CRUST

David Skocic

December 19, 2025

MTH 496H - Honors Senior Project

Fall Semester 2025

Advisor: Dr. Gregory Lupton

Cleveland State University

Department of Mathematics & Statistics

Contents

Abstract

...

1 Introduction and Background

Describe the analysis of algorithms

Define curve reconstruction

give examples why connecting points to the nearest neighbor isn't good

Maybe define "general position"

The crust method of curve reconstruction builds on a few basic geometric tools that are used constantly in discrete geometry in a wide degree of applications. The following sections introduce these tools and prove numerous important and interesting results about them.

2 Convex Hulls

The first geometric tool of interest is the convex hull. A convex hull is used to take a set of points S and connect them to make a convex shape that contains every point in S. Here, containing a point means that it is either be inside the shape or on its edge. It is easy to imagine that there are all sorts of ways to contain every point in S with some sort of convex polygon. You could, for example, draw a square so large that it contains every point in S, but a shape unrelated to how the points are arranged in S provides almost no more utility than just knowing S. A convex hull fixes this problem, and adheres to the following definition:

Definition 1 (Convex Hull)**.** *For a given set of points S, the convex hull is the intersection of every convex shape that contains every point in S.*

Imagining such a shape for a given set of points is easy. To generate the convex hull for a set of points, imagine the points are nails in a wall. Start by tying a string to a nail at some extrema, say, the leftmost nail on the wall, and holding the string left of that. Then, rotate it all the way around all the other nails until it touches the first one again. This shape the string takes on is the convex hull.

The convex hull is a convenient tool because it has an intuitive definition and because convex shapes tend to be simple to work with. Also, it can serve as a starting point in an algorithm that starts with an unlabeled set of points like the crust. Taking the convex hull at least gives us a place to start connecting our points, even if its usefulness may not initially be clear.

Fortunately, lots of time has been spent finding optimal algorithms for computing the convex hull of a set of points. Computing the convex hull really means to find which points from S are vertices of the hull. This can be done in a variety of ways. One being a sort of inductive method that starts with 3 points, then add one at a time from the set, removing points that end up making the shape nonconvex. Another algorithm

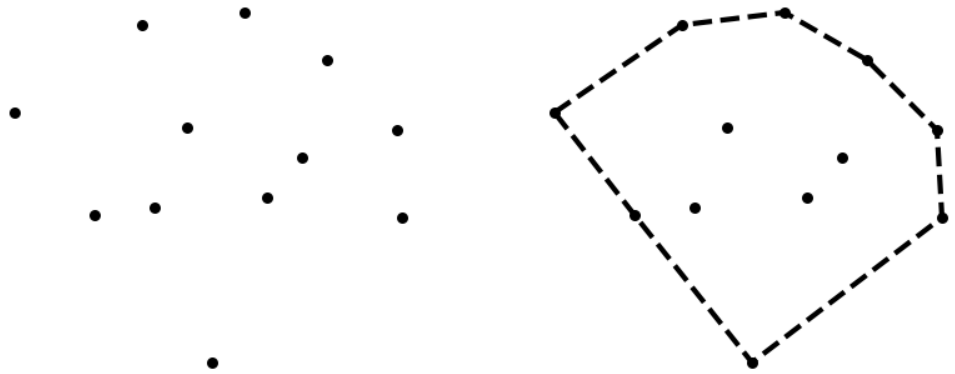


Figure 1: A set of points next to their convex hull

behaves very similarly to the string wrapping algorithm. While these algorithms are both intuitive, there is another class of algorithms that take less time to compute.

There are numerous algorithms that belong to this optimal class, all with time complexity $O(n \log n)$. From these I will focus on one which uses a divide and conquer approach. I choose this method because of its satisfying recursive nature, and because it is the algorithm that most easily extends to a third dimension. My implementation of this algorithm is largely inspired by [Devadoss and O'Rourke, 2011]

The pseudocode for this algorithm is shown in Algorithm 1. But it begins by taking in a list of coordinates sorted by their x-value. Since the algorithm is recursive it must have a base case, and here that base case is if there are three or fewer points in S because a triangle is already a convex polygon. When there are more than three points in S, separate the points into the left half and the right half and generate the convex hull of those points. This splitting continues until the hulls are triangles or line segments, which is the base case. The magic of the algorithm comes when bringing the groups back together. To do this there are two convex hulls, one on the right and one on the left. To combine them, a tangent line is found on the top and bottom (need a figure for this). *continue this description*

