Mini Project

On implementing 2D Straight Skeleton with Python

December 2019

Department of Information Technology
Indian Institute of Engineering Science and Technology, Shibpur

Supervisor
Arindam Biswas
Professor
Information Technology
Indian Institute of Engineering Science and Technology, Shibpur

Group Members
Sahil Baranwal 510817011
Debraj Das 510817005
Joyoshish Saha 510817003

Acknowledgement

It is our privilege to express our sincerest regards to our project supervisor, Professor Arindam Biswas, for his valuable inputs, able guidance, encouragement, whole-hearted cooperation and constructive criticism throughout the duration of our project.

We deeply express our sincere thanks to our Head of Department Professor Hafizur Rahman for encouraging and allowing us to present the project on the topic “Determination of 2D Straight Skeleton using Python“.

We take this opportunity to thank all our lecturers who have directly or indirectly helped our project. We pay our respects and love to our parents and all other family members and friends for their love and encouragement throughout our career. Last but not the least we express our thanks to our friends for their cooperation and support.

Signature of group members

Abstract

Skeleton like structure is often used for the description of basic topological characteristics of a 2D object. In the image processing and computer vision fields, skeleton, informally defined as a set of points located in the centers of such circles included in the object, that they touch the object's boundary in at least two distinct points, is used. Such a structure is the well-known Generalized Voronoi Diagram (GVD).

The Straight Skeleton differs, in general, from the GVD. If the given polygon is convex, then both structures are identical. Otherwise, the GVD contains parabolically curved segments in the neighborhood of reflex vertices. Parabolically curved segments are avoided in the Straight skeleton. Both structures reflect the shape of a polygon in a similar manner, however the Straight Skeleton is more sensible to local changes of the given shape, for adding a reflex vertex with very small external angle may change the skeleton structure completely.

We, in this project, implemented 2D straight skeleton of any non-self intersecting polygon with or without holes using Python and several other packages available online. As an application to this, we developed the roof model given some shape of walls. The application is apt enough to produce straight skeleton with 100-gon and above with decreasing accuracy in building the faces of straight skeleton.

Contents

Introduction

Aichholzer et al. introduced the straight skeleton of simple polygons P by considering a so-called wavefront-propagation process. Each edge e of P sends out a wavefront which moves inwards at unit speed and is parallel to e. The wavefront of P can be thought to shrink in a self-parallel manner such that sharp corners at reflex vertices of P remain sharp, see Figure 1. During this propagation process topological changes, so-called events, will occur: Edges collapse to zero length and the wavefront may be split into multiple parts. Each wavefront vertex moves along the bisector of two edges of P and, while it moves, it traces out a straight-line segment.

Definition (straight skeleton, arcs, nodes, faces). The straight skeleton S(P) of the polygon P comprises the straight-line segments that are traced out by the wavefront vertices. These straight-line segments and are called the arcs of S(P). The loci of the topological changes of the wavefront are called nodes. To each edge e of P belongs a face f(e), which consists of all points that are swept by the wavefront edge that is sent out by e.

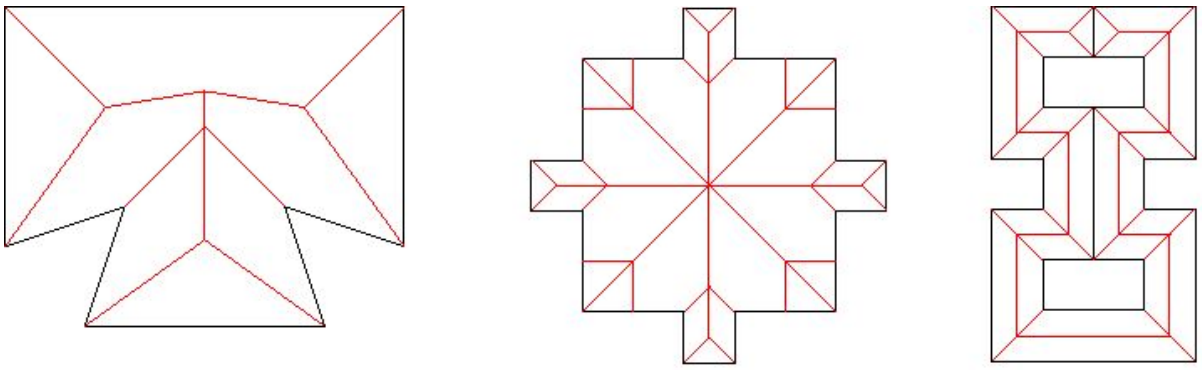


Fig. 1 Simple Straight Skeletons of differently shaped polygons

Objectives and Approach

We implemented the Petr Felkel and Stepan Obdrzalek algorithm to build straight skeleton for polygons with or without holes and that are non-self intersecting. Our objective was to embed this algorithm to build some applet that can create straight skeleton for 2D polygons. The straight skeleton of a polygon is similar to the medial axis and the Voronoi diagram of a polygon in the way it partitions it; however, unlike the medial axis and Voronoi diagram, the bisectors are not equidistant to its defining edges but to the supporting lines of such edges. As a result, Straight Skeleton bisectors might not be located in the center of the polygon and so cannot be regarded as a proper Medial Axis in its geometrical meaning. On the other hand, only reflex vertices (whose internal angle >π) are the source of deviations of the bisectors from its center location.

