Formal study of plane Delaunay triangulation

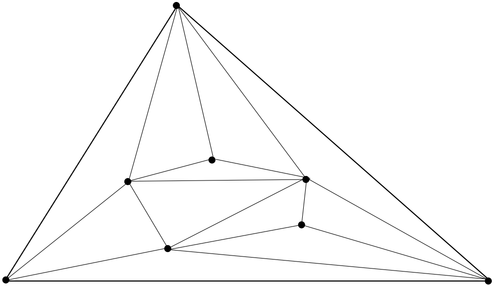
Jean-François Dufourd
Yves Bertot

July 2010

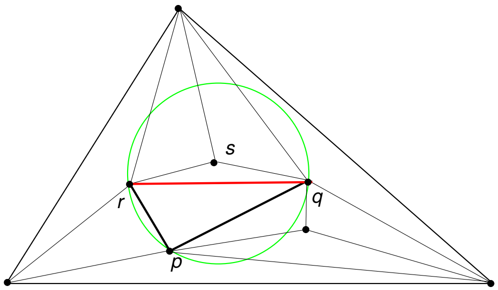
Introduction

- divide objects on the plane into triangles
- Choose a good data-structure for subdivisions
- Remove bad triangles: the ones that are too flat
- Delaunay criterion to recognize bad triangles
- Naive algorithm successive flips
- Method to guarantee termination

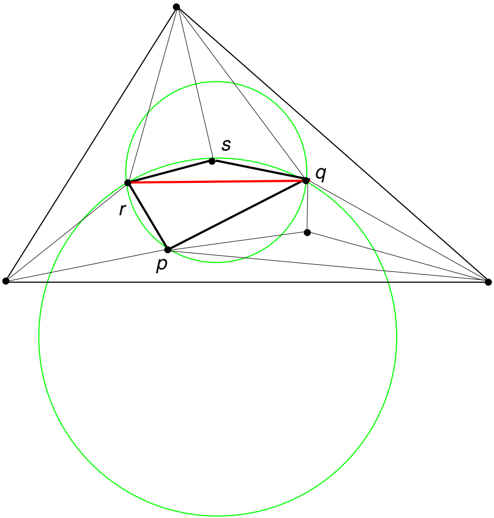
The algorithm at work



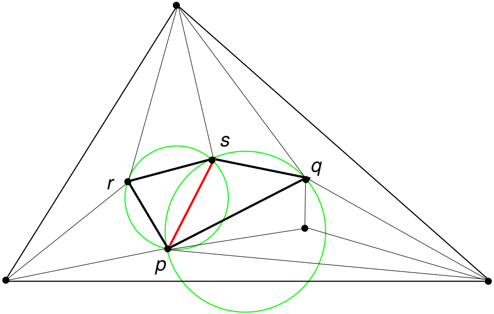
The algorithm at work



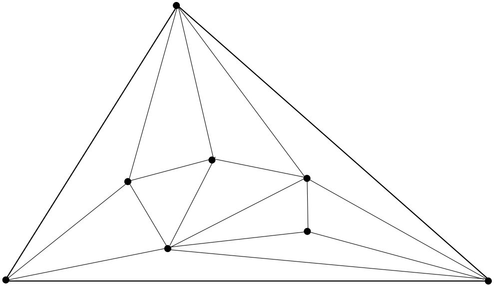
The algorithm at work



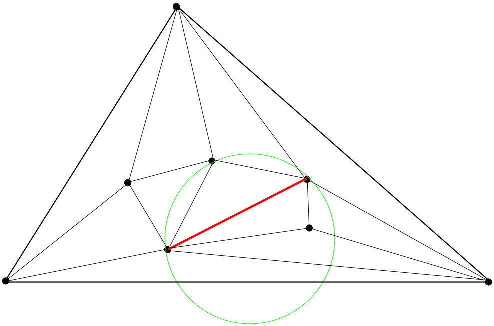
The algorithm at work



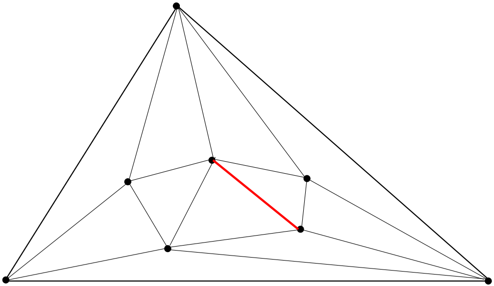
The algorithm at work



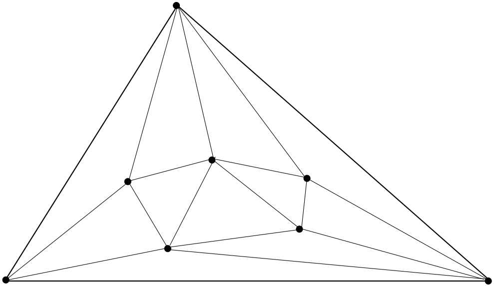
The algorithm at work



The algorithm at work



The algorithm at work



Hypermaps

- What elementary objects compose a triangulation?
- Idea : darts (half-edges),
- Connect darts together : two permutations α_0 α_1
- Two darts linked by α_0 constitute an edge (α_0 involutive)
- all darts in the same α_1 orbit constitute a point
- Implementation content: permutations represent pointers!