3 Triangulations

Triangulations are a key tool in the process of curve reconstruction. Specifically, a type of triangulation called a Delaunay triangulation is used to construct the crust of a given set of points. Before diving into Delaunay triangulations and some interesting results that follow, we first explore an easier problem, the

Algorithm 1 Convex Hull

Require: S, a list of points sorted by their x-coordinate
  function ConvexHull(S)
    if $|S| \leq 4$ **then**
      return S
    else
      $L \leftarrow$ ConvexHull($S[0 \dots \lfloor \frac{|S|}{2} \rfloor]$)
      $R \leftarrow$ ConvexHull($S[\lfloor \frac{|S|}{2} \rfloor \dots |S| - 1]$)
      return Combine(L, R)
    end if
  end function
  function Combine(L,R)
    Fill this in
  end function

triangulation of a polygon, and build up to Delaunay triangulations from there.

3.1 Triangulations of Polygons

To begin, we must first define a triangulation of a polygon.

Definition 2. ***Triangulation of a Polygon in two dimensions*** *A decomposition of a polygon into triangles by a maximal set of non-intersecting diagonals.*

In simpler terms, a triangulation breaks a polygon into triangular pieces by drawing segments between its vertices so that no two segments intersect. The term 'maximal set' is used in the definition to ensure that there is no vertex of the polygon on the edge of any triangle.

As an example, let's look at an easy case: a convex octagon. It is easy to pick out a way to triangulate this shape: just start at one vertex and draw lines to every other vertex as shown in Figure 2. It is easy to see how this technique can extend to any convex n-gon to create a triangulation with n-2 triangles. Something of note is that there can be more than one way to triangulate a polygon, with another example shown in Figure 2. Is it possible that some triangulations are better than others? The fact that there is a section dedicated to a specific type, the Delaunay triangulation, should be a clue here, and the reason why is explored in that section.

A more interesting problem arises when you try to triangulate a shape that is nonconvex. Is this even possible for *any* polygon? If it is, is there an algorithmic way that works to triangulate any polygon? Will there always be the same number of triangles? The following theorems dive deeper into these questions.

Theorem 1. *All polygons can be triangulated*

Proof. This is an inductive proof with the base case being a polygon with 3 vertices. This base case is trivial because a triangle need not be broken down into more triangular pieces. Now, the assumption that any

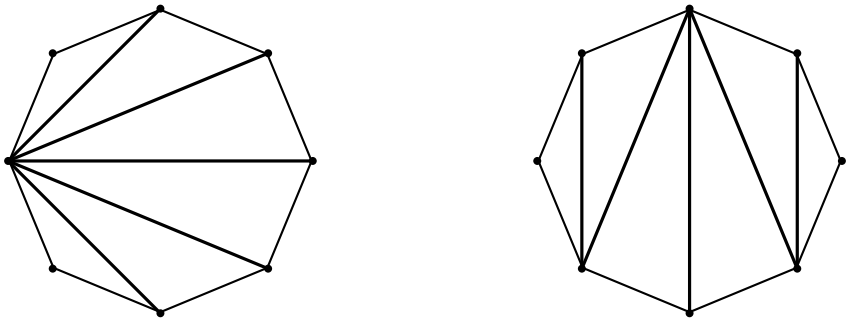


Figure 2: Two example triangulations of a regular octagon.

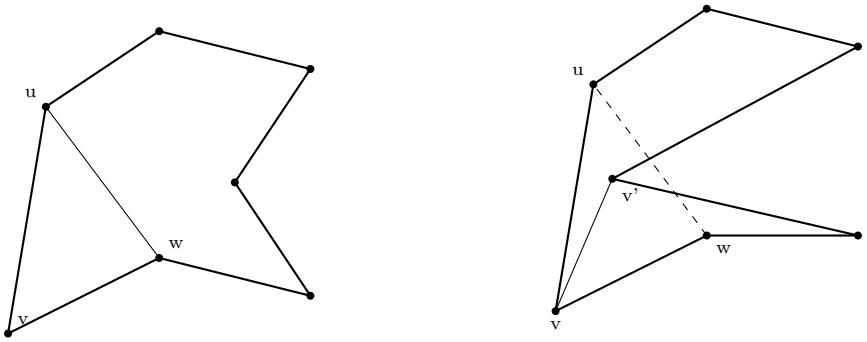


Figure 3: The two cases when trying to draw a diagonal between two points around a starting point.

polygon with m or fewer vertices can be triangulated and our goal is to show that that a polygon with n vertices with $n = m + 1 > 3$ can be triangulated.