Our approach was to define data structures like SLAV, faces, edges and vertices to perform the algorithm proposed, with the help of Python and other packages. We can imagine the principle of the algorithm as a

construction of a roof with constant slope from given shape of the walls. It can be done by a sweep algorithm, which simulates cutting of the roof by parallel planes and checks local changes of the polygonal base in the cross sections. In a 2D view, it appears as a shrinking of the polygon. The polygon edges are moving at constant speed inward the polygon and they are changing their lengths. The polygon vertices move along the angular bisectors as long as the polygon changes its topology.

Theory

Voronoi Diagram: Partitioning of a plane with n sites into convex polygonal cells such that each cell contains exactly one site and every point inside this cell is closer to its site than any other.

The Voronoi region Vor(p) of a site p in S is
Vor(p) = {x ∈ R2 | // x − p // ≤ // x − q // for all sites q in S},
where // p − q // denotes the (Euclidean) distance between the points p and q in the plane. Points on boundary don't have unique nearest site.

A Voronoi diagram is sometimes also known as a Dirichlet tessellation.

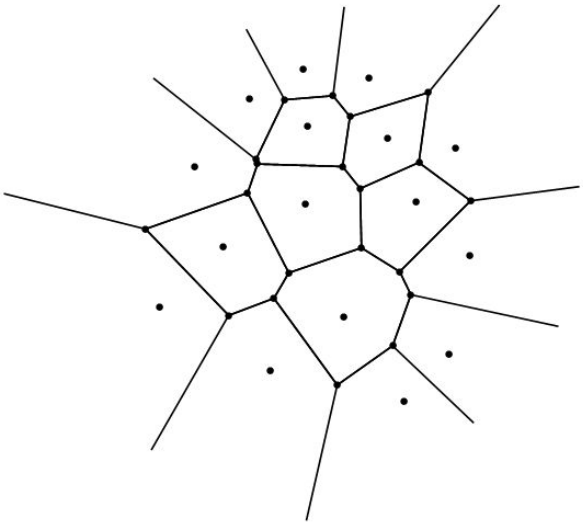
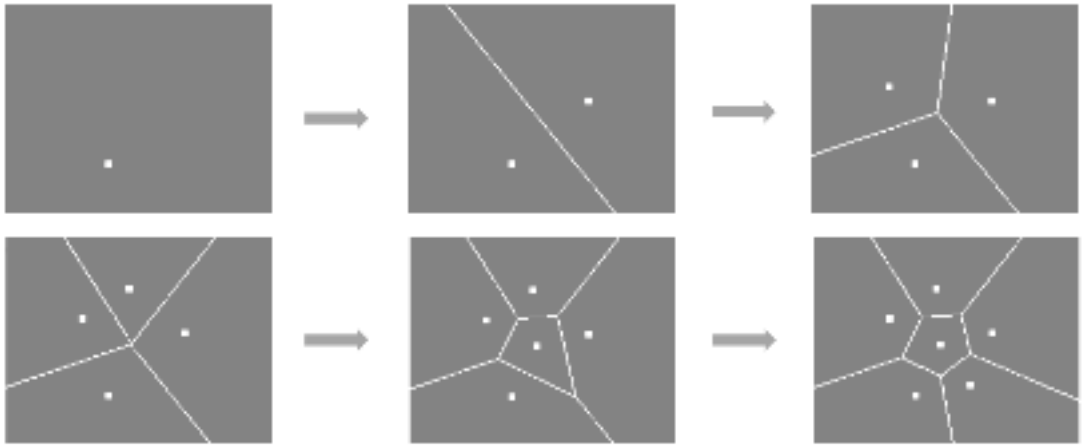


Fig. 2 Voronoi Diagram

Voronoi diagrams can be computed by an incremental construction.



Medial Axis: The medial axis M(P) of a polygon P is the closure of the set of points in P that have two or more closest points among the points of ∂P.

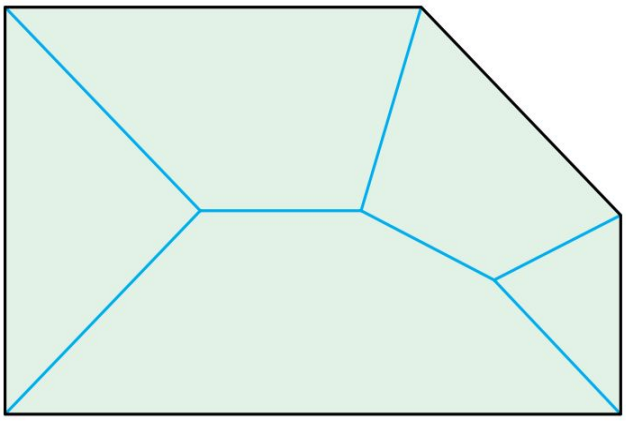


Fig. 3 Medial Axis (Same as Straight Skeleton for convex polygons)

The Medial axis is the locus of the centers of circles that are tangent to the polygon at two or more points.

