Polygon Partitioning

Vidit Goel, Manad Mishra
Department of Electrical Engineering, Department of Computer Science
Indian Institute of Technology, Kharagpur - 721302, India

Abstract

In this paper we study the problem of polygon partitioning. Polygon partition problem is a problem of finding a partition which is minimal in some sense, for example a partition with a smallest number of units or with units of smallest total side-length. We particularly deal with the partition of polygon into convex polygon. Our aim is to find an optimal algorithm in a sense that divides the general polygon into an optimal number of convex polygons while also keeping the computational complexity reasonable. We go through various algorithms and analyse performance of two major algorithms for solving convex partitioning.

Introduction

Polygon partitioning is highly useful in computational geometry. In polygon partitioning we represent a polygon as the union of a number of simpler component parts(Figure 1). Pattern Recognition is one area that uses polygon decomposition as a tool [1]. Pattern recognition techniques extract information from an object in order to describe, identify or classify it. An established strategy for recognising a general polygonal object is to decompose it into simpler components, then identify the components and their interrelationships and use this information to determine the shape of the object [1]. Layouts are represented as polygons, and one approach to preparation for electron-beam lithography is to decompose these polygon regions into fundamental gures. Polygon decomposition is also used in the process of dividing the routing region into channels [3]

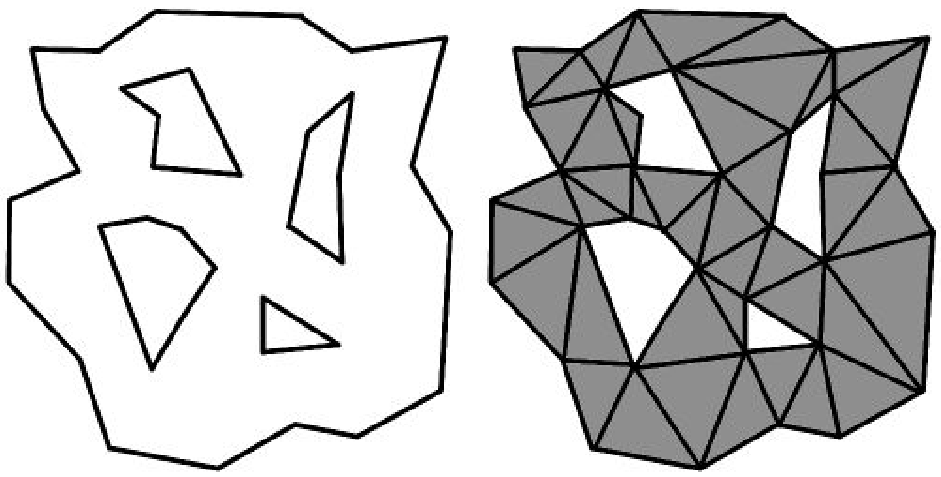


Figure 1 Polygon on the left is input and the one on the right is output.

In particular we deal with **convex partitioning**. As we are all aware that dealing with convex polygons is a lot easier when compared to any general polygons, hence convex partitioning is a well researched topic[2]. There are many algorithms to solve the problem. Some methods focus on producing the optimal number of convex polygons, while some focus on the time complexity. There are also some algorithms which produce a solution optimal in edge length. In this paper we compare various algorithms on their time complexity and optimality of the number of polygons produced. We also study failure cases, for example some algorithms might not work when there is a hole in a polygon whereas some produce very thin polygons or polygons which have very small areas. Polygons with very small areas can be problematic when they are used in pattern recognition to identify or classify objects.

Algorithms for convex partitioning can be divided into two groups. The first group is convex partition by segments. A convex partition by segments of a polygon P is a decomposition of P into convex polygons obtained by introducing arbitrary segments. The second is convex partition by diagonals. A convex partition by diagonals of a polygon P is a decomposition of P into convex polygons obtained by only introducing diagonals. Figure 2 shows both the cases pictorially for better understanding. We will be majorly focusing on the second category that is partition by diagonals. In the next section we will go into details of various algorithms like Triangulation by ear clipping, Convex partition using Hertel-Mehlhorn algorithm, Triangulation by partition into monotone polygons. Later in the paper we discuss the implementation details and compare performance of two algorithms.

