Algorithmic Figurative Tree Structures

David A. Reimann

Math. and Computer Science Dept., Albion College, Albion, MI, USA; dreimann@albion.edu

Abstract

A process for constructing a figurative tree that fills a specified connected bitmap region is presented. The process uses a Voronoi diagram based on sampled points to create a connected graph. A spanning tree is then constructed and drawn using Bézier curves. An example of using this process to create a tree that fills a text phrase is shown for several controlling parameters.

Introduction

Branching structures arise in many natural settings. The structure of trees and other woody plants often starts from a main stem that grows by sending offshoots which repeatedly branch to fill available growing space. River watersheds are formed by small streams that merge with other small streams, which merge to form rivers, which merge to create larger and larger rivers. Branching structures are seen in the vascular structures, airways, and nervous systems of animals. Minerals such as magnesium oxide form elaborate dendrites. Similar structures are seen in Lichtenberg figures and other forms of electrical discharge.

The hierarchical structure of trees lends itself to many applications in mathematics, statistics, and computer science. It is common to use trees for data relationships, such as computer file systems, organization charts, relationship diagrams, and document structures. Trees have applications in parsing, graph theory, searching, and other recursive systems. Presenting hierarchical information with trees, such as tournament brackets, is also quite common.

Given the pervasiveness of trees, it is no surprise that artists throughout history have included trees in their art [6]. Paleolithic rock paintings contain representations of trees. Leonardo da Vinci wrote in his notebooks on how to draw trees in various weather and lighting conditions [3]. Contemporary artists, including the author and others exhibiting art at Bridges Conferences, have featured trees in their works. Trees themselves are the media for bonsai and topiary artists.

This paper describes a algorithmic technique for the creation of tree-like structures that fill a pre-defined space, resulting in figurative branching structures similar to topiary or sculpted vines. Similar methods have been described for creating renderings of gray-scale images using minimal spanning trees [5, 2].

