Voronoi Treemaps

Christopher Oser

Institute of Interactive Systems and Data Science (ISDS),
Graz University of Technology
A-8010 Graz, Austria

07 Oct 2021

Abstract

This seminar paper introduces the reader to the topic of Voronoi treemaps, a data visualization technique for hierarchically structured data. The technique combines two separate fields: Voronoi diagrams and treemaps. The techniques have no obvious connection, but can be merged to produce a practical data visualization.

In order to establish a good understanding of Voronoi treemaps, this paper explains the concept of Voronoi diagrams and some of its history. It also touches on the construction of Delaunay tessellations and their connection to Voronoi Diagrams. Finally, treemaps and the application of Voronoi diagrams to treemaps are presented, followed by an overview of current work and applications around Voronoi treemaps.

© Copyright 2021 by the author(s), except as otherwise noted.

This work is placed under a Creative Commons Attribution 4.0 International (CC BY 4.0) license.

Contents

List of Figures

Chapter 1

Introduction

A Voronoi treemap is a data visualization technique through which hierarchically structured data is represented recursively with Voronoi diagrams. It combines two major techniques, namely Voronoi diagrams and treemaps. To introduce the reader to the topic, the process is explained step by step.

First, Voronoi diagrams are presented in Chapter 2, covering the underlying theory as well as history and applications of the technique. Next, in Chapter 3, the reader is given an overview of the dual of the Voronoi diagram, the Delaunay tessellation, and its relevance to the Voronoi diagram is explained.

Chapter 4 describes treemaps and Voronoi treemaps. This chapter also looks at current approaches to and applications of Voronoi treemaps, and presents some active work on the subject at Graz University of Technology. Finally, Chapter 5 concludes this survey paper with an outlook and pointers to further resources.

Chapter 2

Voronoi Diagrams

Voronoi diagrams are used to split a plane into regions surrounding a given set of points, called *sites*. In simple terms, a Voronoi diagram indicates which points on the plane are closest to which of the sites. Hence, the plane is divided into regions of influence of the chosen sites. This has many uses, ranging from pure aesthetics to data analytics. A simple form of the Voronoi diagram for three sites can be seen in Figure 2.1.

2.1 Applications

Voronoi diagrams have a wide variety of uses. While some find the diagram itself to be pleasing to the eye, computer scientists use them for many functional applications. This includes basic distance problems, the classical postman problem, classification, and various clustering algorithms [Aurenhammer et al. 2013, pages 3–5]. Furthermore, it is used in computer graphics to generate textures.

Voronoi diagrams are also used in other fields, often under other names. The Thiessen polygons in geometry or meteorology, Wigner-Seitz zones in chemistry and physics, and domains of action in crystallography are examples of Voronoi diagrams in other fields [Aurenhammer et al. 2013, pages 1–2]. In biology, Voronoi diagrams are used to model cells or bone micro architecture [Bock et al. 2010; Li et al. 2012]. Some uses of Voronoi diagrams, such as nearest neighbor query, span multiple areas of science [Okabe et al. 2000, pages 10–11].

2.2 History

As far as is known, the first mentions of Voronoi diagrams were in the 17$^{\text{th}}$ century when René Descartes claimed the solar system consisted of vortices. While he did not explicitly define the regions, the process behind his idea is very close to what is understood to be Voronoi diagrams nowadays [page 1 Aurenhammer et al. 2013; Okabe et al. 2000, page 6].

The idea was formalized by the mathematicians Carl Friedrich Gauß in 1840, Gustav Lejeune Dirichlet in 1850, and Georgi Feodosjewitsch Voronoi in 1908. These formalizations led to the structure that is now considered to be a Voronoi diagram [page 3 Aurenhammer et al. 2013; Okabe et al. 2000, page 6].

2.3 Basic Theory

To construct a Voronoi diagram, a set of at least three points, or *sites*, is required. The term sites is used to better differentiate them from generic points on the plane. A Voronoi diagram of just two sites is simply the bisector, which is the locus of all points at equal distance to these two sites, as can be seen in Figure 2.2.

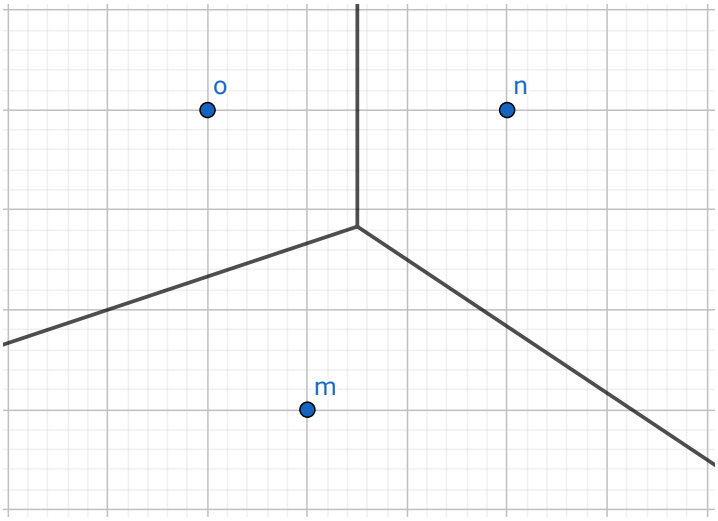

Figure 2.1: The Voronoi diagram for the three points (sites) **m**, **n**, and **o**. [Illustration created by the author using GeoGebra [GeoGebra 2021].]