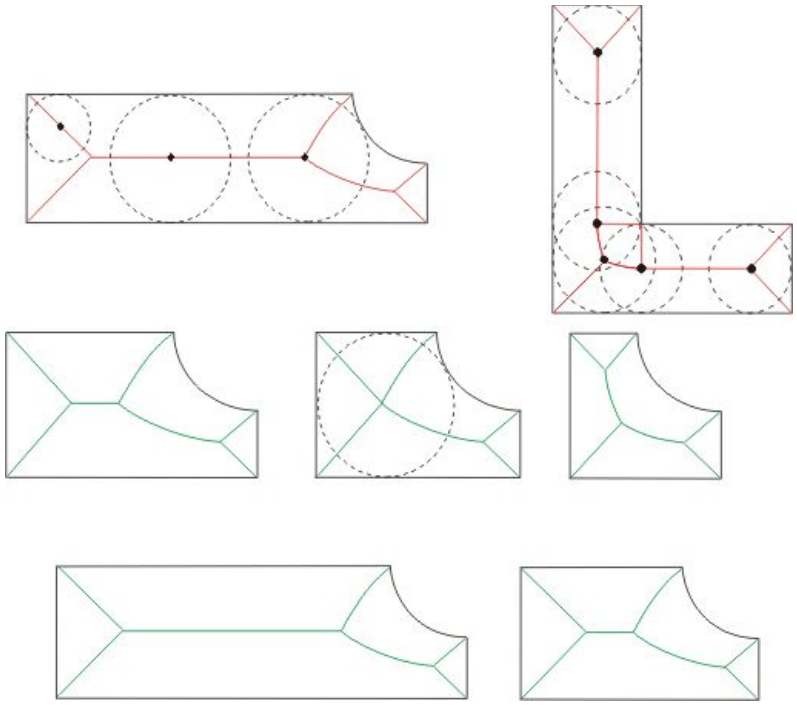


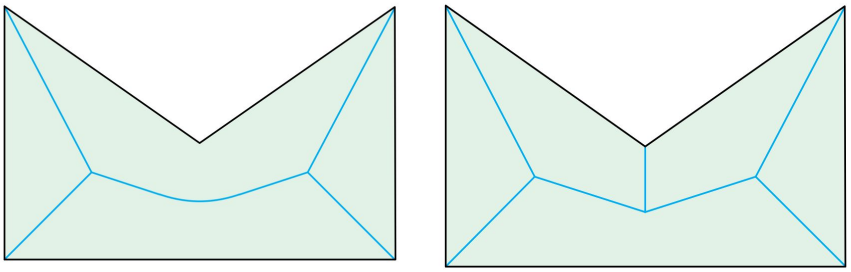
Fig. 4 Medial Axis as the loci of the centers of circles that are tangent to the polygon at two or more points

The Medial Axis comprises straight lines if the polygon is convex. If the object is concave, every reflex vertex induces a curved edge.

Medial Axis Construction

Let P_n be an n-vertex convex polygon. Identify the two adjacent vertices v_i and v_{i+1} whose bisectors meet first, at point x. Extend edges e_{i-1} and e_{i+1} over e_i to meet at a new vertex v and call the resulting polygon P_{n-1}. Compute $M(P_{n-1})$ recursively and delete xv from $M(P_{n-1})$ and add xv_i and xv_{i+1} to form $M(P_n)$.

Straight Skeleton: The Straight Skeleton is a linear approximation of the Medial Axis.



Medial Axis Straight Skeleton

Fig. 5 Straight Skeleton as approximation of the Medial Axis

For a convex polygon, the straight skeleton is identical to the medial axis, but for nonconvex polygons, the straight skeleton has no parabolic arcs. Unlike the medial axis, the straight skeleton is *not* a Voronoi diagram in any sense.

Straight skeleton can be formed with the trace of the angular bisectors of the vertices, as the edges of the polygon are propagating at equal rate, until the polygon vanishes.

Two possible events may occur during the propagation:

Edge event: Portion of an edge vanishes.

Split event: A reflex vertex hits an opposite edge, splitting the polygon into two disconnected parts.

The edges stop propagating when one of these events occur.

Edge Event Split Event

Fig. 6 Edge event and Split Event

Straight Skeleton Computation

Felkel & Obdrzálek offered a straightforward event-based algorithm which applies the principle of the roof construction by sweep algorithm and that simulates cutting of the roof by parallel planes and checks local changes of the polygonal base in the cross sections.

1. Data Structure required for the algorithm:

- A set of circular lists of active vertices (SLAV).
- Stores a loop of vertices for outer boundary and for all holes and sub-polygons created during the straight skeleton computation.
- In the case of convex polygon, it always contains only one list (LAV).

- In the case of a simple non-convex polygon, it stores a list for every sub-polygon and in the case of polygons with holes also a list for each hole.

2. Convex Polygon Skeleton Computation:

Only edge event occurs. The polygon vertices and edges are oriented anticlockwise and the polygon interior is on the left-hand side of its boundary.