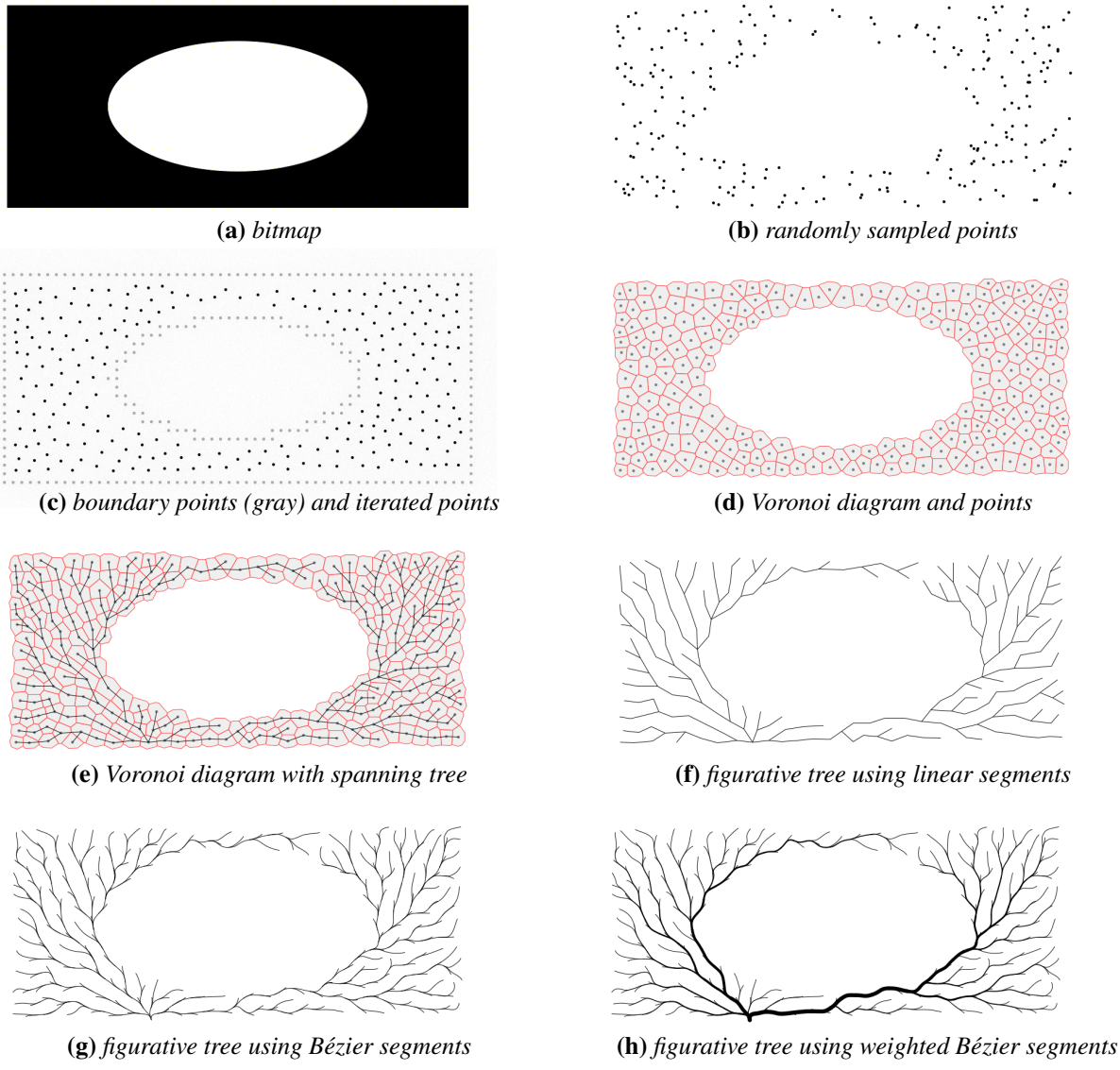
Process

Figure 1 illustrates the construction process for a figurative tree. Figure 1(a) shows a bitmap of the region where the figurative tree will be created. The black region of the bitmap is randomly sampled to create a reasonable number of points as seen in Figure 1(b). The points will become the internal (branching) and external (terminal) nodes of the tree, thus the number of points controls the complexity of the final tree.

The boundary of the region is then sampled to create an additional set of control points shown in gray in Figure 1(c). The sampled points are combined with the boundary points and a Voronoi diagram [8] is computed, as seen in Figure 1(d). The boundary points limit the boundary effects when creating the associated Voronoi diagram by keeping the interior cells relatively compact. In Figures 1(c) and 1(d), the

Voronoi diagram was computed, then Lloyd's algorithm [1] was used to replace the position of the control points with the centroid of the Voronoi cell, creating an approximate centroidal Voronoi tessellation; that process was iterated five times. This results in random points, but more equally spaced; compare the random points in Figure 1(b) and iterated points in Figure 1(c).

The Voronoi diagram is associated with an undirected and unweighted combinatorial graph based on cell adjacencies. A starting point of this graph is selected and Dijkstra's algorithm [4] is used to find the shortest path from the starting point to every other point, resulting in a spanning tree depicted in Figure 1(e). The tree using only straight line segments is shown in Figure 1(f). Because of the Voronoi diagram construction process, the line segments between adjacent cells pass perpendicularly through their shared boundary.



(a) *bitmap*

(b) *randomly sampled points*

(c) *boundary points (gray) and iterated points*

(d) *Voronoi diagram and points*

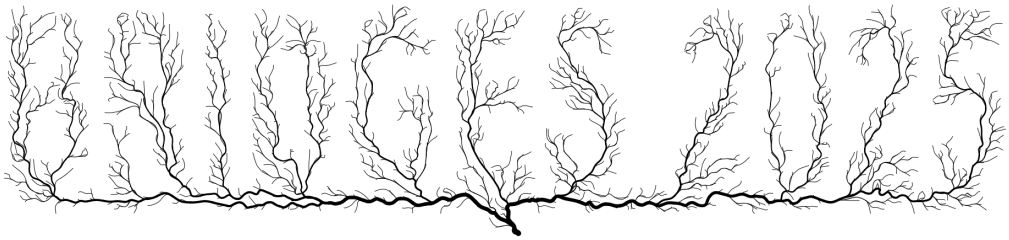
(e) *Voronoi diagram with spanning tree*

