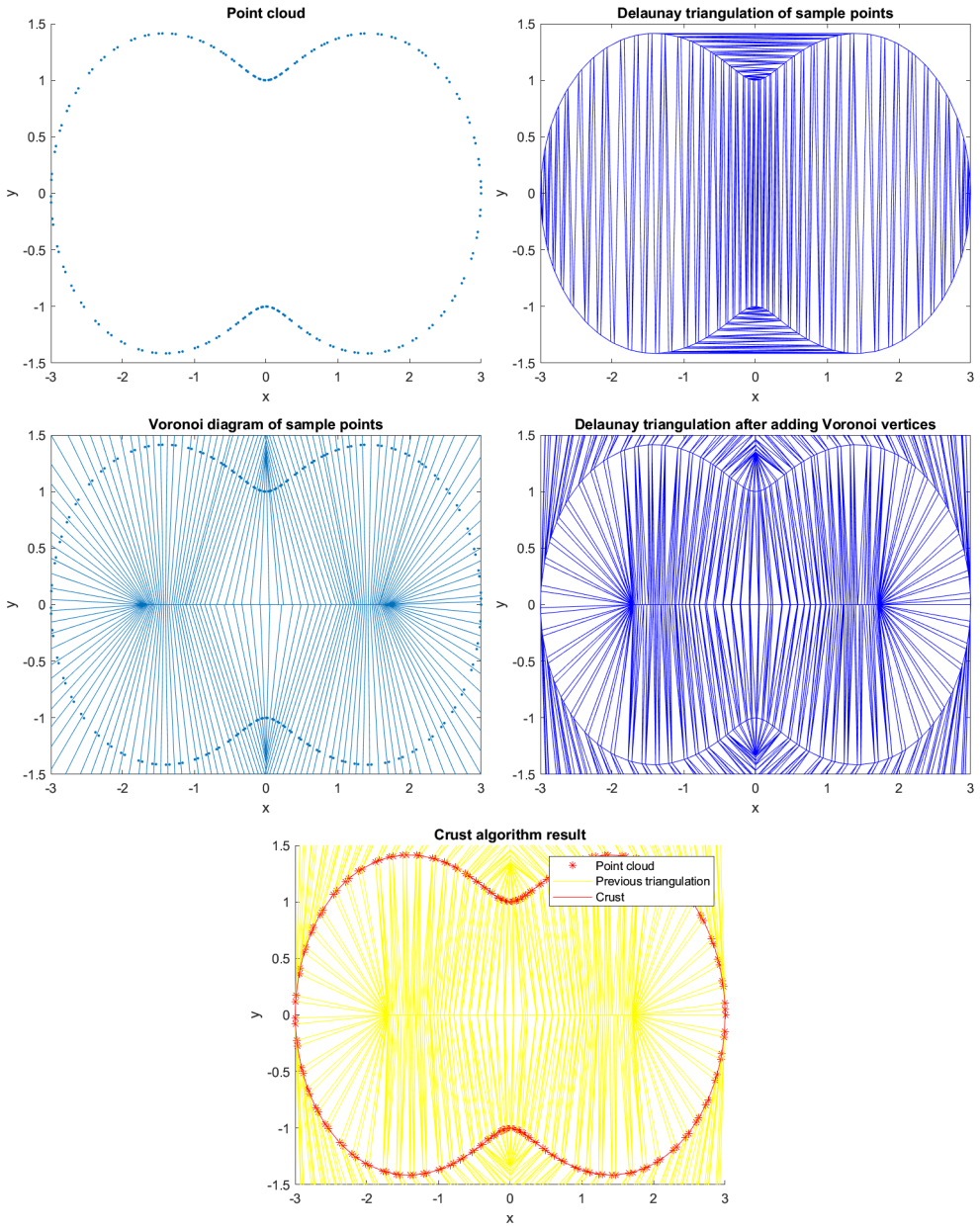


Figure 4.1: Crust algorithm steps

## 4.2 Medial Axis Approximation

It is possible to approximate the medial axis of a curve described by a point cloud using a slight modification of the above algorithm, this time only keeping the edges of the second triangulation if they are between two Voronoi vertices. For clarity, this algorithm will be referred to as the initial medial axis approximation.

As observed in Figure 4.2, while the medial axis obtained from the algorithm is generally following the skeleton (or, in terms of [5], the anti-crust), there are many additional triangles that need to be eliminated in order to achieve a satisfactory approximation. These triangles come from the second triangulation, where additional edges were constructed between Voronoi vertices.

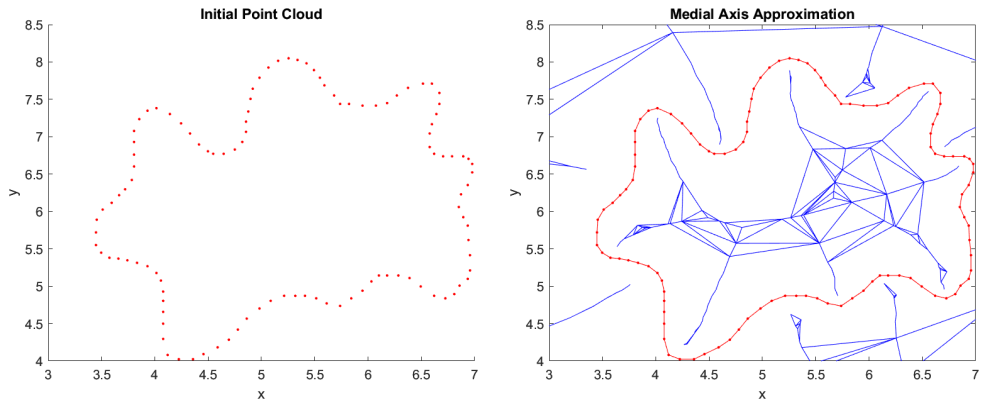


Figure 4.2: Initial medial axis approximation

To obtain the nested curves used for more intricate pattern synthesis, the initial medial axis algorithm is applied repeatedly on a set of points that increases in size at every iteration. More specifically, at every iteration, the Voronoi vertices (i.e. the medial axis vertices) of the previous iteration are added to the point set. From Figure 4.3, it can be further deduced that this initial medial axis approximation is not accurate enough to be used in future applications.