2.1. Initialization

Create a LAV for the polygon – a circular list of its vertices by order. Add pointers for the edges between vertices. Compute a bisector per vertex. All vertices are marked “unused”. Compute the intersection point of every set of adjacent bisectors – this point is the location of the edge event between them. Queue the edge event (marked EDGE_EVENT) in a priority queue according to the distance of the intersection from the line supporting the edge.

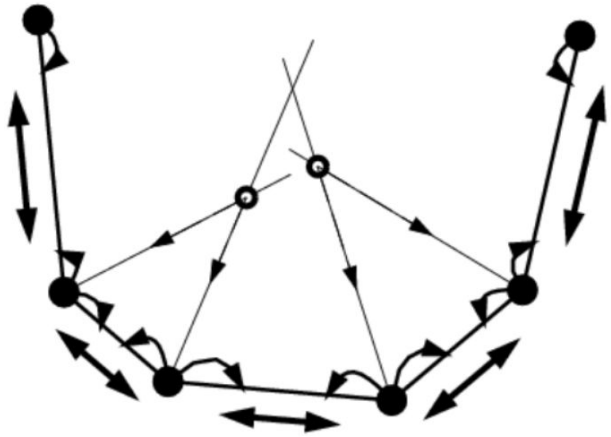


Fig. 7 Creating LAV

2.2. Propagation

- While the events queue != empty do
  - If next event uses used vertices, discard event.
  - Else, handle edge event ;
    - If LAV contains 3 edges, create new vertex at the intersection, and skeletal edges from each of the 3 vertices.
    - If LAV contains more than 3 edges,
      - Create two edges of the skeleton, each from one of the event vertices to the location of the event (the intersection point).
      - Remove these two vertices from the LAV, and mark them as “used”.
      - Create a new vertex, located at the intersection point, and put it in its place in the LAV, pointing to its adjacent edges.
      - Compute new edge events for the vertices of these adjacent edges.

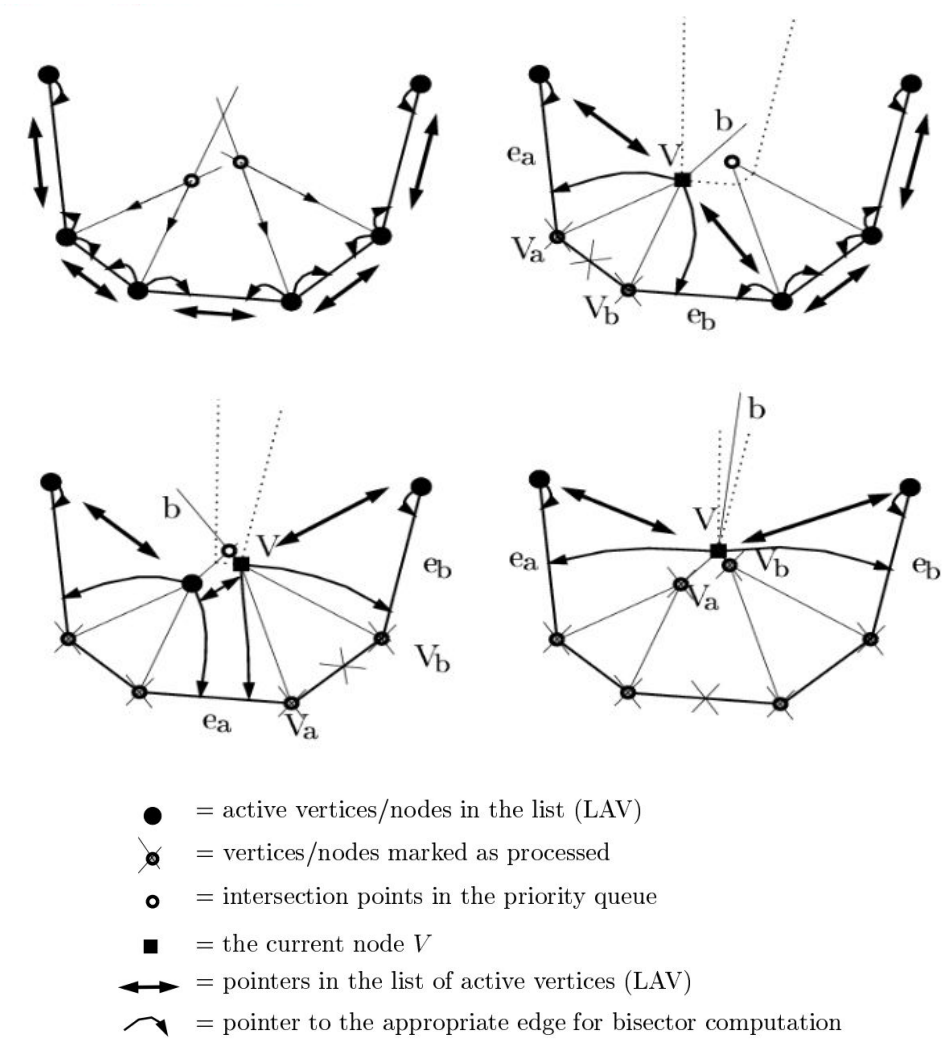


Fig. 8 Propagation in Straight Skeleton COmputation

2.3. Complexity

- Edge events
  - $O(n\ logn)$ time
  - $O(n)$ storage
- Split Events
  - $O(nm)$ time, m reflex vertices
  - $O(n)$ time
- Total $O(nm)$ time with $O(n)$ storage
- The most time-efficient algorithm known has time complexity O($n^{1+\varepsilon}$ +

$n^{8/11 + \varepsilon} r^{9/11 + \varepsilon}$).