I will begin by showing that a diagonal exists in the polygon, P, formed by n vertices. Take some vertex v from P and label the vertices neighboring it u and w. If u and w are connected and do not intersect with any edges of P, then a diagonal exists. Otherwise, there exists one ore more vertices of P that are inside triangle uvw. Now, imagine a line parallel to segment uw that goes through v. Let this line sweep toward segment uw until it encounters a vertex of P inside triangle uvw and call this vertex v'. The segment vv' must be a diagonal because there are no edges of P between v and v' by the definition of v'. So, a diagonal must exist in P.

The existence of this diagonal means that P can be broken into two polygons that share the diagonal as a common edge, with a total of $n + 2$ vertices. There are $n + 2$ because the n outer edges of the polygon remain part of either new polygon, and the diagonal contributes one edge to each of the newly created polygon, adding two edges. Since each new polygon must have at least 3 vertices, the other must have at most $n - 1 = m$ vertices. By the inductive hypothesis, each of these new polygons may be triangulated.

□

In the process of proving that it is always possible to triangulate any polygon, we have also generated

a simple algorithm to triangulate any polygon. If we begin the same way as the proof, drawing a diagonal between some pair of vertices, two smaller shapes are created. Each of these shapes can undergo the same process and create two more shapes. This can be repeated recursively until all the shapes created by the splits are just triangles, so all the diagonals drawn must constitute a triangulation.

3.2 Triangulations of a Set of Points

In the previous section, I showed that any polygon can be triangulated. The method to do this started by drawing a single new edge to form a triangle using two preexisting edges. But what if there are no edges at all, and you just have a set of points? You could construct any number of polygons for a given set of points. To make this algorithmic, we must choose a specific one of these polygons to start to triangulate. To do this, we return to the familiar convex hull.

Definition 3. ***Triangulation of a set of Points*** *FILL OUT THIS DEFINITION*

After computing the convex hull, our situation appears almost like it did in the previous section, but there are still points inside that are not accounted for. To continue down this path, we sincerely hope that the following theorem is true:

Theorem 2. *Any set of points S has a triangulation, T, such that Hull(S)$\subset T$. — notation might not be quite right here, we want the edges of the hull to be a part of the edges of T*

Proof. We follow a similar method to the previous proof, using induction. Again, the base case is a set of 3 points, with the obvious triangulation of a triangle, which is also the convex hull of the points. Our hypothesis is that, assuming that a set of points S of size n has triangulation T with *Hull(S)*$\subset T$, some $S' \supset S$ of size $n+1$ has triangulation T' with *Hull(S')*$\subset T'$.

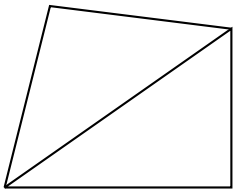
There are two cases for the new vertex v that belongs to S' but not S. Either v is inside *Hull(S)*, or it is outside of it. We will examine each case independently:

Explain that a point outside will have a new hull with two "new edges", and if these two edges are added to the previous triangulation, we still have a triangulation.

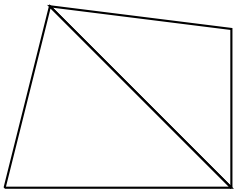
Explain how a point inside the hull is in a triangle, and hence this triangle can just have each vertex connect to the point inside and maintain a triangulation

...

□



(a) Non-Delaunay Triangulation



(b) Delaunay Triangulation

3.3 Delaunay Triangulations

Now that we know that we can generate some triangulation for any set of points, we can begin to look at a certain type of triangulation, the Delaunay Triangulation. The Delaunay Triangulation of a set of points has interesting properties and is the primary tool in generating the crust of a given set of points. But before we explore those properties, let us first define it.