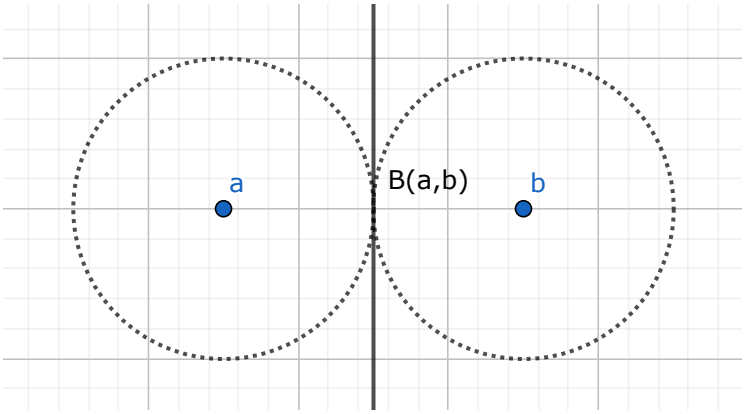

Figure 2.2: The bisector describes the locus of all points at equal distance to two points. [Illustration created by the author using GeoGebra [GeoGebra 2021].]

In its basic form, the Voronoi diagram is constructed on a Euclidean plane which means all distances are given as:

$$d(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2} \tag{2.1}$$

for two sites, a and b. With this, the half plane between two sites can be defined as:

$$H(a,b) = \{x | d(a,x) \leq d(b,x)\}, \tag{2.2}$$

where the bisector of a and b acts as a boundary for this half plane. A Voronoi region is then formed by taking the intersection of all $n-1$ half planes in the set of n sites. Such a region can be seen in Figure 2.3 in the form of a triangle around point **g**. Such a Voronoi region contains all points closest to its site. In Figure 2.3, this means that all points on the plane within the triangle are closer to **g** than to any of the other sites (**e**, **f**, **h**).

Two more definitions are necessary to understand a Voronoi diagram. A Voronoi *vertex* is any endpoint within the Voronoi diagram. The edges connecting the Voronoi vertices are known as Voronoi *edges*. Both can be seen in Figure 2.3.

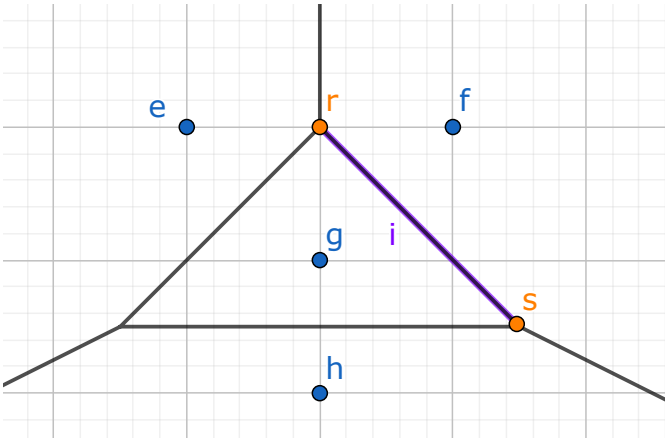

Figure 2.3: The triangle around point **g** indicates its Voronoi region. The line segment **i** is a single Voronoi edge. The two points that define the Voronoi edge, **r** and **s**, are examples of Voronoi vertices. [Illustration created by the author using GeoGebra [GeoGebra 2021].]

2.4 Algorithms

There are four types of construction algorithms for Voronoi diagrams. While they do not vary greatly in performance, they have very different approaches to building the diagram.

2.4.1 Incremental

The most intuitive way of constructing a Voronoi diagram is the incremental method. To construct the diagram, the sites from the set are added to the plane one by one. Every time a new site is added, the diagram is adapted to the new site. This means finding all now faulty edges, deleting them, and adding the new correct edges to accommodate the new site. Once all sites have been added, the final Voronoi diagram has been constructed.

This method has the benefit of being on-line, which means the set of sites does not have to be complete when starting the algorithm as they are read in one by one. Additional sites can be added later on. The downside to this algorithm is that it has a worst case runtime of $O(n^2)$. However, the expected runtime for a randomized algorithm is $O(n \log n)$ [Aurenhammer et al. 2013, pages 18–23].

2.4.2 Divide & Conquer

The Divide & Conquer algorithm for Voronoi diagrams needs the entire set of sites right from the start. This is because, as the name suggests, it splits the set into a left half and a right half, according to the placement of the sites on the plane. Once the sites are split up, this is repeated recursively, until the only sets remaining have two or three sites each. For these small sets, simple Voronoi diagrams are easily constructed. Once the diagrams are constructed for all sets, the smaller diagrams are merged back together. After the final two diagrams have been merged, this represents the finished Voronoi diagram for the set of sites.

This algorithm has an upper boundary of $O(n \log n)$ for all cases and therefore performs better than the incremental algorithm in general [Aurenhammer et al. 2013, pages 24–27].