3. Non Convex Polygon Skeleton Computation

We have to find out when split events occur. Another step in initialization: Determine all possibilities of a reflex vertex hitting an opposite edge. Queue these events as SPLIT_EVENT.

3.1. Obtaining Split Events

A splitting location B is equidistant from the lines supporting the edges adjacent to the reflex vertex, and from the line supporting the opposite edge. For every reflex vertex, we traverse all of the edges in the polygon and test for intersection. A simple intersection test between the bisector of the reflex vertex and the opposite edge isn't enough. The intersection point between the reflex vertex and the line supporting the opposite edges must be in the area defined between the edge and the bisectors of its two vertices. The intersection point is the meeting point of the three bisectors between all three participating edges (the two defining the reflex vertex and the split edge). Not all reflex vertices eventually cause split events. (A is an edge event, and B is a split event).

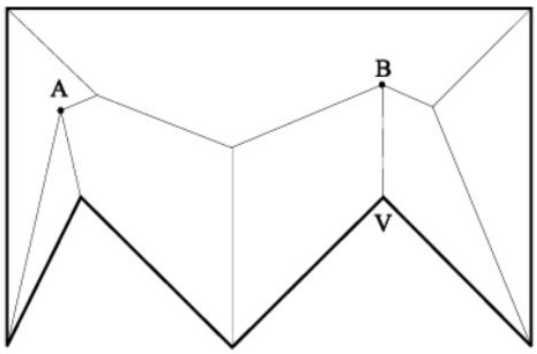


Fig. 9 Obtaining split events

3.2. Handling Split Events

When a split event occurs, the polygon splits into two parts. The LAV in context is split into two LAVs as well.

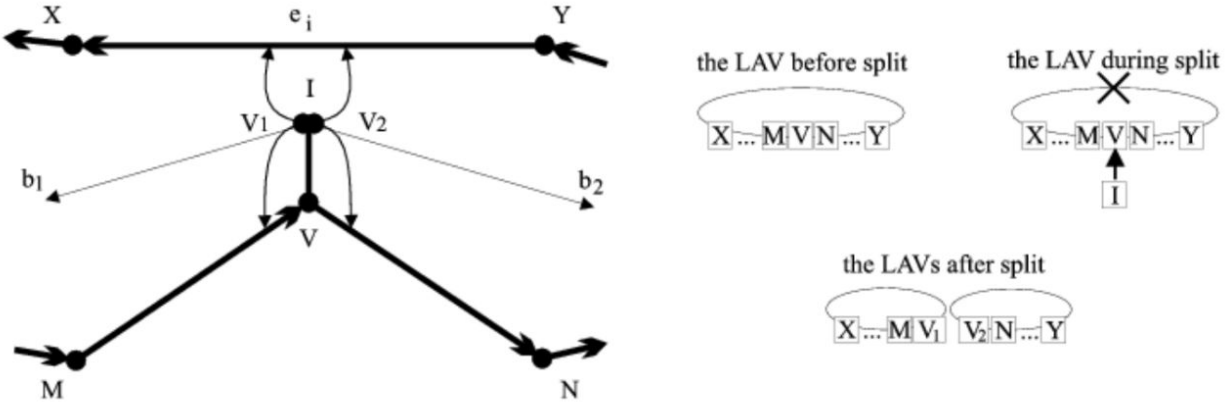


Fig. 10 Handling Split Events

The splitting vertex is replaced with two new vertices, each in the appropriate place in a different LAV. New bisectors and edge events are calculated for each of these vertices.

3.3. Handling Multiple Splitting

An edge can be split several times. Any split event handling must realize what part of the edge it is splitting (i.e. what are the proper endpoints). It is done by traversing the LAV in context at each handling of a split event.

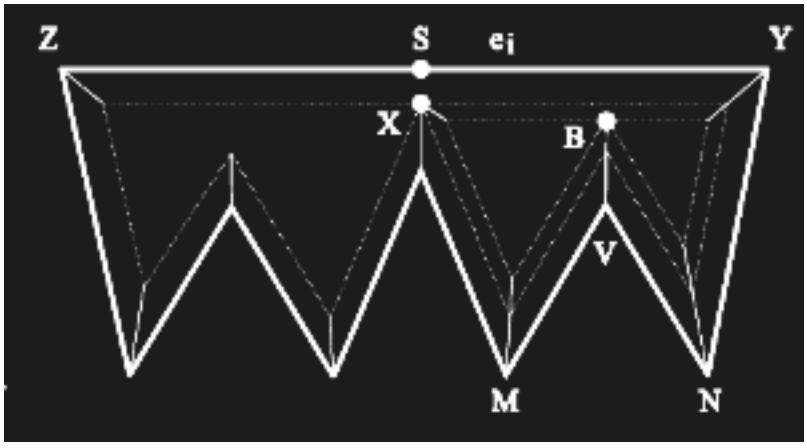


Fig. 11 Handling multiple splitting

3.4. Initialization

Create one LAV.