(f) *figurative tree using linear segments*

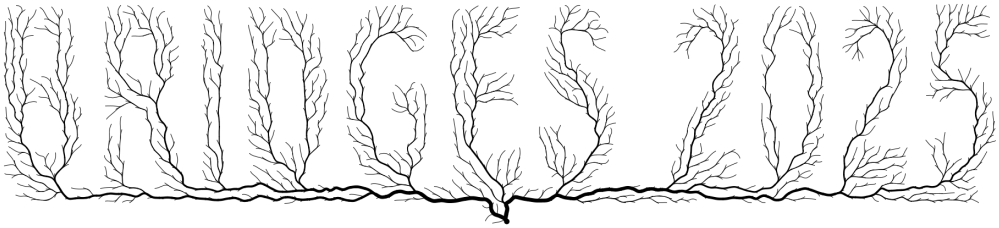
(g) *figurative tree using Bézier segments*

(h) *figurative tree using weighted Bézier segments*

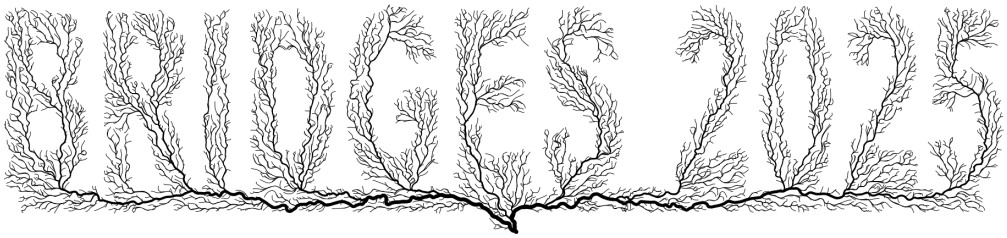
Figure 1: *An illustration of the process used for constructing a figurative tree. A given bitmap (a) is randomly sampled (b) and boundary points are determined (c). A Voronoi diagram is constructed (d) that contains an implicit cell adjacency graph. A spanning tree is then constructed (e) and (f), and a drawn using Bézier curves (g) which can be weighted to show the number of descendant cells.*



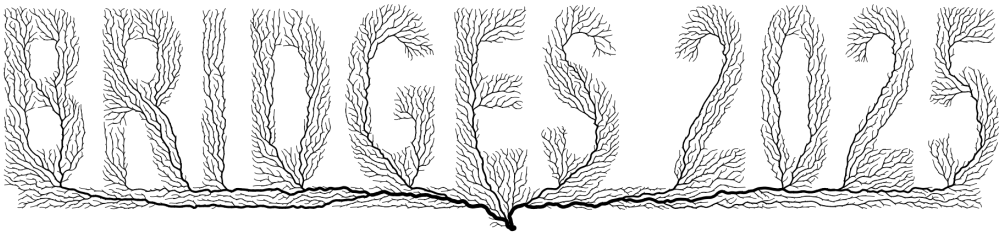
(a) *figurative tree from 1322 random points*



(b) *figurative tree from 1322 points after 5 centroidal iterations*



(c) *figurative tree from 4407 random points*



(d) *figurative tree from 1322 points after 5 centroidal iterations*

Figure 2: *Examples of figurative trees for the text "Bridges 2025" showing how different parameters affect the resulting trees. Trees using random points — (a) and (c) — exhibit ragged appearance compared with the trees after point centroidal iteration — (b) and (d). The number of sampled points increases the fullness of the final tree. The original text letters are disjoint and a thin baseline was added to the bitmap to form a single connected region, perhaps best seen in (d).*

The line segments can be replaced with cubic Bézier curve segments [7] to form a more flowing structure as shown in Figure 1(g). For each segment, the Bézier curve's endpoints connect the input and output boundary edges. The interior Bézier control points are at a slight offset perpendicular to the boundary edges. This results in a path that is continuous and has a continuous derivative across each boundary.