2.4.3 Sweep Line (Fortune's Algorithm)

First introduced by Steven Fortune [Fortune 1986], the sweep line algorithm uses a line which moves across one axis of the plane generating events in a queue as it moves. The closer the event is to the sweep line, the earlier it will be handled. There are two types of events: site events and circle events. Site events occur at each site in the set. A circle event is created during a site event and describes a point at which

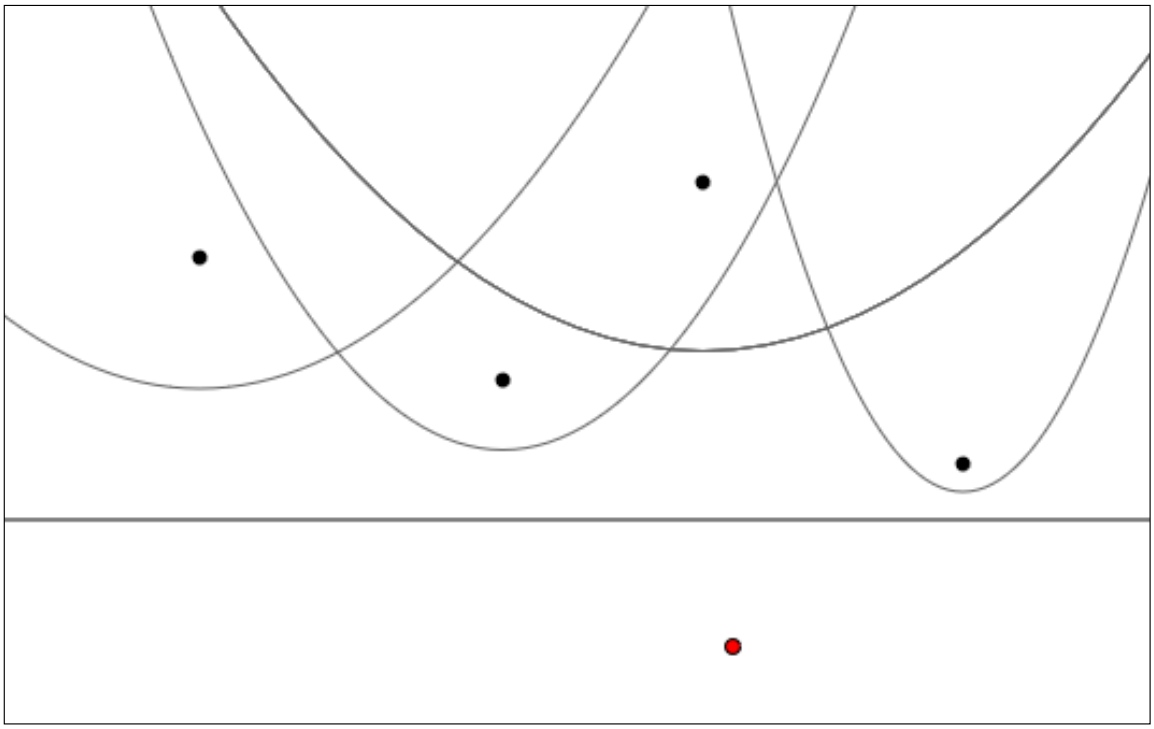

Figure 2.4: The black horizontal line is the sweep line moving downwards. All the black sites and their events have already been handled. The red dot indicates a circle event, through which most likely a new Voronoi vertex is found. [Screenshot taken by the author using a self-constructed demonstrator.]

a parabola vanishes from the beach line that is formed by the site events. During a circle event, a new Voronoi vertex can appear.

As soon as the sweep line has crossed the entire plane, the Voronoi diagram is constructed. This algorithm also has an upper boundary of $O(n \log n)$, which means it is comparable in performance to the Divide & Conquer algorithm. A screenshot of the algorithm in action can be viewed in Figure 2.4 [Aurenhammer et al. 2013, pages 28–30].

2.4.4 Lift to 3-Space

This construction algorithm works by lifting the sites into the 3$^{\text{rd}}$ dimension. This is usually done via a simple projection like:

$$z = x^2 + y^2. \tag{2.3}$$

This projection can be viewed in Figure 2.5. Each site is lifted up from a 2-dimensional plane to be represented on a paraboloid. Once the projection is done, the convex hull of all sites on the paraboloid is required. There already exist many algorithms that can be used to compute the convex hull of the sites. These algorithms have the added advantage that they have been used for some time and are optimized and easily accessible.

By translating the resulting convex hull back to two dimensions, what remains is the Voronoi Diagram of the specified set of sites. As with the previous algorithms, the upper boundary on runtime is $O(n \log n)$ [Aurenhammer et al. 2013, pages 31–34].

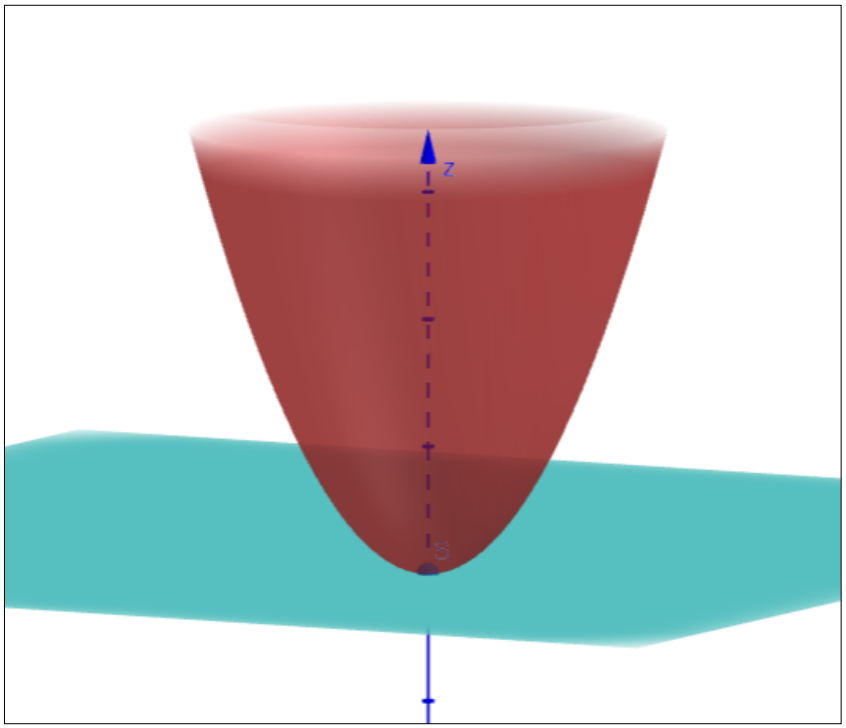

Figure 2.5: Sites in a plane are lifted up to 3-dimensional space by being projected onto a paraboloid. In this case $z = x^2 + y^2$. [Illustration created by the author using GeoGebra [GeoGebra 2021].]