Compute bisectors.

Compute split and edge events.

Queue all events according to time (distance).

3.5. Propagation

- While event queue has events,
  If new event contains used vertices, discard event.

  If event is edge event, handle as in the convex case.

  Mark vertices as “used”. If the LAV in context contains 3 vertices, close up the skeleton.

  If event is split event, split the LAV into two, and maintain pointers accordingly. Mark the splitting vertex as “used”.
- In the end, there are no LAVs left!

3.6. Complexity

The total complexity is $O(rn + nlogn)$

n = number of vertices

r = number of reflex vertices

Implementation in Python 3.6

Random polygon generation

python random_polygon_generator.py in console

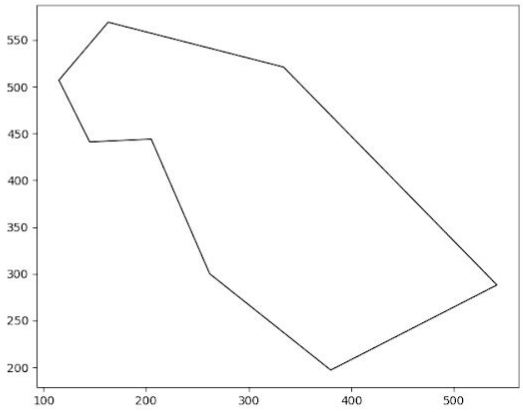


Fig. Random Polygon

Straight Skeleton generation and vertical elevation of the skeleton to build rooftop model

python skelGenerate.py polygonpoints.txt in console

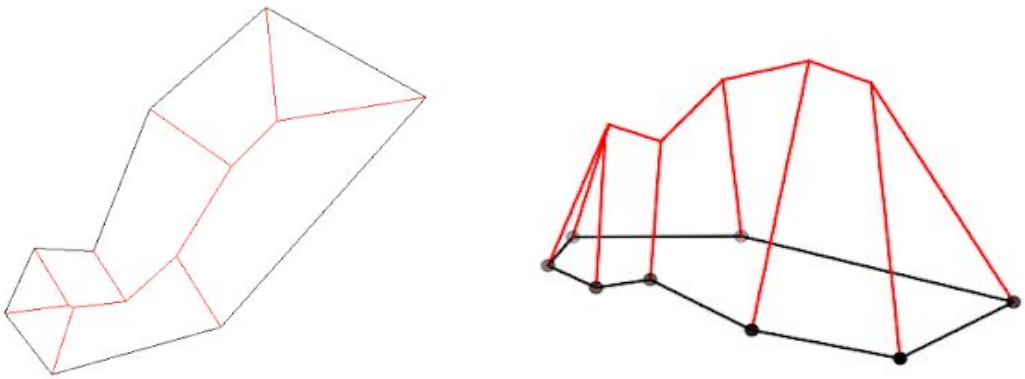


Fig. Straight Skeleton and vertical elevation

Straight Skeleton generation for orthogonal polygons

python skelGenerate.py polygonpoints.txt in console

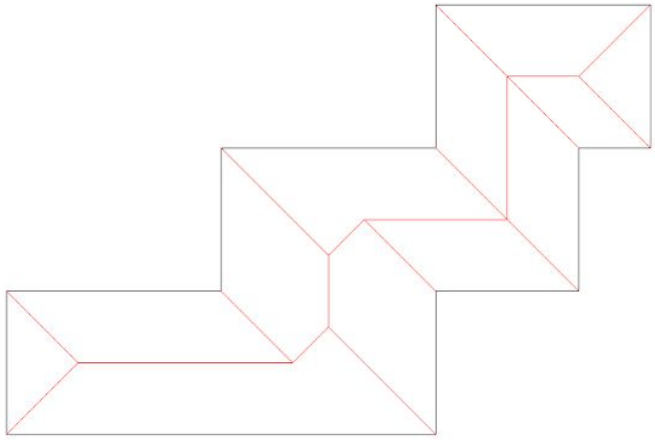


Fig. Straight Skeleton for Orthogonal Polygon

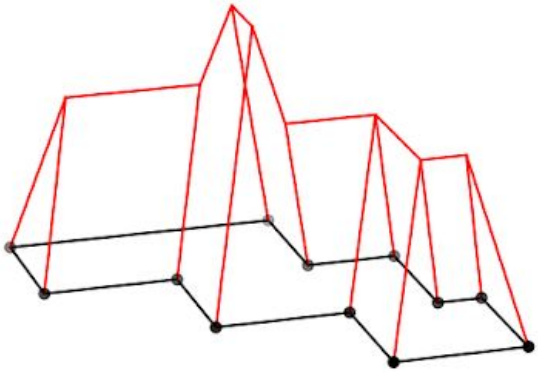


Fig. Vertical Elevation for the straight skeleton of orthogonal polygon

Applications of Straight Skeleton

Origami

The origami polygon cutting theorem tries to solve the fold-and-cut problem. The fold and cut problem can be stated as follows:
We are given an initial shape drawn on a piece of paper. Let's fold the paper down to some other flat shape. Next, let's cut our folded paper along one straight line. Now, unfold the pieces of paper. Can we do this so that the boundary of some of our new pieces of paper outline the drawn shape we initially had? Can we do this for all drawn shapes?
We can also look at the problem like this:
Given a piece of paper in any shape, can we fold it down flat so that all of the shape's edges end up along one straight line? The origami polygon cutting theorem is as follows:
Given any collection of straight edges, there exists a flat folding and a line in that folding such that cutting along it results in the desired pattern of cuts.