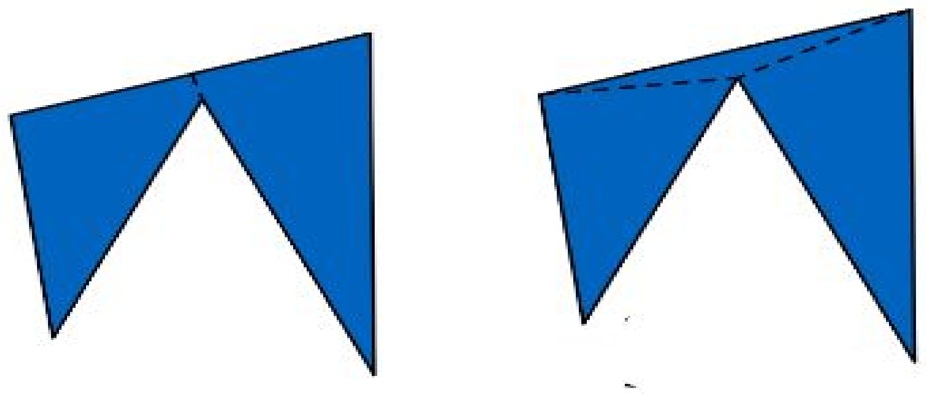


Figure 2 The left figure depicts convex partition by segments and right corresponds to convex partition by diagonals.

Literature Survey

Triangulation, the partitioning of the interior of a polygon into triangles, is a central problem in computational geometry. Many algorithms for polygons begin by triangulating the polygon. As triangles are also convex polygons we first look into different algorithms of triangulation.

One way to triangulate a simple polygon is based on the two ears theorem[4], as the fact that any simple polygon with at least 4 vertices without holes has at least two 'ears', which are triangles with two sides being the edges of the polygon and the third one completely inside it. The algorithm then consists of finding such an ear, removing it from the polygon (which results in a new polygon that still meets the conditions) and repeating until there is only one triangle left. This algorithm is easy to implement, but slower than some other algorithms, and it only works on polygons without holes. An implementation that keeps separate lists of convex and concave vertices will run in *$O(n^2)$* time. This method is known as ear clipping and sometimes ear trimming.

Another approach is Triangulation by partition into monotone polygons. A simple polygonal is monotone with respect to a line L, if any line orthogonal to L intersects the polygon at most twice. A monotone polygon can be split into two monotone chains. A polygon that is monotone with respect to the y-axis is called y-monotone. A monotone polygon with n vertices can be triangulated in *$O(n)$* time. Assuming a given polygon is y-monotone, the greedy algorithm begins by walking on one chain of the polygon from top to bottom while adding diagonals whenever it is possible[5].It is easy to see that the algorithm can be applied to any monotone polygon. If a polygon is not monotone, it can be partitioned into monotone sub polygons in *$O(n \log n)$* time using a sweep-line approach. The algorithm does not require the polygon to be simple, thus it can be applied to polygons with holes. Generally, this algorithm can triangulate a planar subdivision with n vertices in *$O(n \log n)$* time using *$O(n)$* space

An optimal triangulation is one that minimizes some cost function of the triangles. A common cost function is the sum of the lengths of the legs of the triangles, i.e. (where |a, b| is the Euclidean distance from point a to point b). Optimal triangulation can be solved using dynamic programming. The algorithm does not support holes. Time complexity for this algorithm is *$O(n^3)$*.
All the above mentioned algorithms divide the polygon into triangles. Now we will look into algorithms which will divide the polygon into a general convex polygon, not necessarily triangle.

The Hertel-Mehlhorn[6] heuristic for convex decomposition using diagonals is simple and efficient. It starts with an arbitrary triangulation of the polygon and then deletes any chord that leaves only convex pieces. A chord deletion creates a nonconvex piece only if it creates an internal angle that is greater than 180 degrees. The decision of whether such an angle will be created can be made locally from
the chords and edges surrounding the chord, in constant time. The result always contains no more than four times the minimum number of convex pieces. This algorithm also supports holes. Space and time complexity is *$O(n)$* and *$O(n^2)$* respectively.

Algorithm proposed by Keil and Snoeyink produced a minimum number of sub polygons. It is based on dynamic programming with space and time complexity of *$O(n^3)$*. The algorithm was proposed for only simple polygons and does not support holes. Still it is of great applicability as it produces the optimal number of sub polygons.