2.5 Weighted Voronoi Diagrams

In data visualization, it is often desirable to assign higher weight to certain properties. This is also possible for Voronoi diagrams. Weighted Voronoi diagrams can be constructed in a number of ways. The idea remains the same: the higher the weight of a certain site, the larger the resulting Voronoi region should become. This can be seen in Figure 2.6. The standard Voronoi diagram of the four sites is represented by the dashed line. The solid line is a weighted Voronoi diagram, where higher weight has been assigned to sites **a** and **b**, increasing the size of their regions.

One way of computing weighted Voronoi diagrams is by using *additively weighted* or *multiplicatively weighted* diagrams. This means the usual distance calculation, as in 2.1, is substituted by either:

$$add_d(a, b) = d(a, b) - weight(a) \tag{2.4}$$

for an additively weighted diagram or by:

$$mult_d(a, b) = d(a, b)/weight(a) \tag{2.5}$$

for a multiplicatively weighted diagram. The main drawback of these methods is that they yield curved rather then straight edges, which is often undesirable.

Thus, *power diagrams* are often used when weights should be taken into account. In power diagrams, circles or spheres define the area of influence of a site. The circles' or spheres' radius is calculated via the assigned weight. For example for the site **p**:

$$r(p) = \sqrt{weight(p)} \tag{2.6}$$

This method then also yields a power function through which the influence of a site on a point can be calculated:

$$pow(x, p) = (x - p)^T (x - p) - weight(p) \tag{2.7}$$

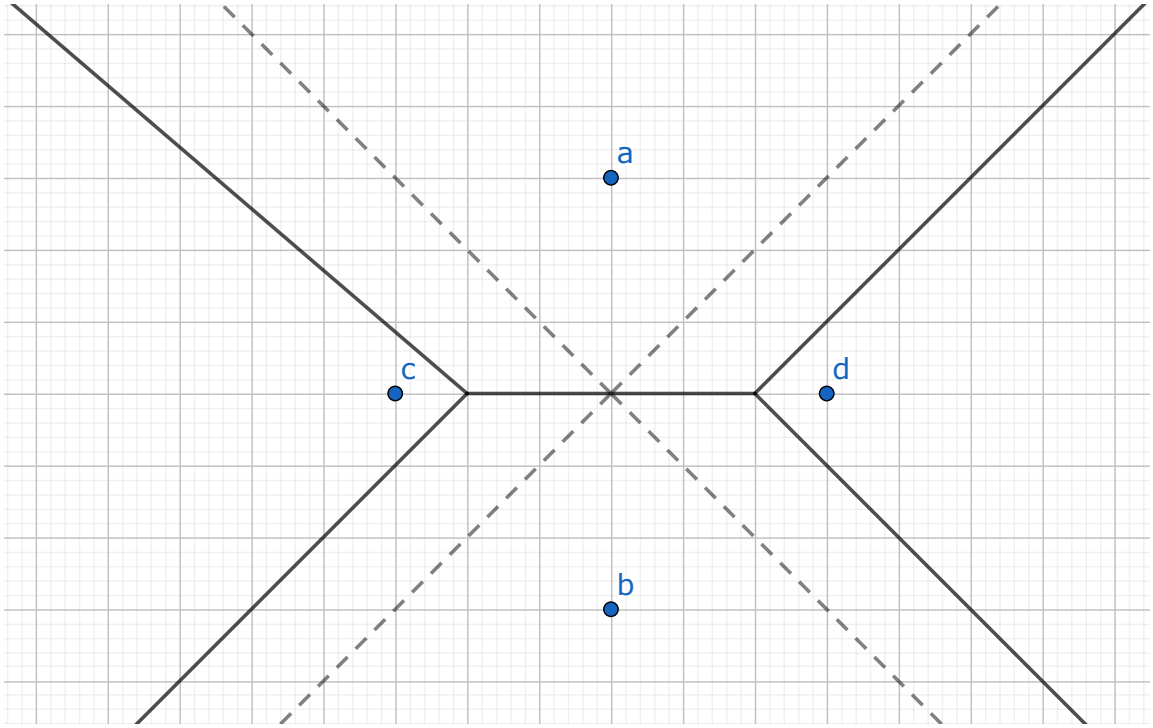

Figure 2.6: The dashed line represents the standard Voronoi diagram of the four sites. The solid line is a weighted Voronoi diagram in which the sites **a** and **b** have been assigned higher weight, increasing the size of their respective regions. [Illustration created by the author using GeoGebra [GeoGebra 2021].]

The function is 0 on the outline of the circle or sphere and negative within. For all points outside of the circle or sphere it returns a positive value. An example of a power diagram for three points can be seen in Figure 2.7.