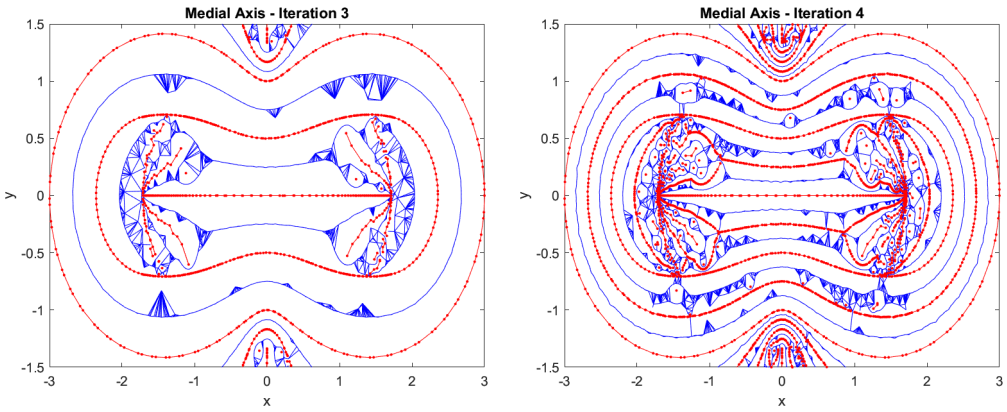


Figure 4.3: Third and fourth medial axis iterations for the point cloud presented in Figure 4.1 (data set in red, medial axis in blue)

# Section 5

# Medial Axis Approximation Refinement

As observed in the previous section, in order to generate patterns of interest using this method, the medial axis output needs to be significantly cleaner. To reduce the amount of additional edges and thus improve the initial approximation, a refinement algorithm is required. Two such algorithms are described in this section, one using a point-based filtering and the other using an edge-based approach. The latter is a new method that is then compared with the one-step filtering algorithm presented in [5].

## 5.1 Point Filtering

An attempt at refining the medial axis is made through pole-based filtering, a method known to work in three dimensional modelling [1]. The point of this algorithm is to eliminate non-necessary Voronoi vertices, thus not including them in the second triangulation and reducing the amount of additional triangles considered to be noise. Iterating over every cell of the Voronoi diagram, this method only keeps

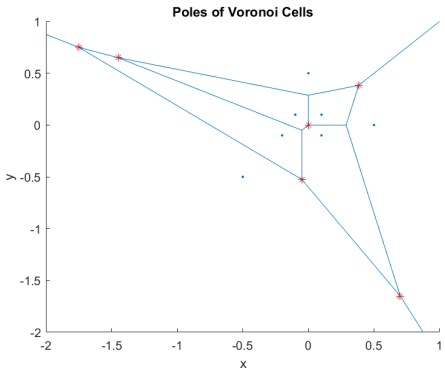


Figure 5.1: Voronoi poles for a random set of points

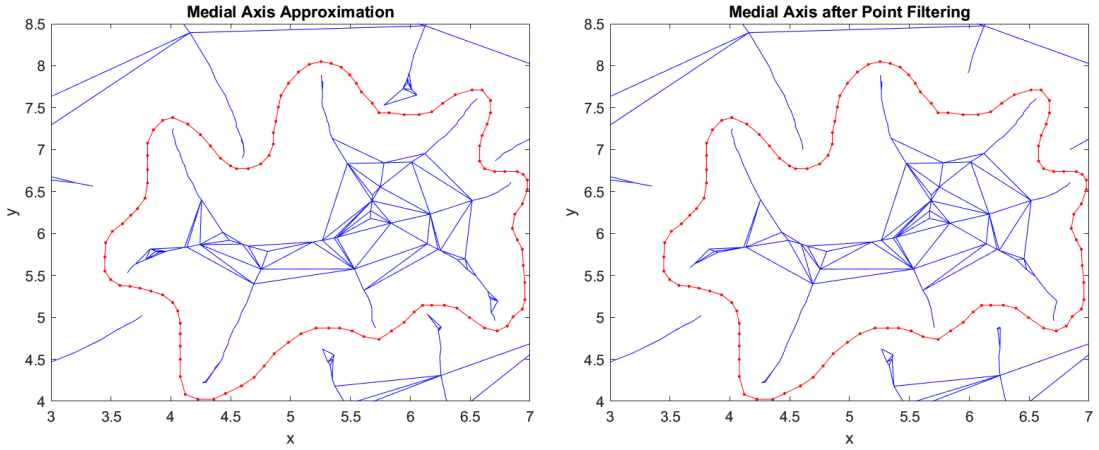


Figure 5.2: Medial axis before and after pole filtering

two of the Voronoi vertices that are adjacent to the polygon - the cell's poles - and discards the others (Figure 5.1). The first pole is the point farthest away from the generating point, and the second pole is the farthest point from the first pole that is also on the opposite side of the cell. To ensure that the second pole is on the opposite side of the first one, the implemented algorithm finds it by minimizing the inner product between the two. The comparison between the initial approximation and the pole refinement can be observed in Figure 5.2.

In the case of nested curves, the refinement of the medial axis is more visible, but still subtle. From both comparisons, it can be concluded that while this method slightly improves the medial axis output, the refinement is not significant enough to produce a relatively clean tree-like shape. Therefore, this technique is not as efficient in 2D as it is in 3D and it is not useful in the context of this thesis.

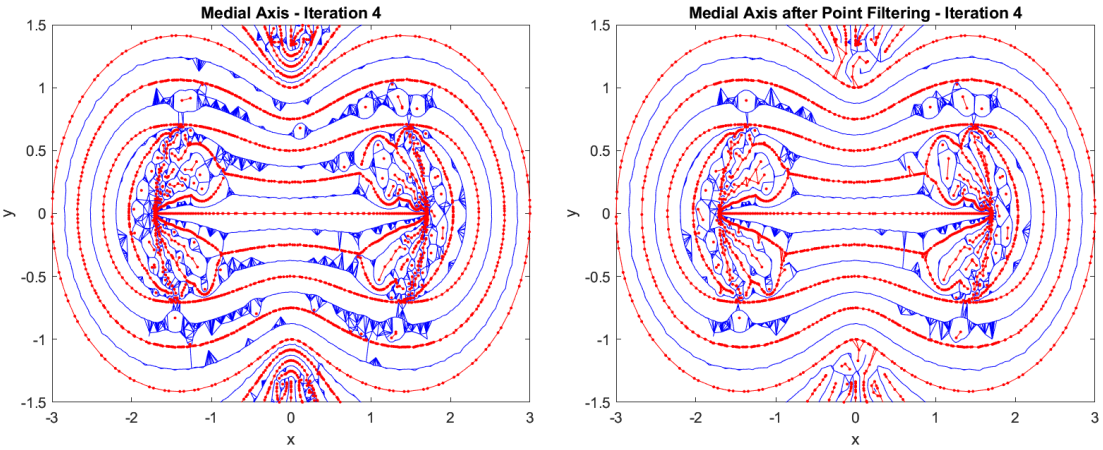


Figure 5.3: Nested curves: medial axes before and after pole filtering

## 5.2 Edge Filtering

The mathematical intuition behind this new algorithm is based on the convergence of Voronoi vertices to the medial axis. Then, since the approximation of the medial axis is a subset of the initial edges of the Voronoi diagram, this method filters the edges obtained in Section 4.2, only keeping those that were in the original Voronoi diagram. To identify these edges, the algorithm first identifies every triangle of the point cloud's Delaunay triangulation and its neighbors. Then, if two triangles are neighbors (i.e. they share an edge), by the duality of the Delaunay and Voronoi graphs, their circumcenters must be the end-points of a Voronoi edge which is added to a list. Given the output of Section 4.2, any medial axis approximation edge is kept only if it is also an initial Voronoi edge.

