SHORTEST DIAGONAL TRIANGULATION OF CONVEX LAYERS

Anders Hast
Department of Information technology
Uppsala University
Uppsala, Sweden
e-mail: Anders.Hast@it.uu.se

Peter Jenke
Faculty of Engineering and Sustainable Development
University of Gävle
Gävle Sweden
e-mail: Peter.Jenke@hig.se

Stefan Seipel
Department of Information technology
Uppsala University,
Faculty of Engineering and Sustainable Development
University of Gävle
e-mail: Stefan.Seipel@{it.uu.se, hig.se}

ABSTRACT
One problem in the field of computational geometry is the triangulation of convex layers. The rotating caliper algorithm is an alternative to the constrained Delaunay triangulation method. We present an improved triangulation algorithm, which gives a mesh quality close to that of the Constrained Delaunay but substantially faster. Each layer will be connected to the neighboring layer by edges and from the two vertices constituting an edge the proposed algorithm will select the shortest diagonal to its next neighbors in the polygonal chain on the other side, i.e. from the outer layer to the inner layer or vice versa. We discuss quality issues regarding the rotating caliper method and some improvements to it, as well as how a Constrained Delaunay can be efficiently implemented for convex layers.

KEY WORDS
Computational Geometry, Triangulation, Convex Layers, Rotating Caliper, Constrained Delaunay, Shortest Diagonal.

1 Introduction

The convex hull is a fundamental concept in computational geometry [17, 19], with a large number of applications [1, 8, 16]. A method often referred to as onion peeling [10, 3], is used to construct what is called convex layers [7] from a set of points, so that each convex hull contains a set of convex hulls that do not intersect each other. The convex layers of a planar point set are constructed by incrementally computing the convex hull of the set. In each iteration points in the hull are removed from the set and the hull is computed for the remaining points in the set. This process is repeated until the set is empty.

Convex layers find their applications in different fields like map labeling [3] and point cloud triangulation [13], just to mention a few. The computational geometry problem of triangulating convex layers is often solved by a method known as Rotating Calipers [22, 18], even if there is nothing that prevents the use of a Constrained Delaunay triangulation [20].

One useful property for a triangulation of convex layers is that a pair of neighboring hulls encloses a triangle strip [2]. Such triangle strips minimizes vertex data flow in the rendering pipeline. There exist a number of methods for constructing triangle strips for general meshes [12] or point sets [2, 5].

We present a new approach to the triangulation of convex layers, the Shortest Diagonal method, which is easy to implement and aimed at finding the shortest diagonal from one hull to other giving well behaved triangle strips. This method will be compared to the Rotating Caliper method and the Constrained Delaunay method. We also show some improvements to the Rotating Caliper method as well as how a Constrained Delaunay can be implemented efficiently for convex layers.

2 Convex Layer Triangulation

The convex layers are created by an appropriate convex hull algorithm [7], where each layer is removed from the set and the process is repeated until there are no more points left in the set. However, the innermost layer can consist of a hull with arbitrary size depending on the size of the set or even two points forming a line or just one single point. Each layer is a sequence of points or polygonal chain. The triangulation algorithm triangulates between an outer and an inner layer, which together form an annulus.

Let the outer layer be a polygonal chain or a set \( A \) with the points \( \{a_1, a_2...a_n\} \in A \) and the inner layer the set \( B \) with the points \( \{b_1, b_2...b_n\} \in B \), each connected with an edge.

2.1 The Rotating Caliper Algorithm

The Rotating Caliper algorithm can be applied to many problems [18, 22] in computational geometry. When dealing with the triangulation of convex layers [23] the algorithm starts by finding the leftmost points \( a_k \in A \) and \( b_m \in B \) in each layer and connects them with an edge. Two parallel lines called calipers are placed in these points as shown in figure 1. Since the leftmost points are chosen it is safe to create vertical calipers as they will have all other points in that layer on the right side of them or just on the caliper. These calipers are rotated clockwise until either \( a_{k+1} \) or \( b_{m+1} \) is hit. An edge is connected between the...
point \((a_1)\) being hit by the caliper in the outer layer and the base point \(b_0\) of the caliper in the inner layer. This process is repeated until the whole annulus is triangulated.