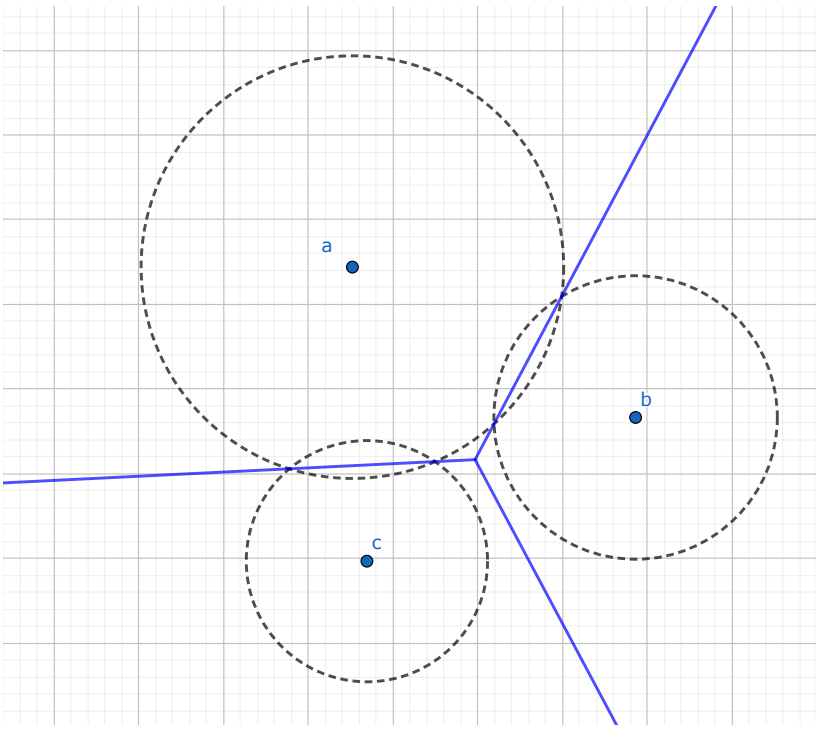

Figure 2.7: The power diagram for the three sites is represented by the blue lines. As can be seen from the dashed circles, site **a** has a larger weight assigned to it compared to the other two sites. This results in its region being considerably larger. [Illustration created by the author using GeoGebra [GeoGebra 2021].]

Chapter 3

Delaunay Tessellation

When discussing the Voronoi diagram, it is imperative to also talk about the Delaunay tessellation, as they are connected in many ways. While a Voronoi diagram constructs regions around every site in the set, the Delaunay tessellation uses the sites as endpoints for edges that form polygons [Aurenhammer et al. 2013, pages 11–14]. The Delaunay tessellation is the dual of the Voronoi diagram, and one can be derived from the other.

3.1 Construction

To construct a Delaunay tessellation, all sites which are co-circular, while not including any other sites within the circle, are joined via an edge. So for every two sites that can be connected via a circle, one edge to the tessellation can be added [Aurenhammer et al. 2013, page 12].

3.2 Delaunay Triangulation

If the sites used for the Delaunay tessellation are in general position, it turns into a Delaunay triangulation, since all polygons in the tessellation are triangles. Sites are in general position, if no three of them are co-linear and no four of them are co-circular. In this case, any three sites that are co-circular and whose circle does not include other sites, can be joined to form a new triangle in the triangulation. Such a Delaunay triangulation can be seen in Figure 3.1 [Aurenhammer et al. 2013, page 12].

3.3 Relevance to Voronoi Diagrams

The Delaunay tessellation or Delaunay triangulation is the dual of the Voronoi diagram. This means they have several properties that relate to each other and also means that one can be constructed from the other. For every Voronoi edge there is a Delaunay edge, for every Voronoi vertex there is a Delaunay face, and for every Delaunay vertex there is a Voronoi region. This can also be seen in Figure 3.1. This duality is very useful for many algorithms. Whenever one structure is given, the other can be computed, which provides many possibilities to approach the construction of either one of them [Aurenhammer et al. 2013, page 12].