As it can be observed in Figure 5.4, not only does this method eliminate virtually all noise for closed curves, but it also provides a branched structure for the first point cloud chosen. A more comprehensive visual comparison can be observed in Figure 5.7, where the outputs for the same point cloud are given for the three algorithms: initial medial axis approximation, pole-filtering and edge-filtering.

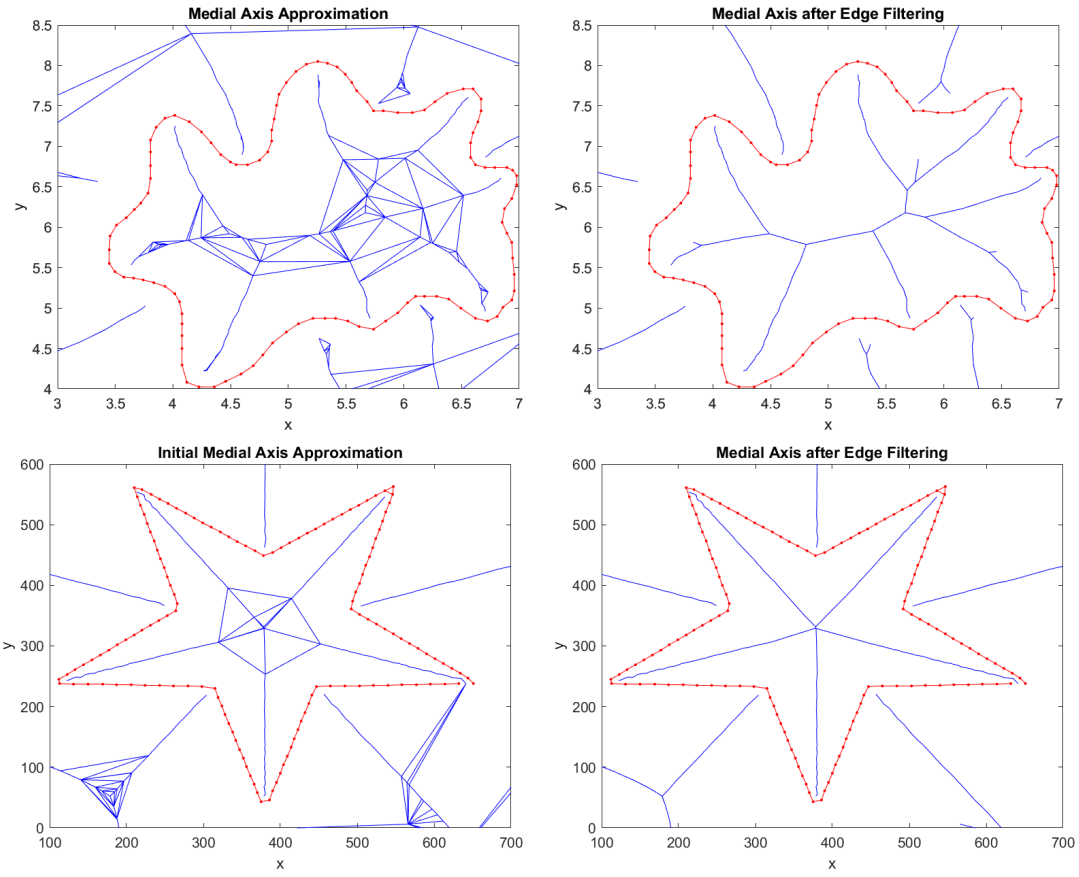


Figure 5.4: Medial axis before and after edge filtering

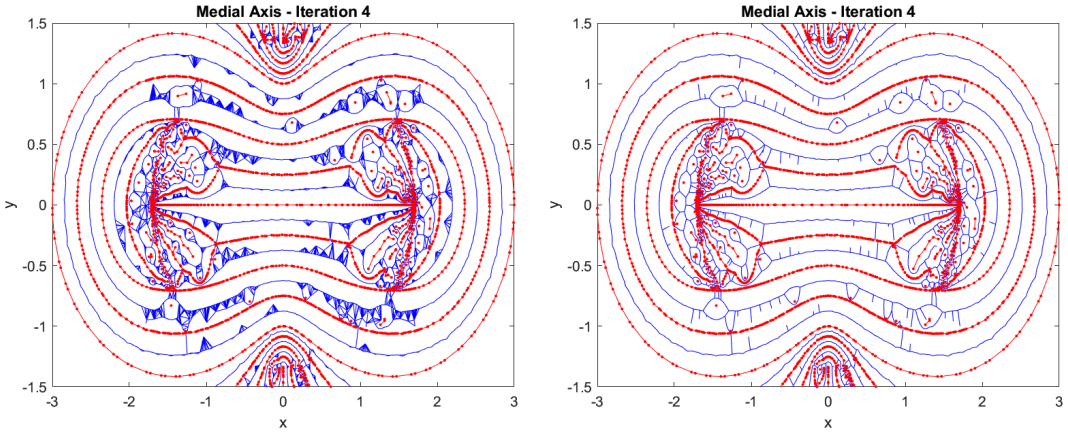


Figure 5.5: Nested curves: medial axes before and after edge filtering

In the case of nested curves (Figure 5.5), the improvement remains consistent and the refinement is much more significant compared to the pole filtering. As this algorithm proves to be better fitted to the scope of this thesis, this is the one that will be used in the applications that follow.

Figure 5.6 presents the nested curves of the point cloud in Figure 4.2 at different iterations using the edge filtering method to approximate the medial axis.