![Figure 1](image1.png)

Figure 1. The setup for the Rotating Calipers algorithm. The leftmost points are chosen as a start yielding undesirable triangles. Better would be to choose the closest point \(b_7\) in the inner layer.

In practice the algorithm does not rotate the calipers, instead the angles \(\mu_0\) and \(\mu_1\) can be used in order to determine which point is going to be hit first.

The algorithm can be improved in at least two ways. First of all it is obvious that a better triangulation of the case in figure 1 would be to connect \(a_0\) to \(b_7\), which is its closes point. This cannot be done if the calipers are not first rotated accordingly. However, the computation of the angles can be removed and a simple area test \([17]\) will tell which point to choose as the next one to connect. In figure 2 two vectors \(a_1 - a_0\) and \(b_7 - b_0\) are created from the four points in question. Now a simple area test can be done following the same order and the general test becomes:

\[
\tau = \text{area}((0, 0), b_{m+1} - b_m, a_{k+1} - a_k)
\]

(1)

The sign of the test will tell which point to choose. This test is not depending on where the start point is located and therefore it is safe to choose the closest point in the inner layer to create the first edge in the triangulation.

The main part of the algorithm is described in Algorithm 1. There are couple of details in this algorithm omitted here for clarity. First of all the special case when there is only one vertex in the centre layer. Then it is more efficient to iterate through all points in the neighboring outer layer and create triangles from each outer point to the single inner point. Furthermore, when the algorithm reaches the final point in the outer or inner layer there is only one way to connect it to the opposite layer, if not that point is already connected and no area test is therefore necessary. Hence, the algorithm will iterate one step less and check if the only possible connection needs to be done or not after the main loop.

![Figure 2](image2.png)

Figure 2. The edges in figure 1 have been converted to a pair of vectors that span a triangle. A simple area test can determine the mutual orientation of the vectors and hence which one will be hit first by the "calipers".

Algorithm 1: The Rotating Caliper algorithm.

```
i=0;
while k < length(A) and m < length(B) do
    \(\tau = \text{area}((0, 0), b_{m+1} - b_m, a_{k+1} - a_k)\);
    if \(\tau > 0\) then
        triangle(i)=(a_k, b_m, a_{k+1});
        k = k + 1;
    else if \(\tau < 0\) then
        triangle(i)=(b_m, b_{m+1}, a_k);
        m = m + 1;
    else
        triangle(i)=(a_k, b_{m+1}, a_{k+1});
        triangle(i)=(b_m, b_{m+1}, a_k);
        k = k + 1;
        m = m + 1;
    end
end
```

3 Delaunay Triangulation

Delaunay triangulation \([21]\) is very popular due to its desirable properties \([15]\) and has many areas of application such as in geoscience \([14]\) and surface reconstruction \([6]\). The main idea is that a triplet of points can be used to construct a circle. If the resulting circumcircle contains no other points in the set, then the three points are used as vertices for the new triangle. This is known as the empty circle criterion \([4]\). If the set of points has one or more predefined edges, then some kind of Constrained Delaunay \([20, 9]\) method must be used for the triangulation so that these edges override the empty circle criterion. When triangulating the convex layers the Constrained Delaunay becomes rather simple as there are only two possible points to choose from, namely the next points in each polygonal chain. The previous edge is drawn between \(a_k\) and \(b_m\), and the next edge will be drawn from either of those to points \(a_k\) and \(b_m\) or \(b_{k+1}\), depending of the empty circle criterion. Moreover,
Algorithm 2 tests whether point \( b_{m+1} \) is considered to be inside the circle spanned by the points \((a_k, b_m, a_{k+1})\), which is the case to the right in figure 3. Then the point \( a_{k+1} \) shall not be connected. This either implies that the circle \((a_k, b_m, b_{m+1})\) is empty and the point \( b_{m+1} \) shall be connected instead, or that we have the special case that both points shall be connected but this will be handled in the next iteration.