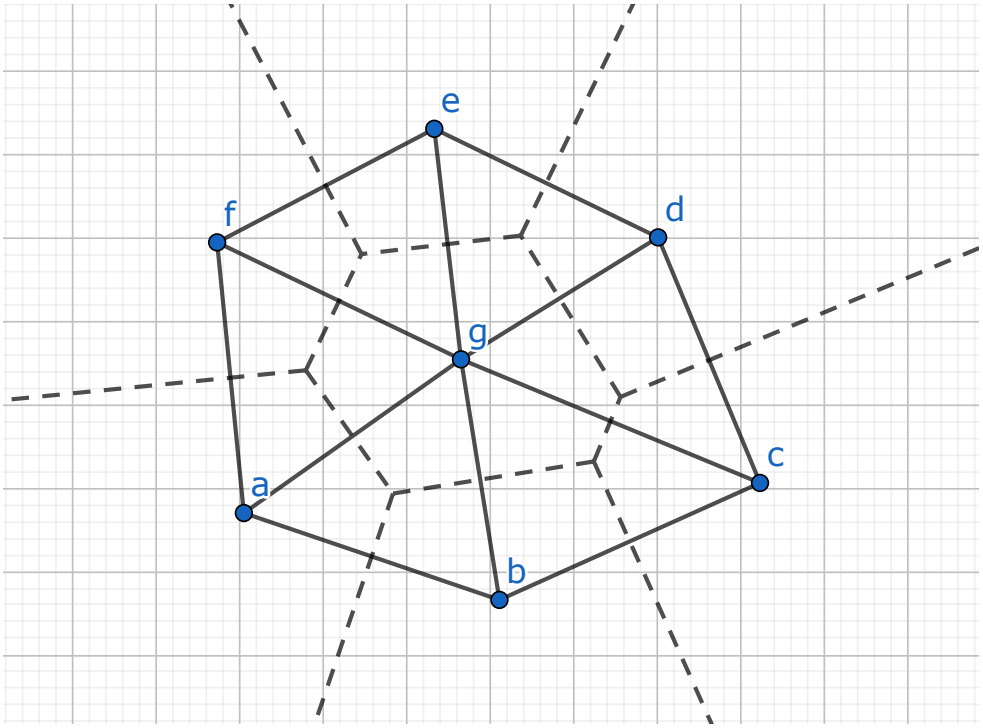

Figure 3.1: The dashed line represents the Voronoi diagram of the set of seven sites. The solid line is the corresponding Delaunay tessellation, which in this case is a Delaunay triangulation, since the sites are in general position. [Illustration created by the author using GeoGebra [GeoGebra 2021]]

Chapter 4

Voronoi Treemaps

Voronoi treemaps are a combination of treemaps and Voronoi diagrams with the goal of visualizing data hierarchies in a practical way. In standard treemaps, the space allocated to children is divided into strips or rectangles recursively. In Voronoi treemaps, the space is partitioned into Voronoi regions. In interactive treemaps, levels of the tree can be explored by zooming, expanding, and collapsing. An example of a Voronoi treemap can be seen in Figure 4.1.

4.1 Treemaps

Treemaps were first introduced by Shneiderman [1992] in the 1990s. His team at the University of Maryland then also developed the Treemap application, which was improved and maintained until 2004 [Shneiderman 2004].

The idea is to visualize hierarchical data with nested rectangles. The size and color of the rectangles is determined by attributes of the corresponding tree nodes. A screenshot of the Treemap application is shown in Figure 4.2. The visualization can be explored interactively by the user. Double-clicking a node drills down into that subtree, right-clicking moves up one level.

4.2 Voronoi Treemaps

A Voronoi treemap is a data visualization which combines treemaps with Voronoi diagrams. Rather than recursively subdividing rectangles, the space allocated to children at each level is determined by constructing Voronoi regions.

The polygons in a Voronoi treemap are based on Voronoi diagrams. So for every polygon in the visualization a site is placed to construct Voronoi regions for all the sites in the set, which is based on the underlying hierarchical data. This way numerous different polygons are created and the available space in the plane can be used optimally. Using weighted Voronoi diagrams, the size of the polygons to be adjusted to represent an attribute of the data.

The ideal Voronoi treemap is interactive. With a static flat visualization, the data can still be represented for smaller data sets, but it will become very crowded and unintuitive, the larger the data set becomes. By making the visualization interactive, it is possible to hide deeper layers initially and render them only when necessary. This can be managed either by monitoring the user's zoom focus and level or through dedicated choices the user makes to expand or collapse certain children.

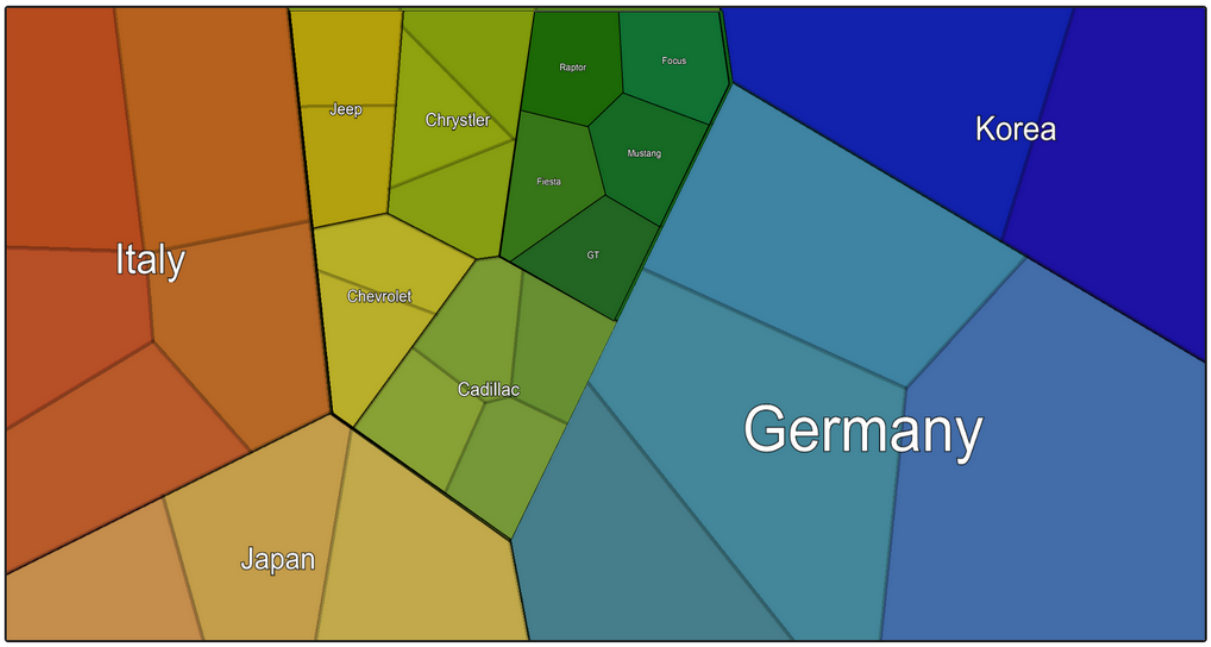

Figure 4.1: A hierarchy containing models of car shown in the Interactive Voronoi Treemap (IVT) application. The first level indicates nationality, the second level manufacturer, and the third level model of car. [Screenshot taken by the author using IVT [Oser et al. 2021].]