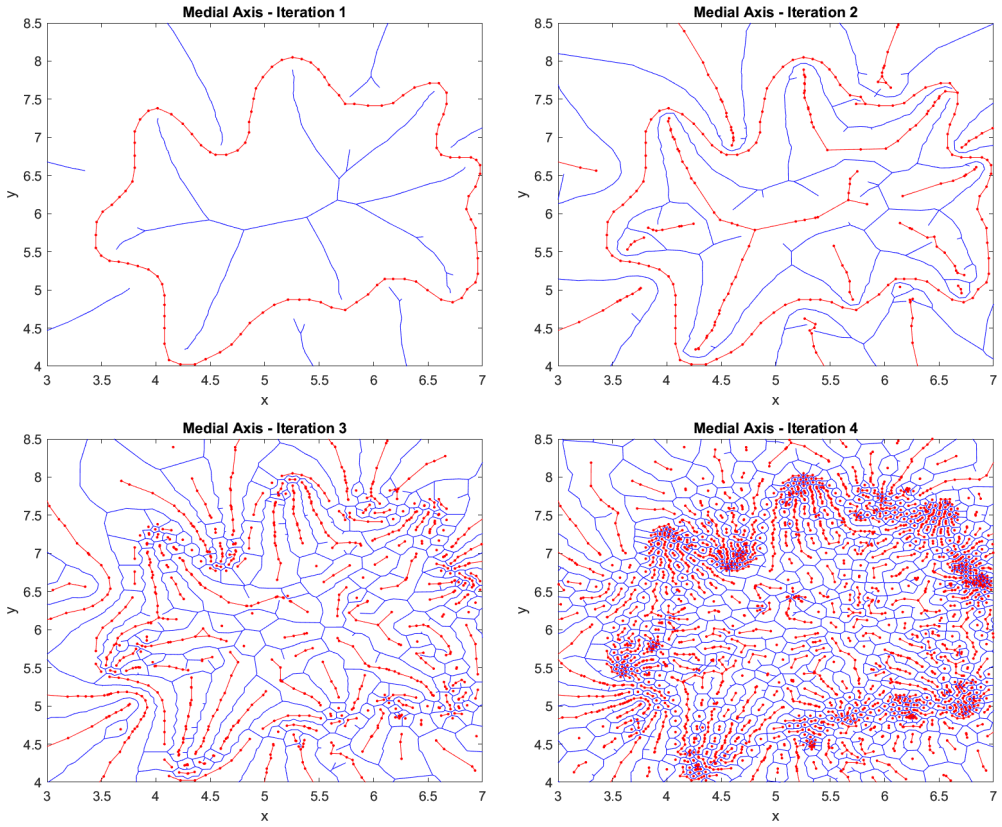


Figure 5.6: Nested curves at different iterations using the edge-filtering algorithm

## 5.3 Results Comparison

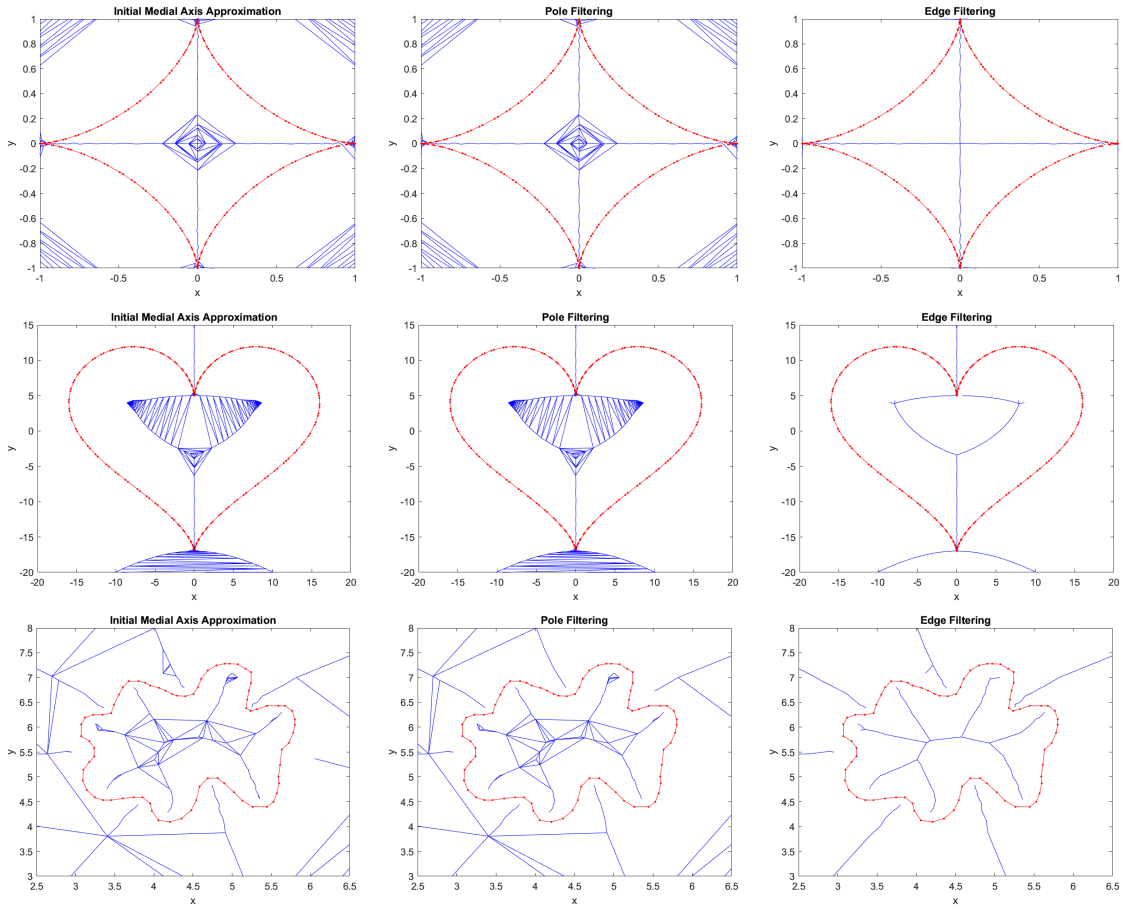


Figure 5.7: Comparison between medial axis algorithms

Figure 5.7 showcases the significant advantage of the edge-filtered medial axis over both the initial medial axis approximation output and the pole refinement one.

While the edge-filtering algorithm is clearly yielding better results for the scope of this thesis, it should be compared to other algorithms that proved useful in the medial axis refinement of two-dimensional shapes. One of these algorithms is the one-step method presented in [5], which is also based on information provided by the initial Voronoi diagram.

The one-step algorithm uses a Quad-edge data structure with two pointers for both Delaunay and Voronoi vertices. It relies on assigning every edge to either the crust or the anti-crust (the skeleton of the curve). More specifically, the algorithm checks whether a Voronoi edge crosses the Delaunay edge and, if it does, considers the latter as part of the crust. The edges that remain are then the anti-crust. The medial axis is a subset of the anti-crust, and can be approximated with further refinement and sampling density conditions.

When computing the crust, both algorithms can present mismatches where the

ideal sampling conditions are not met, and especially at acute or right angles, fact that can be observed in the maple leaf crust from Figure 5.8. The figure also shows that the edge-filtering algorithm produces a cleaner output which gives a good approximation of the medial axis, without having to refine the "hairs" mentioned in [5], which are a result of either the distribution of the sample set or its perturbations. It is also worth taking into account that, while the output of the edge-filtering algorithm is more convenient in terms of retrieving the medial axis, its complexity is higher than the one-step method. The limitations of this comparison are further discussed in the first part of Section 6.