The number of descendants of a node was used to control the thickness of a segment going through a particular cell, resulting in a the tree in Figure 1(h). In both of these figures, an additional curve within each cell was added to provide a fuller structure than in Figure 1(f).

Examples

In addition to the example in Figure 1 of a rectangle with an interior elliptical cavity, this process was also used on a bitmap containing the text "Bridges 2025" as seen in Figure 2; it illustrates the level of complexity possible. Because the original text letters were disjoint, a thin baseline was added to the bitmap to form a single connected region. The effects of changing the control points and centroidal iteration are clearly visible and give a sense of the type of figurative trees that can be created with the process described above.

Discussion

The process described in this paper allows one to create a figurative tree bounded by a region defined by a bitmap. While the region needs to be connected, it does not need to be convex and it can contain holes. For disconnected bitmaps, the process can be run on each connection region separately, and the resulting disconnected trees can be assembled to create the final result. Unlike previous techniques [5, 2] that place points in the entire image, the process described in this paper only places points in a region defined by a bitmap, and uses the bitmap's boundaries to help constrain cell geometry.

The number of points used controls the density of the final tree as seen in Figure 2. In Figure 2(a) and Figure 2(b), 1322 points were used (based on a percentage of the points in the bitmap). Whereas Figure 2(c) and Figure 2(d) each use 4407 points. The number of points influences the point spacing, which in turn influences the number of boundary points needed. Too few boundary points can result in regions that extend beyond the bitmap. Too many boundary points can lead to connections between boundary points on opposite sides of a relatively thin region, rather than sampled points; this can create disconnections in the final graph and thus no spanning tree is possible. The author has relied on trial and error to determine the number of boundary points needed for a given bitmap and the number of sampled points.

The effects of the iterative centroidal Voronoi process can be seen by comparing the tree in Figure 2(a) with the tree in Figure 2(b) and the tree in Figure 2(c) with the tree in Figure 2(d), demonstrating this can be used to control the final tree's raggedness. The starting points were selected to be near the boundary in the given examples, but could have been selected anywhere in the bitmap. Thickening the edges based on distance from a source node and the use of Bézier curve segments are other aspects that distinguish this and previous techniques. While not shown in this paper, leaves or flowers can be added to the terminal ends of the trees for additional visual interest.

References

[1] Lloyd's algorithm. https://en.wikipedia.org/wiki/Lloyd's_algorithm.

[2] R. Bosch, T. Chartier, and M. Rowan. "Minimalist approaches to figurative maze design." *The Mathematics of Various Entertaining Subjects: Research in Recreational Math*. J. Beineke, J. Rosenhouse, and R. Smullyan, Eds. Princeton University Press, 2019.

[3] L. da Vinci. *The Notebooks of Leonardo da Vinci*. J. P. Richter, Ed. Dover Publications, 2012.

[4] L. Goodman, A. Lauschke, and E. W. Weisstein. Dijkstra's Algorithm. From MathWorld–A Wolfram Web Resource. https://mathworld.wolfram.com/DijkstrasAlgorithm.html.

[5] K. Inoue and K. Urahama. "Halftoning with minimum spanning trees and its application to maze-like images." *Computers & Graphics*, vol. 33, no. 5, 2009, pp. 638–647.

[6] Phaidon Editors and T. Kirkham. *Tree: Exploring the Arboreal World*. Phaidon Press Limited, 2024.

[7] Pomax. A Primer on Bézier Curves. https://pomax.github.io/bezierinfo/.

[8] E. W. Weisstein. Voronoi Diagram. From MathWorld–A Wolfram Web Resource. https://mathworld.wolfram.com/VoronoiDiagram.html.