Figure 4.2: Hierarchically structured data from the NBA basketball league shown with Treemap 4.1.1. At the top level, the two divisions (Central and Atlantic) are displayed on the left and right. The next level shows the teams in each division. Finally, the colored rectangles show individual players. Here, the size of the respective rectangle corresponds to the minutes per game and the color to the points per game of each player. [Screenshot taken by the author using Treemap 4.1.1 [Shneiderman 2004]]

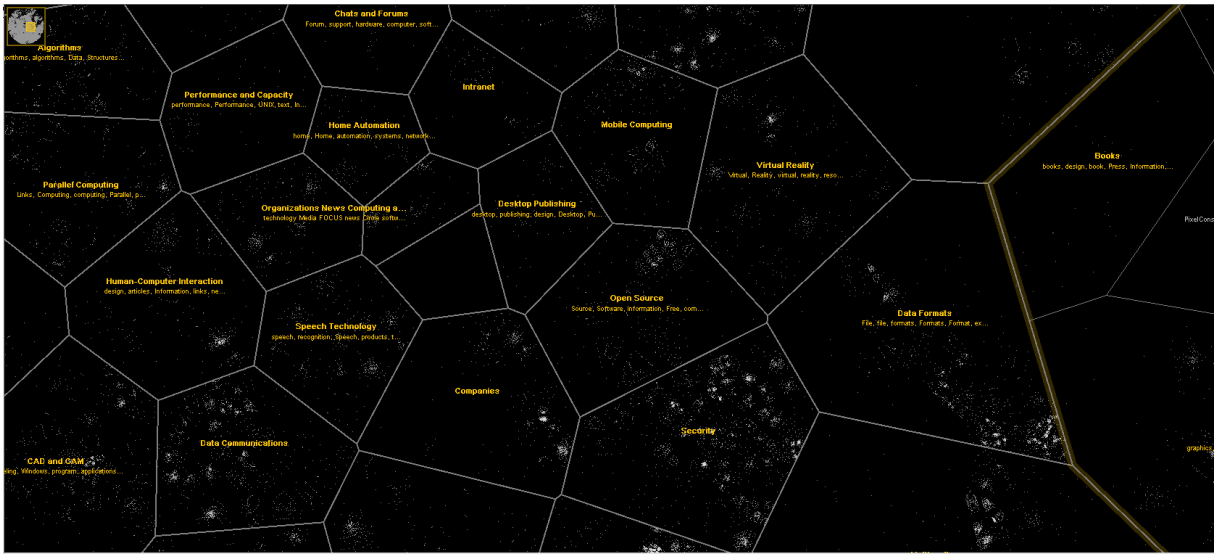

Figure 4.3: Part of the Mozilla Open Directory hierarchy [Mozilla 2017] shown in InfoSky. The size of a region is based on the number of entries in that branch. Each entry is a textual description of a site on the web and is represented as a white dot. The dots resemble stars in a galaxy. [Screenshot taken by the author using the InfoSky application [Andrews et al. 2002].]

4.3 History

The Voronoi treemaps technique was first introduced in 2002 by Andrews et al. [2002] and Granitzer et al. [2004]. A team at Graz University of Technology and their partners developed a visual data explorer based on Voronoi diagrams called InfoSky. Hierarchy levels were represented as Voronoi regions and every subsequent level was made up of Voronoi regions within the parent region, as can be seen in Figure 4.3. Individual documents in the hierarchy were represented by white dots. Three years later, Balzer and Deussen [2005] published their paper on Voronoi treemaps, which coined the name for this technique, without referring to the prior work.

4.4 Implementations

Currently the only published Voronoi treemap implementation is FoamTree by Carrot Search [Carrot Search 2021]. It is a commercial software library, but offers a free branded version too. FoamTree is written in JavaScript and is intended to be integrated into web pages. The application is highly customizable and allows broad personalization options to address any wishes a user might have. Even though the possibilities are numerous with FoamTree, there is a large overhead that needs to be added to the construction of every visualization. Data sets need to be in their expected format and no standard formats, such as basic .json or .csv files, are supported. Also any personalization requires in-depth knowledge of the application documentation or prior knowledge of Carrot Search applications.

In 2020, development on another Voronoi treemap implementation started at Graz University of Technology under the working name “Interactive Voronoi Treemap” (IVT) [Oser et al. 2021]. Just like FoamTree, it is a web-based application that offers an interactive Voronoi treemap that can be browsed by the user. Whilst offering similar basic functionality, IVT is free of charge and open source. It works with both standardized .json and .csv files and also offers example data sets to test the application. For calculation of the Voronoi diagrams, IVT uses the d3-voronoi-treemap library [LeBeau 2020] and PIXI.js [Groves 2021] for all visual aspects. A screenshot of the application can be seen in Figure 4.1. Features still to be implemented include: improvement of the UI, support for touch devices, browsing by zooming, and some performance optimization.

Chapter 5

Concluding Remarks

This paper serves as an introduction to Voronoi treemaps, which combine treemaps with Voronoi diagrams. Good starting points for further reading about Voronoi diagrams are Aurenhammer et al. [2013] and Okabe et al. [2000]. Further reading on treemaps can be found in Shneiderman and Wattenberg [2001] and Shneiderman [1992]. They include actual implementations, theoretical algorithms for constructing treemaps, as well as the math behind the structures.