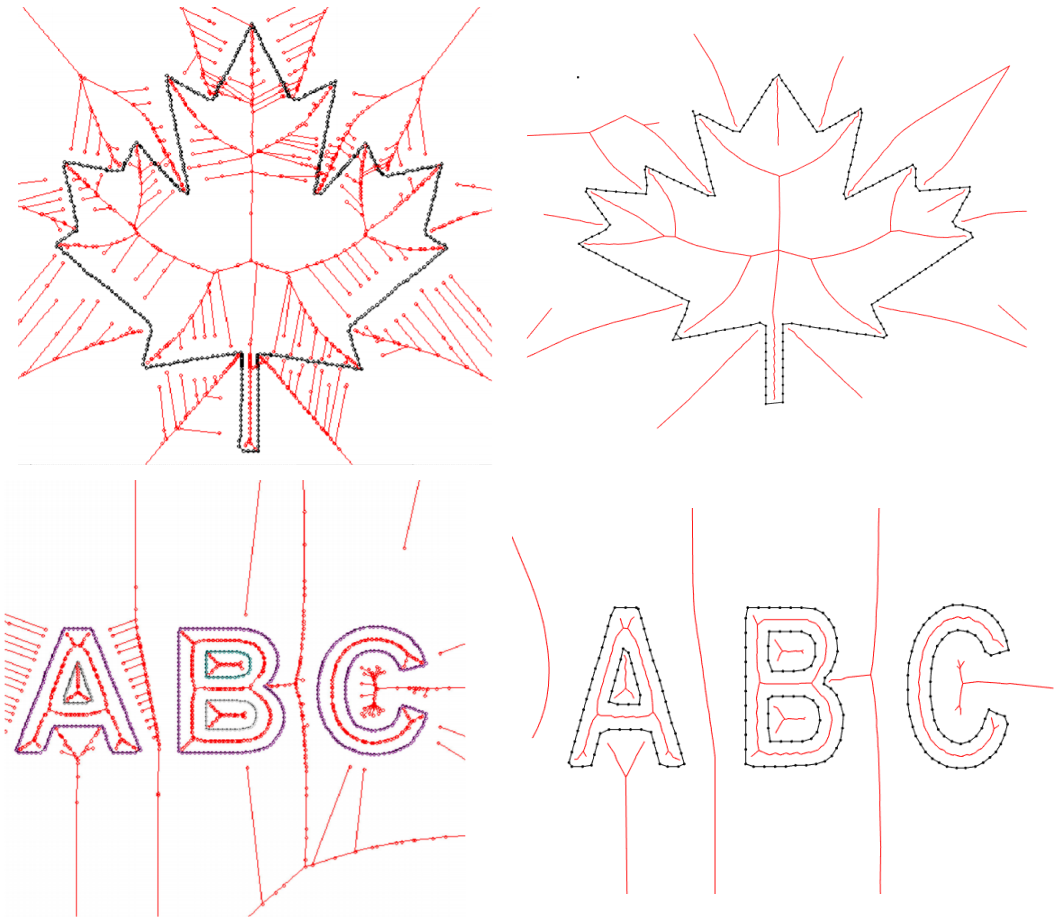

Figure 5.8: One-step skeleton extraction algorithm (left) compared to the edge-filtering medial axis refinement (right)

# Section 6

# Medial Axis Applications

Applications of the algorithm introduced previously are done on shapes of natural occurring structures, since branching patterns often appear in nature.

## 6.1 Maple Leaf

This subsection presents a short analysis of the branching structures obtained by applying the edge-filtering medial axis algorithm to the shape of a maple leaf.

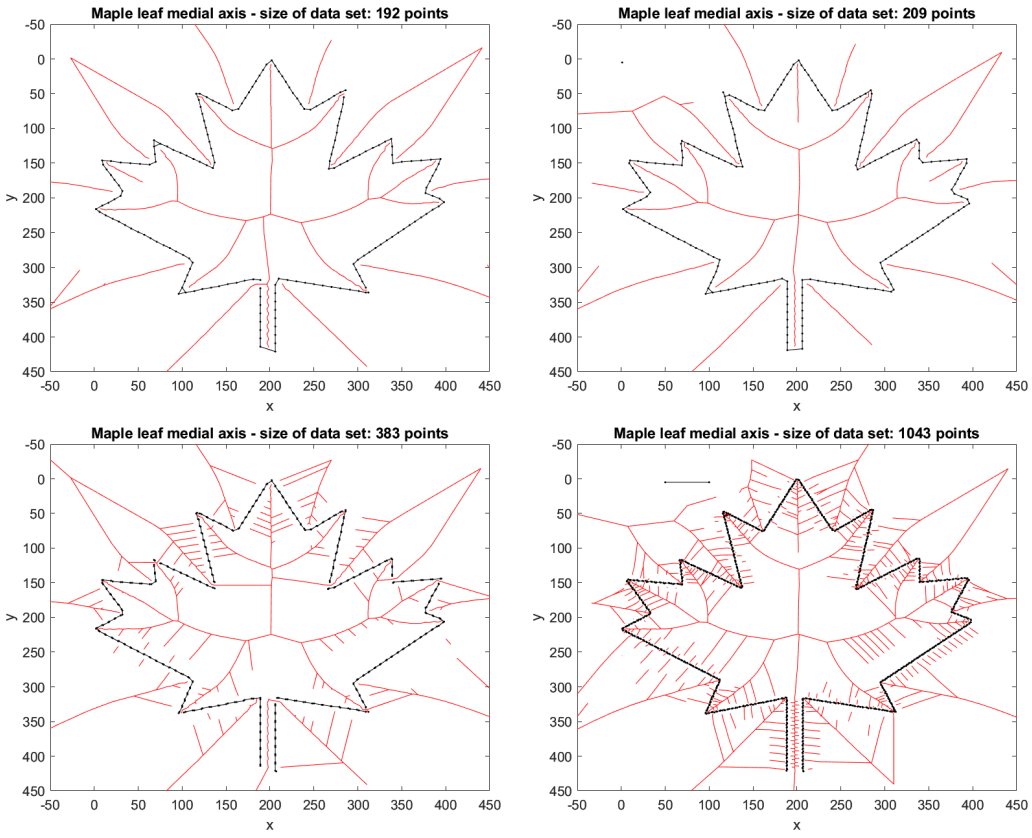


Figure 6.1: Maple leaf edge-filtered medial axis based on data sets of different sizes

An image analysis method is used to extract the contour of the shape of a maple leaf image. Then, the crust and refined medial axis algorithm is applied to contour subsets of different sizes. The sensitivity of the medial axis to boundary conditions as well as the data set dependency of the crust algorithm is clearly shown in Figure 6.1, which graphs the output of the algorithms for initial point clouds of 192, 209, 383, and 1043 points. It can be observed that the outputs in both the crust and the medial axis change significantly with small changes in the data set. The image analysis functions of MATLAB allow the extraction of the contour of a shape at different levels, or heights. The first two point clouds are subsets of the contour at height $z = 1$, which gives a cleaner output than the third and fourth point cloud, which are subsets of the contour at heights $z \in \{1, ..., 10\}$.