Proposed Method

We saw various algorithms in the previous section. Many of them resulted in triangles as sub polygons which is highly sub optimal. Algorithm proposed by Keil and Snoeyink produces a minimum number of sub polygons, but it only works on simple polygons. As a solution we propose to use the **Hertel-Mehlhorn**[6] algorithm. The result always contains no more than four times the minimum number of convex pieces with a time complexity of *$O(n^2)$* and space complexity of *$O(n)$.* This algorithm also works for general polygons (polygons which can have holes in them). As a baseline algorithm we also show results of triangulation by ear clipping.

Algorithm:

1. Start with any triangulation of simple polygon P *(time $O(n^2)$)*
2. Remove inessential diagonals, in any order (*time O(n), since we can test a diagonal locally in time O(1) to see if it is essential; if we remove a diagonal, we only have to update the "inessential" flag of O(1) other diagonals*)

Hence, the time complexity is *$O(n^2)$*. Non essential diagonals are the diagonals upon whose removal the resulting vertex will still have a non reflex angle, i.e the resultant polygon will remain convex.

Claim:
This algorithm is never worse than 4 × optimal in the number of convex pieces.

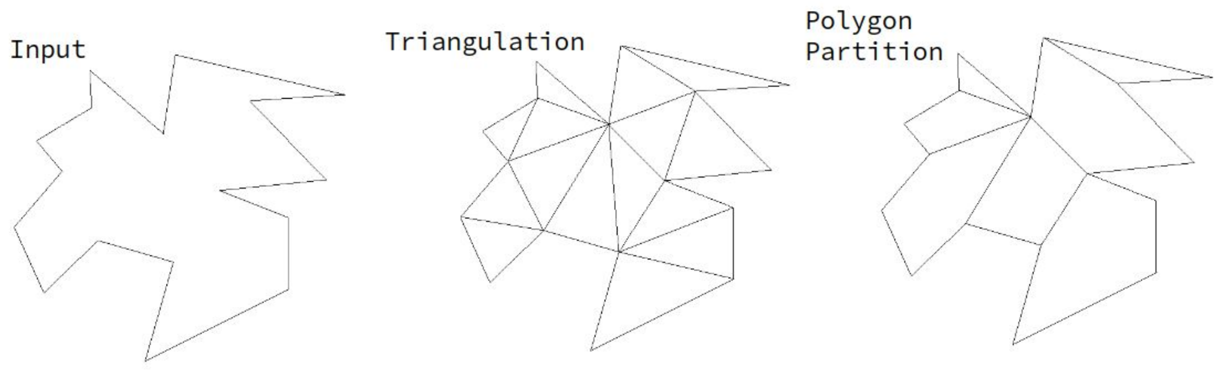
Proof:

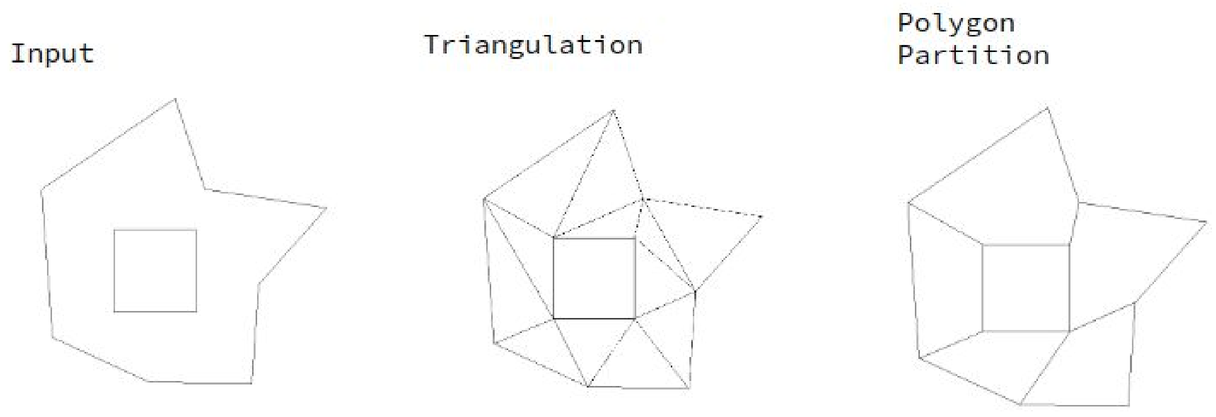
1. When the algorithm terminates, every remaining diagonal is essential for some (reflex) vertex. Each reflex vertex can have at most two essential diagonals.
2. This implies that there can be at most $2r + 1$ pieces in the partition. Since at least $\lceil r/2 \rceil$+1 are required, the result is within 4×optimal. (where r is the number of reflex angles in the polygon)