Work on the Interactive Voronoi Treemap (IVT) is still ongoing, although perhaps the project will be renamed. The goal is to produce a viable open-source implementation of Voronoi treemaps.

Bibliography

Andrews, Keith, Wolfgang Kienreich, Vedran Sabol, Jutta Becker, Georg Droschl, Frank Kappe, Michael Granitzer, Peter Auer, and Klaus Tochtermann [2002]. *The InfoSky Visual Explorer: Exploiting Hierarchical Structure and Document Similarities*. Information Visualization 1.3/4 (Dec 2002), pages 166–181. doi:10.1057/palgrave.ivs.9500023 (cited on page 15).

Aurenhammer, Franz, Rolf Klein, and Der-Tsai Lee [2013]. *Voronoi Diagrams and Delaunay Triangulations*. World Scientific, Aug 2013. ISBN 9814447633. doi:10.1142/8685 (cited on pages 3, 5–6, 11, 17).

Balzer, Michael and Oliver Deussen [2005]. *Voronoi Treemaps*. Proc. IEEE Symposium on Information Visualization 2005 (InfoVis 2005) (Minneapolis, MN, USA). 2005, pages 49–56. doi:10.1109/INFVIS.2005.1532128. https://researchgate.net/publication/225075883_Voronoi_Treemaps (cited on page 15).

Bock, Martin, Amit Kumar Tyagi, Jan-Ulrich Kreft, and Wolfgang Alt [2010]. *Generalized Voronoi Tessellation as a Model of Two-dimensional Cell Tissue Dynamics*. Bulletin of Mathematical Biology 72.7 (Oct 2010), pages 1696–1731. doi:10.1007/s11538-009-9498-3. https://arxiv.org/pdf/0901.4469.pdf (cited on page 3).

Carrot Search [2021]. *FoamTree*. 28 May 2021. https://carrotsearch.com/foamtree/ (cited on page 15).

Fortune, Steven [1986]. *A Sweepline Algorithm for Voronoi Diagrams*. Proc. Second Annual Symposium on Computational Geometry (SCG '86) (Yorktown Heights, New York, USA). ACM, 02 Jun 1986, pages 313–322. ISBN 0897911946. doi:10.1145/10515.10549 (cited on page 5).

GeoGebra [2021]. *GeoGebra*. 28 May 2021. https://geogebra.org/ (cited on pages 4–5, 7–9, 12).

Granitzer, Michael, Wolfgang Kienreich, Vedran Sabol, Keith Andrews, and Werner Klieber [2004]. *Evaluating a System for Interactive Exploration of Large, Hierarchically Structured Document Repositories*. Proc. IEEE Symposium on Information Visualization (InfoVis 2004) (Austin, Texas, USA). Oct 2004, pages 127–134. doi:10.1109/INFOVIS.2004.19 (cited on page 15).

Groves, Mat [2021]. *PixiJS*. 28 May 2021. https://pixijs.com/ (cited on page 15).

LeBeau, Franck [2020]. *d3-voronoi-treemap*. 10 Dec 2020. https://github.com/Kcnarf/d3-voronoi-treemap (cited on page 15).

Li, Hui, Kang Li, Taehyong Kim, Aidong Zhang, and Murali Ramanathan [2012]. *Spatial Modeling of Bone Microarchitecture*. Three-Dimensional Image Processing (3DIP) and Applications II (Burlingame, California, United States). Edited by Atilla M. Baskurt and Robert Sitnik. 8290. International Society for Optics and Photonics. SPIE, 30 Jan 2012, pages 232–240. doi:10.1117/12.907371. https://cse.buffalo.edu/DBGROUP/bioinformatics/papers/Hui_SPIE2012.pdf (cited on page 3).

Mozilla [2017]. *Mozilla Open Directory Project (DMOZ)*. 14 Mar 2017. https://archive.org/web/20170314015905/http://www.dmoz.org/ (cited on page 15).

Okabe, Atsuyuki, Barry Boots, Kokichi Sugihara, and Sung Nok Chiu [2000]. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. 2nd Edition. Wiley, 2000. ISBN 0471986356. doi:10.1002/9780470317013 (cited on pages 3, 17).

Oser, Christopher, Lisa Weissl, Markus Ruplitsch, and Romana Gruber [2021]. *Interactive Voronoi Treemap*. 06 Apr 2021. https://github.com/somestudentcoder/IVT (cited on pages 14–15).

Shneiderman, Ben [1992]. *Tree Visualization with Tree-Maps: 2-d Space-Filling Approach*. Transactions on Graphics 11 (Jan 1992), pages 92–99. doi:10.1145/102377.115768. https://cs.umd.edu/~ben/papers/Shneiderman1992Tree.pdf (cited on pages 13, 17).

Shneiderman, Ben [2004]. *Treemap 4.1.1*. 17 Feb 2004. https://cs.umd.edu/hcil/treemap/ (cited on pages 13–14).

Shneiderman, Ben and Martin Wattenberg [2001]. *Ordered Treemap Layouts*. Proc. IEEE Symposium on Information Visualization 2001 (InfoVis 2001) (San Diego, California, USA). 22 Oct 2001, pages 73–78. doi:10.1109/INFVIS.2001.963283. https://cs.umd.edu/~ben/papers/Shneiderman2001Ordered.pdf (cited on page 17).