Furthermore, the data sets of 383 and 1043 points produce a medial axis that starts to resemble the results of [5], which brings forward the main limitations of the two algorithms comparison, namely the lack of information regarding the initial data set used in the one-step algorithm test and the differences in the point clouds used, even though the same shaped is analysed. However, given that there are featuring techniques to refine a noisy data set (such as the ones in [4]), this new edge-filtering algorithm is still valid.

## 6.2 Other Natural Occurring Patterns

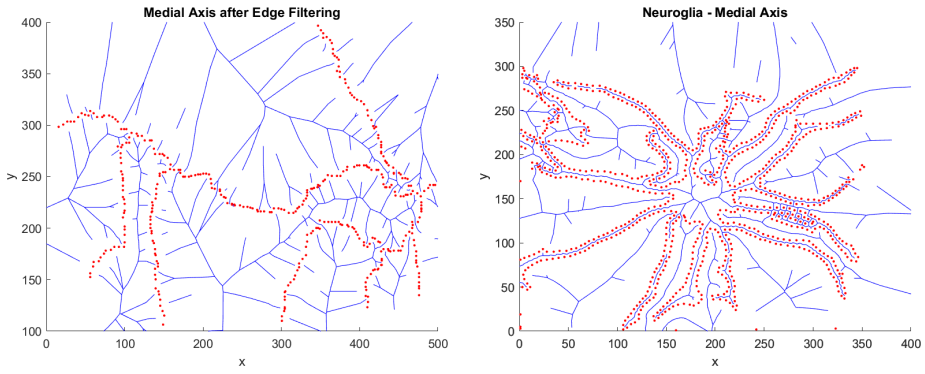


Figure 6.2: Medial axis of river network (left), medial axis of glial cell (right)

Figure 6.2 shows the medial axes of two natural occurring shapes. Firstly, taking a snippet of a river network as a data set, the algorithm produces branching structures that could either represent watersheds or mountain chains. According to [5], there is sufficient accuracy in the medial axis watersheds representation for the algorithm to prove useful beyond the field of visual computing. The second image displays a supporting cell found in the nervous system, and the external medial axis could represent neuron dendrites that they are surrounded by in reality. These images further show the applicability of branching patterns and the efficiency of the edge-filtering algorithm in producing them.

# Section 7

# Maze Generation

## 7.1 Voronoi Diagram Maze

Generating mazes based on graph theory methods is not a new topic in computer science. However, the methods presented in this section are using Voronoi diagrams to generate mazes that, through their irregular shapes, can be used in creating patterns such as cracks.

The maze generating algorithm starts with the Voronoi diagram of a set of points and uses the well-known Depth-first search algorithm (DFS). Having the initial Voronoi diagram, the algorithm first computes an adjacency matrix $A$, considering two Voronoi cells as neighbors if they have a common edge. This is equivalent to checking whether the generating points of the cells are connected in the Delaunay triangulation, since the two graphs are duals of each other. For clarification, this adjacency matrix follows convention such that for some cells $m, n$, $A(m, n) = 1$ if they are neighbors and 0 otherwise. Then, the DFS algorithm is applied. After creating an initially empty stack and choosing a random cell $i$ to be the current cell, the algorithm selects a random unvisited neighboring cell $j$ and deletes the common edge between the two by modifying the entry in the adjacency matrix ($A(i, j) = 0, A(j, i) = 0$). Then, the neighboring cell is pushed to the stack and marked as visited. The algorithm repeats with $j$ as the current cell. If there are no unvisited neighbors of the current cell, the algorithm backtracks by using the *pop*() function of the stack: the last cell in the stack become the current cell and the algorithm continues until the stack is empty (hence, all the cells were visited). The maze is then displayed by reconstructing the edges that still appear in the adjacency matrix.

Since the order of neighbors visited is randomized, there are several mazes that can be obtained from one Voronoi diagram. A solution to the mazes generated by this algorithm can be found using a well-known algorithm such as Breadth-first search (BFS).

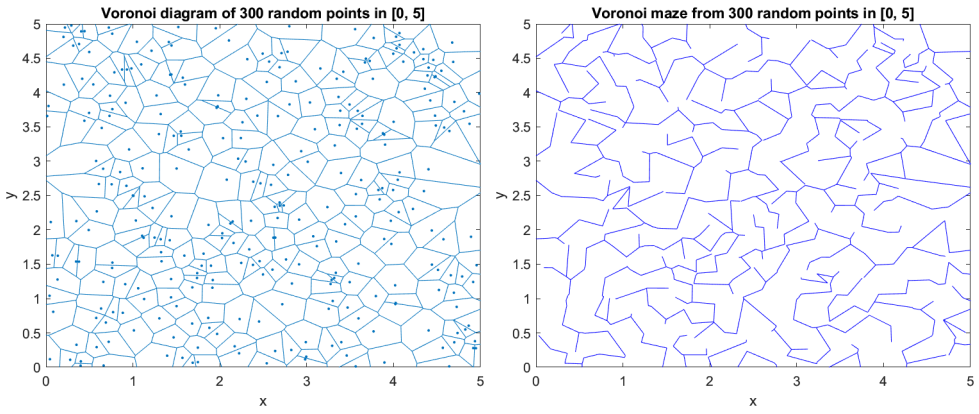


Figure 7.1: Voronoi diagram and one of its respective mazes

## 7.2 Delaunay Triangulation Maze

It could prove interesting at this point to apply the maze algorithm to a Delaunay triangulation. This can be achieved with small adjustments to the code to account for neighboring triangles, which is entirely done through the use of a connectivity list - a matrix of size $n \times 3$ where $n$ is the number of triangles in the triangulation. Each row represents a triangle and each value inside the matrix a vertex. Then, to find the vertices that belong to a specific triangle $i$, it is enough to read the output of the connectivity list's $i^{th}$ row. The output is shown in Figure 7.2, and while it generates an interesting maze, the angles of the shapes obtained are too sharp, such that the Voronoi diagram seems to be better suited to the purpose of this thesis, i.e. creating a satisfactory crack pattern.