Definition 4. *A **Delaunay Triangulation** is the triangulation of a set of points S such that the circumcircle of each triangle only contains the vertices of that triangle, and no other points in S*

Something you may have noticed in the definition of a Delaunay triangulation is that I claimed that it is *the* triangulation that fits the criteria, not just some triangulation that does. This uniqueness is an important property of Delaunay Triangulations, but it is not at all apparent. Let us show why this is true:

SHOW THAT THE EXISTENCE IS ALSO GUARANTEED

Theorem 3. *There is only one Delaunay Triangulation for given set of points S.*

Proof. □

It is of note that if there are four points that are concentric, then there cannot be a triangulation that fits the Delaunay criteria. So, the four points have, in some sense, two valid Delaunay triangulations. We call these triangulations degenerate.

Another interesting property of Delaunay triangulations is that they guarantee that the smallest internal angle of any triangle in the shape is the highest out of all possible triangulations.

Theorem 4. *The Delaunay Triangulation has the highest minimum internal angle out of all possible triangulations*

Proof. □

Effectively, Delaunay triangulations reduce "sliver triangles" which are very long, thin triangles.

talk about angle sequences and transition that into "flipping"

show how you can get from one triangulation to the Delaunay triangulation using progressive steps

dive into talking about the 'flip graph'

show the connection to voronoi diagrams

4 Voronoi Diagrams

5 The Crust

It is finally time to put the geometric tools that this paper has focused on so heavily to use to define the crust of a set of points. As a reminder, the crust is a method of curve reconstruction introduced by [Amenta et al., 1998], which is where all the definitions come from in this section. It begins with a set of points taken from a smooth curve with the assumption that they are close enough together to be reconstruction. This "close enough" condition is defined later in the section on sampling. The crust has the following simple definition:

Definition 5 (The Crust)**.** *Let S be a set of finite points in the plane, and let V be the vertices of the Voronoi diagram of S. Next let $S' = S \cup V$ and consider the Delaunay Triangulation of S'. Any edge of the Delaunay triangulation of S' that connects two points in S is an edge of the crust.*

INSERT A FIGURE IN THIS SECTION WITH AN EXAMPLE SIMILAR TO PAGE 3 OF THE PAPER

This is a fairly simple definition, and yet I claim that it can often successfully reconstruct a curve from a set of points in the plane. Before formally proving that this is true for a certain sampling condition, I will provide some intuition on why this is the case. To do this, I will give an alternate but equivalent definition of the crust that uses the familiar empty circle property:

Definition 6 (Alternate Definition of The Crust)**.** *Let S be a set of finite points in the plane, and let V be the vertices of the Voronoi diagram of S. An edge connecting two points in S belongs to the crust of S if there is a disk containing no other points in $S \cup V$.*

It also helps to note that Voronoi vertices are at the corners of regions of proximity, so they are closer to the points from S in adjacent Voronoi regions than to any other points in S. This means that points that

obey the alternative definition are connected with those in nearby proximity regions, not simply points that are closest together.

Also, looking at a few examples of Voronoi diagrams of many points, one can see that the diagram draws a line approximately through the middle of the shape that they create. This line is called the medial axis and will be explained further in the next section, but the idea is that the Voronoi edges trace out a line through the middle of a shape, then the empty circle properly serves to prevent lines from being drawn through this center line of the shape.

6 The Medial Axis

This section begins to lay the foundation to prove that the crust is a valid method to reconstruct a curve, with a few interesting results along the way. First, I will define the medial axis, but intuitively it can be thought of as a Voronoi diagram for an infinite set of points [Amenta et al., 1998].

Definition 7 (The Medial Axis)**.** *The medial axis of a curve F is the set of all points in the plane which have two or more closest points in F.*

INCLUDE EXAMPLE FIGURE OF THE MEDIAL AXIS

7 Sampling

7.1 Local Feature Size

8 Examples

References

[Amenta et al., 1998] Amenta, N., Bern, M., and Eppstein, D. (1998). The crust and the -skeleton: Combinatorial curve reconstruction. *Graphical Models and Image Processing*, 60(2):125–135.

[Devadoss and O'Rourke, 2011] Devadoss, S. L. and O'Rourke, J. (2011). *Discrete and computational geometry*. Princeton University Press, Princeton, NJ.