4 Shortest Diagonal Triangulation

In this section the new triangulation algorithm is presented. The idea is to start with a point \( a_k \) in hull \( A \) and then finding the nearest (i.e. having minimal Euclidean distance) visible point \( b_m \) (i.e. not hidden by an edge) in hull \( B \). These points are connected with a triangle edge. Then the algorithm proceeds, as shown in figure 4 by determining which point \( a_{k+1} \) or \( b_{m+1} \) having the closest visible point in the opposite hull. This pair of points \((a_k, b_{m+1})\) or \((b_m, a_{k+1})\) is then connected depending on which of them has the shortest distance. As the edge always proceeds forward in the polygonal chain from one hull to the other, the algorithm is called the Shortest Diagonal Triangulation.

There are two special cases that must be handled in order to not cross the inner layer \( B \) with a new edge. In fact the cases presented here are also applied in the same way for the Constrained Delaunay presented later in section 3. However, the Rotating Caliper method has the advantage of not having any such cases due to the way the algorithm proceeds. What this algorithm gains in speed it looses in triangle mesh quality [11] as explained in section 5.

Figures 5 and 6 shows the two special cases when the shortest edge (dotted) cannot be chosen as it crosses the inner layer and the longer (red) edge must be chosen instead. Case A can be detected by the sign of the area of the triangle: \((a_k, b_m, b_{m+1})\), which gets an opposite sign compared to the general case.

The first case (A) in figure 5 can be detected by an area test of the triangle \((b_m, a_k, b_{m+1})\). This can be compared to the situation at the left of figure 4 where the triangle has the opposite orientation. The second case (B) in figure 6 can be detected by computing the double area of
A method will do, but Rotating Caliper cannot be connected as it to the inner layer must be chosen.

Figure 5. Case A: The shortest (dotted) edge from the outer layer \( a_k \) to the inner layer \( b_{m+1} \) cannot be connected as it crosses the inner layer and therefore the longer edge (red) from the inner layer \( b_m \) to the outer \( a_{k+1} \) must be chosen.

Figure 6. Case B: The shortest edge (dotted) from the inner layer \( b_m \) to the outer layer \( a_{k+1} \) cannot be connected as it crosses the inner layer and therefore the longer edge (red) from the outer layer \( a_k \) to the inner \( b_{m+1} \) must be chosen.

the triangle \( (b_m, a_{k+1}, b_{m+1}) \), which has an opposite sign compared to the situation at the right in figure 4.

5 Discussion

One example of running the three different algorithms is shown in figure 7. One can note that the Rotating Caliper algorithm generally tends to produce more skinny triangles than the other two methods. The Constrained Delaunay and the Shortest Edge produce triangles that generally are more similar but some are different and it seems obvious that the Constrained Delaunay does not always choose the shortest edge due to the empty circle criterion.

Timings are shown in the top in figure 8, which clearly show that the empty circle criterion makes the Constrained Delaunay algorithm substantially slower compared to the others that have a much simpler test. The linear scaling is due to the fact that all three compared algorithms have \( O(n) \) complexity. Moreover, the two diagrams below show the mesh quality expressed as the ratios of shortest/longest edge (middle) and smallest/largest angle (bottom) for all triangles in the layers. All tests where done 10000 times for each method on the same point set and all programs were written in matlab\textsuperscript{®}. The size of the point sets varies from 4 up to 87 points. The point locations were randomly picked and convex layers were constructed from them. All three algorithms were run on the same layers in each test.