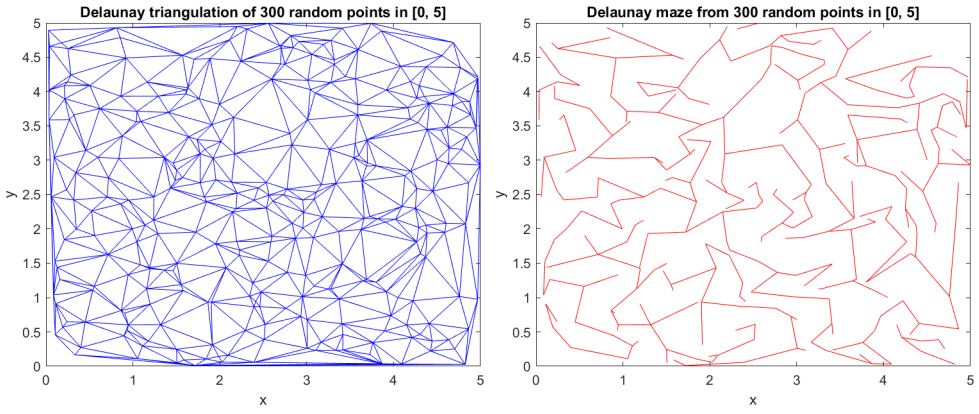


Figure 7.2: Delaunay triangulation and one of its respective mazes

# Section 8

# Crack Patterns

## 8.1 Uniform Distribution

By observing the maze in Figure 7.1, the potential of this application in synthesizing crack patterns becomes clearer. As Figure 8.1 shows, a Voronoi maze could, at this point, represent a rudimental crack pattern. This result was obtained after experimenting with data sets of different sizes, each data set containing randomly sampled points (in MATLAB, the default distribution for randomly generated points is the uniform distribution). The figure shows the path of the cracks, but the patterns can be improved in order to look more realistic through strategic shadowing and line thickness. The result of [6] presented in this figure is obtained by working with a mesh and using stress and relaxation fields, which makes it look very realistic but also more complex to produce. For 2D applications that require a fast algorithm for creating crack patterns and are not concerned with the physics behind them or their precision, the basic model obtained from the Voronoi maze could prove very useful.

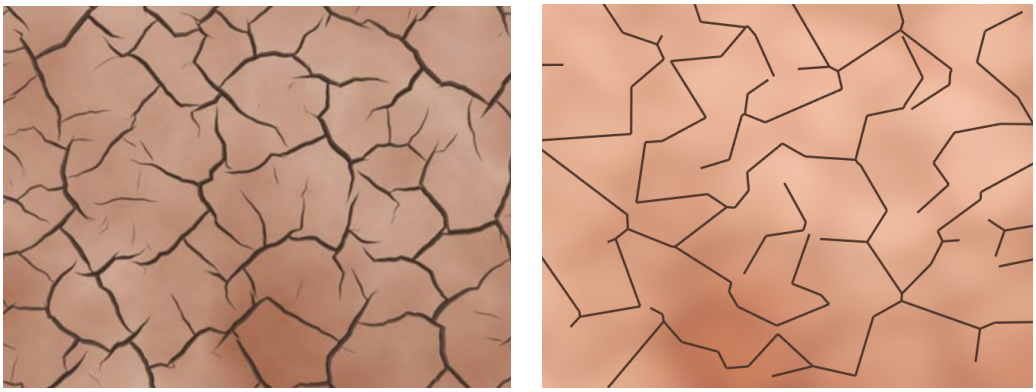

Figure 8.1: Crack pattern comparison between rendered dried mud obtained in [6] (left) and the maze algorithm on 75 random points (right)

## 8.2 Nested Curves

In order to generate more intricate mazes and patterns, the maze algorithm is applied to data points obtained by the nested curves algorithm introduced in Section 4. Figure 8.2 shows how the maze paths get narrower in the corners of the star shape and wider in areas with fewer points, which is to be expected considering the Voronoi diagram they derive from. What is interesting about the resulting mazes is the fact that they adapt to the contour of the shape, making them a useful application on their own. However, apart from using these results to generate mazes in specific shapes, they also represent the inspiration for the following idea. Given that, when a crack produces in an object, the cracks are more fine (or frequent) at the source of the distress and start getting progressively less frequent as they deviate from it, a maze pattern that would have narrower walls in the center and wider on the sides could better mimic a crack.

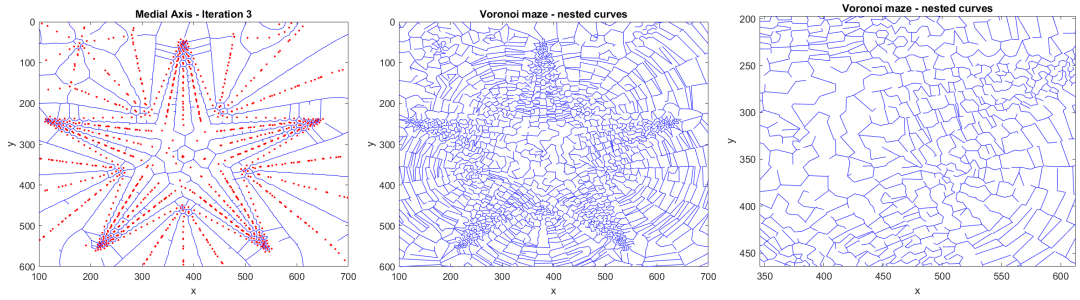


Figure 8.2: Nested curves obtained from a star shape at the third iteration (left), their respective maze algorithm (middle) and an enlarged snippet of it (right)

## 8.3 Gaussian and Laplace Distributions

To achieve this type of maze, a set of points that is denser in the center and progressively less dense towards the edges of the graph is wanted. Then, instead of using uniformly distributed random points to construct an initial Voronoi diagram, the use of another distribution is needed. One that fits the description above is the Gaussian distribution (also known as normal distribution). In particular, the standard normal distribution will be used, with expectation $\mu = 0$ and variance $\sigma^2 = 1$. Another fitting choice is the Laplace distribution with location parameter $\mu = 0$ and scale parameter $b = 1$. As the probability density functions in Figure 8.4 show, both distributions

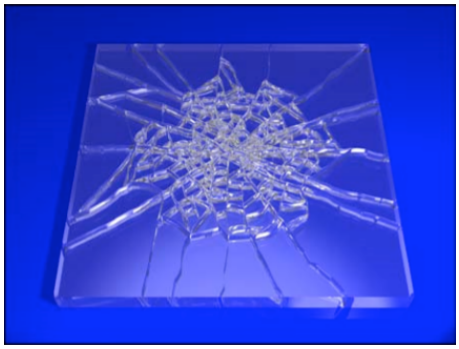

Figure 8.3: Crack obtained in [6]

can generate a set of points similar to the one described.

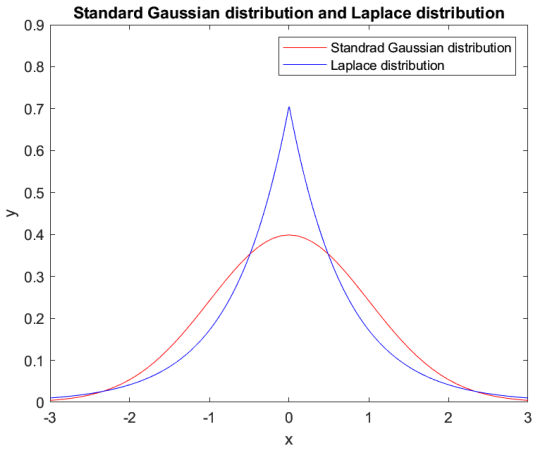


Figure 8.4: Probability density functions for the two distributions

A main difference is that with Laplace distribution there will be more points around the source of the distress than with the normal distribution (since the probability density function is more peaked). Furthermore, the points might fit more nicely within a crack pattern, since in the Laplace distribution there are eventually more points near the edges of the figure (this can also be observed in Figure 8.5).