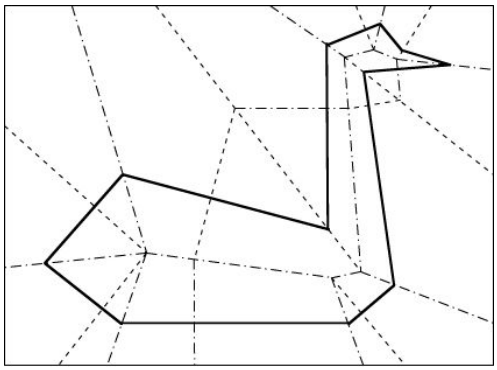


Fig. 12 Origami using Straight Skeleton(Creases are the straight skeleton)

Rooftop

Rooftop can be designed by the concept of straight skeleton of varied 2d polygons. 2d skeleton can be lifted into 3d space.
Aichholzer et al. introduced the roof model, which is a handy way of interpreting the straight skeleton. One considers the wavefront propagation embedded in three-space where the z-axis constitutes time. Then traces out a surface, namely the roof model . This concept gives us the means to investigate over its entire lifespan.
The roof model is sometimes also called terrain model since is a terrain, i.e., any line parallel to the z-axis intersects it in at most one point. As this property may be violated for the weighted version of straight skeletons, we prefer the term “roof model”.

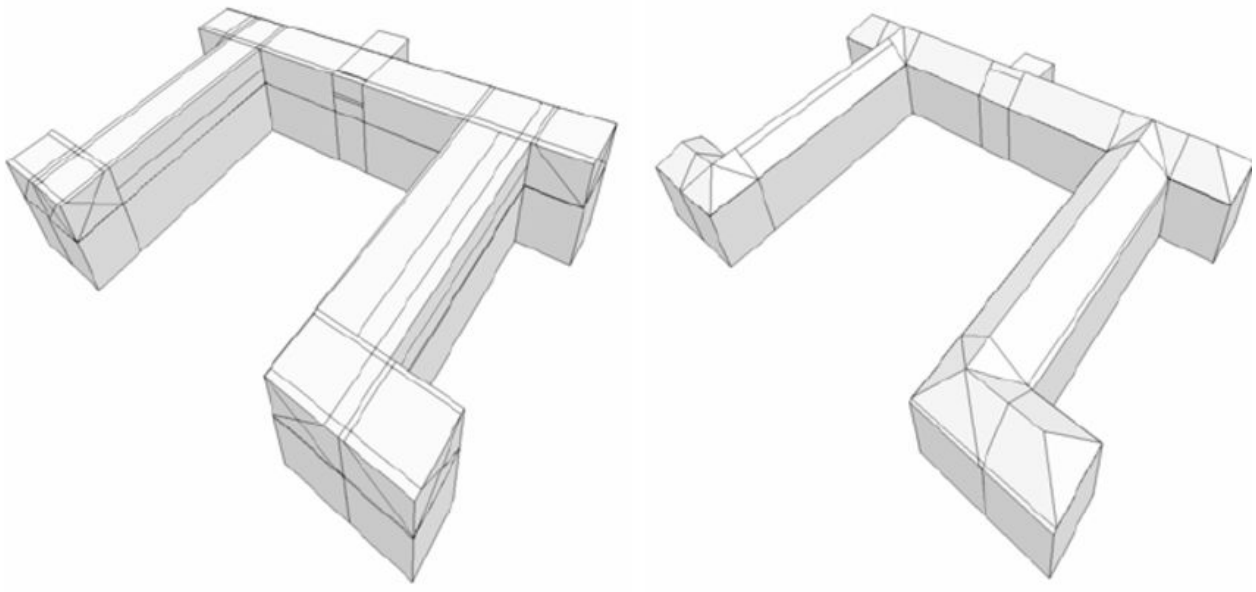


Fig. 13 Obtaining rooftops with Straight Skeleton

Image processing

Tănase and Veltkamp propose to decompose concave polygons into unions of convex regions using straight skeletons, as a preprocessing step for shape matching in image processing.