Figure 8 (middle) and (bottom) show different quality metrics for the resulting triangulations in terms of averaged triangle asymmetries for increasing number of points in the point sets, where larger values express more well-behaved non-skinny triangles. For larger point sets, the quality metrics approaches a steady. The salient peaks in Figure 8 (middle) and (bottom) suggest better overall triangulation for all methods for point sets with only few points, which requires some clarification. One explanation for this is the way in which convex layers are constructed in an Onion Peeling manner. For small point sets this results in only a couple of layers, where the average distance between layers is comparably large and almost equal to the average distance between consecutive points in a layer, hence leading to less skinny triangles. For large point sets we obtain rather closely spaced layers with relatively long distances between consecutive points in the layer. Figure 9 illustrates this for two datasets where only layers are shown. The most interesting observation is, however, that overall triangulation quality for both metrics is clearly better for Constrained Delaunay triangulation and Shortest Diagonal Triangulation, whereas the Rotating Caliper tends on average to produce more skinny triangles.

6 Conclusion and Future Work

The Rotating Caliper algorithm triangulates convex layers faster than a Constrained Delaunay method will do, but with lesser mesh quality. Both methods were implemented with some improvements in order to be fast while producing as high quality triangles as possible. Two mesh quality measures were used: the ratio of shortest/longest edge

\begin{algorithm}
\begin{algorithmic}
\State i=0;
\While {k < length(A) and m < length(B)}
\State \( d_1=\text{distance}(a_k, b_{m+1}) \);
\State \( d_2=\text{distance}(b_m, a_{k+1}) \);
\If {\( d_1 \leq d_2 \)}
\If {not case A}
\State \( \text{triangle}(i)=(b_m, b_{m+1}, a_k) \);
\State \( m=m+1 \);
\Else
\State \( \text{triangle}(i)=(a_k, b_m, a_{k+1}) \);
\State \( k=k+1 \);
\EndIf
\Else\EndIf
\If {not case B}
\State \( \text{triangle}(i)=(a_k, b_m, a_{k+1}) \);
\State \( k=k+1 \);
\Else\EndIf
\State \( \text{triangle}(i)=(b_m, b_{m+1}, a_k) \);
\State \( m=m+1 \);
\EndIf
\EndWhile
\end{algorithmic}
\caption{The Shortest Diagonal algorithm.}
\end{algorithm}

i=0;
while \( k < \text{length}(A) \) and \( m < \text{length}(B) \) do
\( d_1=\text{distance}(a_k, b_{m+1}); \)
\( d_2=\text{distance}(b_m, a_{k+1}); \)
if \( d_1 \leq d_2 \) then
if not case A then
\( \text{triangle}(i)=(b_m, b_{m+1}, a_k); \)
\( m=m+1; \)
else
\( \text{triangle}(i)=(a_k, b_m, a_{k+1}); \)
\( k=k+1; \)
end
else
if not case B then
\( \text{triangle}(i)=(a_k, b_m, a_{k+1}); \)
\( k=k+1; \)
else
\( \text{triangle}(i)=(b_m, b_{m+1}, a_k); \)
\( m=m+1; \)
end
end
end
and the ratio of smallest/largest angle in the triangles. Ten thousand tests were conducted for each method and point set, varying the point set from 4 points up to 87 points, that were randomly picked. The proposed algorithm: Shortest Diagonal performed almost as fast as the Rotating Caliper with a mesh quality comparable to that of the Constrained Delaunay method. For future work we propose to compare the three algorithms also for convex spirals.

Figure 7. Top: Rotating Calipers, middle: Constrained Delaunay and below: Shortest Diagonal.

Figure 8. Timings and mesh quality measurements of the the three methods. Constrained Delaunay takes more time due to the empty circle test. The quality of the Shortest Diagonal method is comparable to Constrained Delaunay.

References

Figure 9. Convex layers for differently dense point sets. Top: 15 points. Bottom: 90 points.