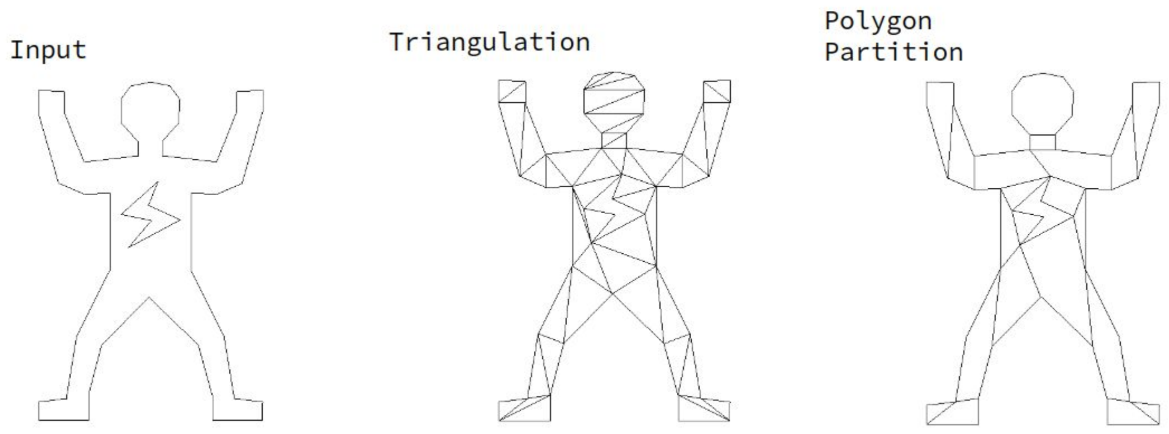
Experiments & Results

We used C++ to implement the Hertel-Mehlhorn Algorithm. This algorithm takes a triangulated polygon as its input. For triangulation we used the Ear Clipping method. We defined a data structure for representing a 2D point along with various operations, and implemented a polygon as an array/list of these points with a flag indicating the presence of a hole. Our current code only supports polygons with at max 1 hole. For visualizing the results we used a pre-existing bitmap image input/output library.
We ran on code for various simple as well as complex polygons with holes.
Our code is uploaded on GitHub and is available at the following link:
https://github.com/manadmishra/Polygon-Partitioning

Here are some of the polygonization results we obtained :







**More results available on our Github Repository*

Conclusion

We observed that the Hertel-Mehlhorn Algorithm divided the input polygons into the optimal number of convex segments in accordance to the number of reflex angles as defined by Chazelle's claim.

Chazelle's Claim:
Assume the polygon P has r reflex vertices. If Φ is the fewest number of polygons required for a convex partition by segments of P then:

$$\lceil r/2 \rceil + 1 \leq \Phi \leq r + 1$$

It was also observed that the algorithm supported large polygons with holes and also optimally partitioned them. The original paper claims that the algorithm can sometimes falter and give upto 4 times the optimal number of polygons, however we did not encounter any such case and observed that the number of convex partitions generated by the algorithm were within the optimal range across all our test cases.

References

[1]H. Y. F. Feng and T. Pavlidis. Decomposition of polygons into simpler components: feature generation for syntactic pattern recognition. IEEE Trans. Comput., C-24:636{650, 1975.
[2] Armaselu, Bogdan, and Ovidiu Dăescu. "Algorithms for fair partitioning of convex polygons." *Theoretical Computer Science* 607 (2015): 351-362.
[3] Struzyna, Markus. Flow-based partitioning and position constraints in VLSI placement. IEEE, 2011.
[4] G.H. Meisters, Polygons have ears, Amer. Math. Monthly, vol. 82, pp. 648-651, 1975
[5] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf (2000), Computational Geometry (2nd revised ed.), Springer-Verlag, ISBN 3-540-65620-0 Chapter 3: Polygon Triangulation: pp.45–61.
[6] Hertel, Stefan, and Kurt Mehlhorn. "Fast triangulation of simple polygons." International Conference on Fundamentals of Computation Theory. Springer, Berlin, Heidelberg, 1983.
[7] Keil, Mark, and Jack Snoeyink. "On the time bound for convex decomposition of simple polygons." *International Journal of Computational Geometry & Applications* 12.03 (2002): 181-192.