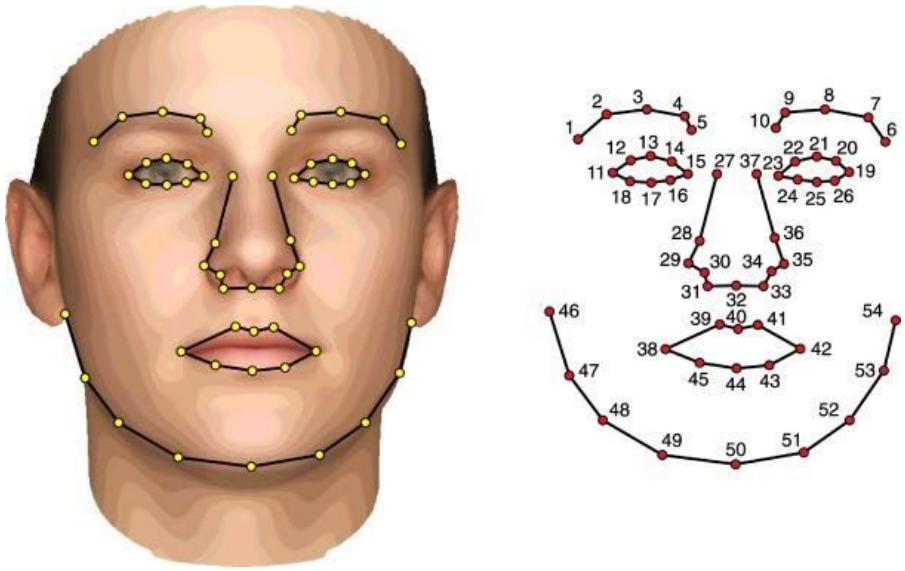


Fig. 14

Curve and surface reconstruction

We can reconstruct curved surface in simulated environment by getting some input points and reconstructing the whole object by computing straight skeleton of the figure.

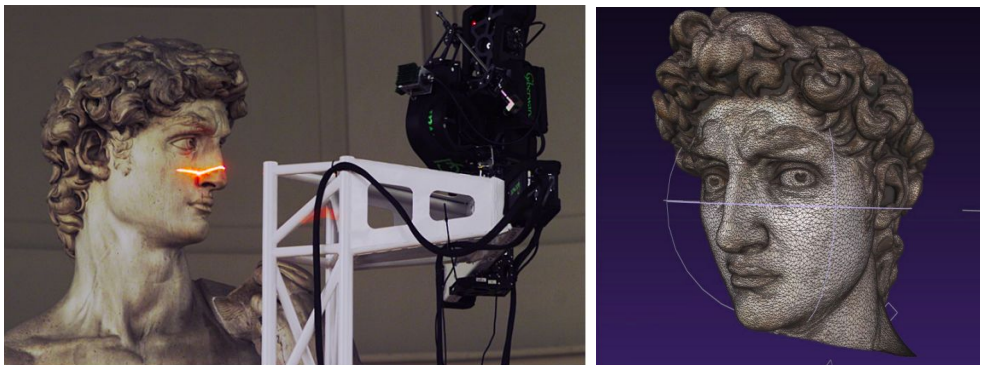


Fig. 15

GIS (Geographic Information System)

It can be used to collapse a two-dimensional area to a simplified one-dimensional representation of the area.
Haunert and Sester describe an application of this type for straight skeletons in geographic information systems, in finding the centerlines of roads.

Fig. 16

Polygonal Chains

Barequet et al. use straight skeletons in an algorithm for finding a three-dimensional surface that interpolates between two given polygonal chains.

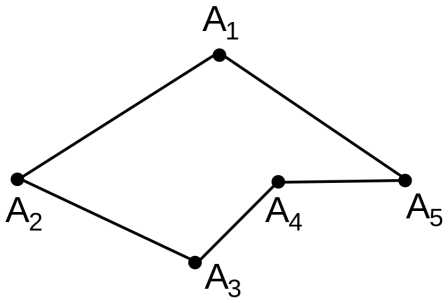


Fig. 17

Conclusion

The straight skeleton can be used to construct a polygonal roof over a set of ground walls, reconstruct a geographical terrain from a river map, or define a pattern of folds that will make all the edges of a paper polygon collinear. The definition of straight skeleton can be generalized to arbitrary planar straight line graphs, where it has (potential) applications to planar motion planning, and three-dimensional polyhedra, where it has potential applications to solid modelling.

Although the medial axis can be constructed in linear time, the fastest known algorithms for straight skeletons are much slower. The main difficulty is that changing the positions or angles of reflex vertices has a significant non-local effect on the skeleton. This nonlocality makes techniques such as incremental construction or divide-and-conquer fail. The following table lists the time and space bounds of various algorithms. Here n is the total number of vertices, r is the number of reflex (nonconvex) vertices, and e is an arbitrarily small positive constant.

In the project, we implemented the algorithm by Felkel et al. using Python and also applied the applet to create model rooftop for building with input wall shape.

References

1. Discrete and Computational Geometry by Satyan L. Devadoss and Joseph O'Rourke

2. http://jeffe.cs.illinois.edu/open/skeleton.html (Straight skeleton of a simple polygon)

3. https://pdfs.semanticscholar.org/cc72/31fa261790e762abb3dacf19f5858a1eb3a9.pdf (Computing Straight Skeletons and Motorcycle Graphs: Theory and Practice)

4. http://www.inf.u-szeged.hu/~palagyi/skel/skel.html#Voronoi

5. http://www.dma.fi.upm.es/personal/mabellanas/tfcs/skeleton/html/documentacion/Straight%20Skeletons%20Implementation.pdf (Straight Skeleton Implementation by Petr Felkel and Stepan Obdrzalek)