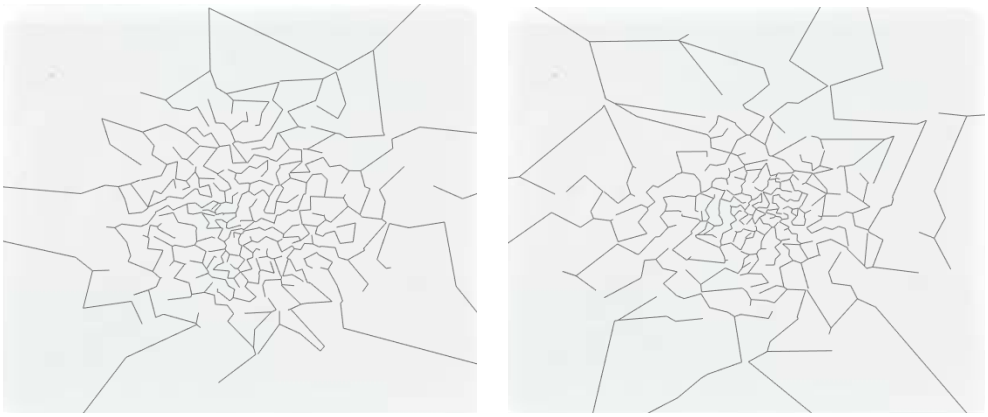

Figure 8.5: Crack patterns obtained from 300 points distributed according to the standard normal distribution (left) and the Laplace distribution (right)

The outputs of Figure 8.5 are a two-dimensional simplification of the crack produced in Figure 8.3 by using stress and relaxation fields over a mesh of 4094 triangles. Both distributions produce satisfactory crack patterns, but the Laplace one seems to better mimic reality. The results of this section are a very convenient and efficient approach to creating crack patterns in two dimension, and they can prove valuable in the field of visual computing.

# Section 9

# Conclusions and Future Work

The two new methods for creating two-dimensional patterns presented in this paper are useful in computer graphics for multiple reasons. Firstly, both algorithms are relatively fast to compute and do not need an extensive data set to produce sufficiently well defined patterns. Secondly, they give a simple, easy to implement solution to problems usually solved by more complex methods, which can be valuable for applications that are not concerned with the physical accuracy of the patterns, but the pattern itself. Furthermore, both synthesis methods are highly dependent on the data set, which makes the outputs customizable and easy to adapt to different applications. However, this could be viewed as a main limitation, given that it makes the outputs unpredictable. In fact, most limitations of the presented methods are related to the distribution of the data set, which is convenient since there are known methods of refining a noisy sample of points (such as the one presented in [4], used specifically for medial axis computation).

Between the two algorithms presented, the use of maze generation to model crack patterns represents a more valuable contribution to the computer graphics research field since it is a new approach. However, the refinement of the medial axis and its use to generate branching patterns can prove more useful outside of the Computer Graphics field, since the medial axis itself can be used to mimic the position and behaviour of patterns found in nature.

There are many improvements and further work that can be done building on the results of this paper. Regarding the medial axis pattern generation algorithm, there are plenty of applications that can be studied and implemented, impacting fields such as biology and orometry. Even though not the center aim of this thesis, the generation of mazes that adapt to the contour of a shape as well as the generation of a solution can be particularly interesting as an application on their own. Lastly, the mazes used for crack pattern generation can be implemented on data sets with different distributions to fit various types of cracks.

It can be concluded that this thesis achieves its goal of creating interesting 2D patterns, building on well-known concepts and algorithms in order to create new methods that could enrich the field of visual computing.

# Section 10

# References


[1] Nina Amenta and Marshall Bern. "Surface reconstruction by Voronoi filtering". In: *Discr. Comput. Geom. 22* (1999). DOI: 10.1145/276884.276889.

[2] Nina Amenta, Marshall Bern, and David Eppstein. "The Crust and the -Skeleton: Combinatorial Curve Reconstruction". In: *Graphical Models and Image Processing* 60.2 (1998), pp. 125–135. ISSN: 1077-3169. DOI: https://doi.org/10.1006/gmip.1998.0465. URL: http://www.sciencedirect.com/science/article/pii/S1077316998904658.

[3] Tamal K. Dey and Wulue Zhao. "Approximate medial axis as a Voronoi subcomplex". In: *Computer-Aided Design* 36.2 (2004). Solid Modeling and Applications, pp. 195–202. ISSN: 0010-4485. DOI: https://doi.org/10.1016/S0010-4485(03)00061-7. URL: http://www.sciencedirect.com/science/article/pii/S0010448503000617.

[4] Chazal F. and Lieutier A. "The -medial axis". In: *Graphical Models (GMOD)* 67.4 (2005), pp. 304–331. DOI: https://doi.org/10.1016/j.gmod.2005.01.002. URL: http://www.sciencedirect.com/science/article/pii/S1524070309000058.

[5] Christopher Gold. "Crust and Anti-Crust: A One-Step Boundary and Skeleton Extraction Algorithm". In: *Proceedings of the ACM Conference on Computational Geometry* (Jan. 1999). DOI: 10.1145/304893.304971.

[6] Hayley N. Iben and James F. O'Brien. "Generating surface crack patterns". In: *Graphical Models* 71.6 (2009). 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 2006), pp. 198–208. ISSN: 1524-0703. DOI: https://doi.org/10.1016/j.gmod.2008.12.005. URL: http://www.sciencedirect.com/science/article/pii/S1524070309000058.

[7] Pooran Memari, Patrick Mullen, and Mathieu Desbrun. "Parametrization of Generalized Primal-Dual Triangulations". In: *Proceedings of the 20th International Meshing Roundtable*. Ed. by William Roshan Quadros. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. ISBN: 978-3-642-24734-7.

ÉCOLE
POLYTECHNIQUE
UNIVERSITÉ PARIS-SACLAY

**Statement of Academic Integrity Regarding Plagiarism**

I, the undersigned Bărbulescu Maria, hereby certify on my honor that:

1. The results presented in this report are the product of my own work.
2. I am the original creator of this report.
3. I have not used sources or results from third parties without clearly stating thus and referencing them according to the recommended rules for providing bibliographic information.

**I hereby declare that this work contains no plagiarized material.**

**10/04/2020**