Two levels

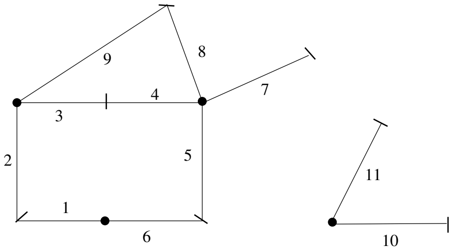
- Use a set of dart identfiers (typically `nat`)
- Use a list structure as a *free* map containg
  - darts
  - links between darts
- Add predicates for:
  - when darts can be added (only if not already present)
  - when links can be added (permutation properties, geometry constraints)
- Recognize faces: actually orbits for $\alpha_1^{-1} \circ \alpha_0^{-1}$

Characterizing triangulations

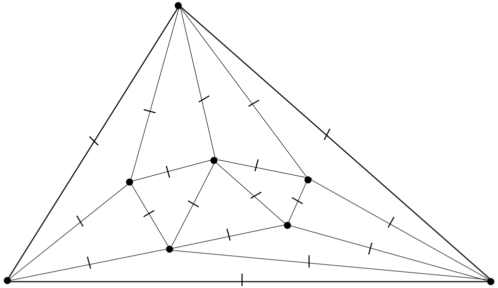
- Planar graphs: Euler formula on point, edge, and face counts
- Triangulations: all faces must be triangular
- Geometry: all triangles must be oriented
  - Except the outer face

Geometric presentation of subdivisions

D	1	2	3	4	5	6	7	8	9	10	11
α0	2	1	4	3	6	5	7	9	8	10	11
α1	6	3	9	5	7	1	8	4	2	11	10



Triangulations as hypermaps



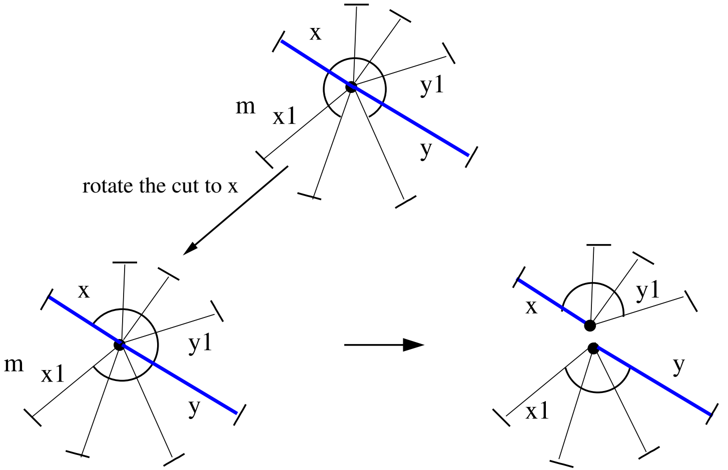
Fine points of permutation handling

- Invariant for map data-structure: only simple links
- $d' = \alpha_i(d)$,
  - if there is a link at dimension i from d to d'
  - if there is no link to d', no link from d, and a chain of links from d' to d
- Orbits are open, they have a cut
- The cut can be "rotated" in the orbit

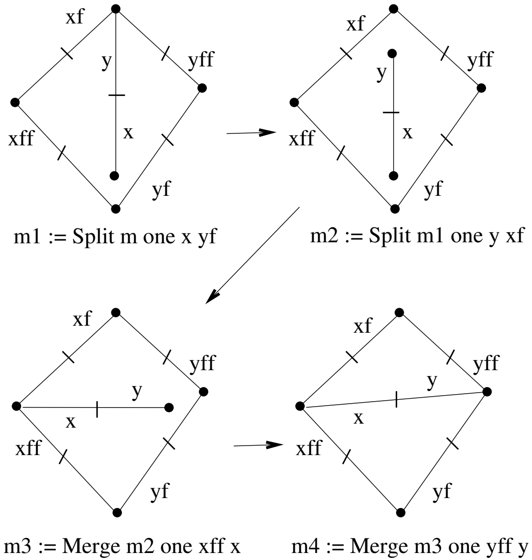
Splitting and merging

- Detaching objects: splitting point orbits
  - Keep together adjacent edges in two sets
  - Done by first rotating, then removing a link
- Merging points
  - Need rotating the orbits to choose how orbits arrange
  - Then add only one link
- Proofs required: make sure the hypermap invariants are preserved
- Merging points also requires a change of coordinates

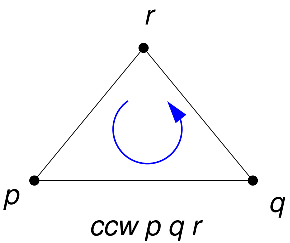
Splitting illustration



Flip : two splits, two merges



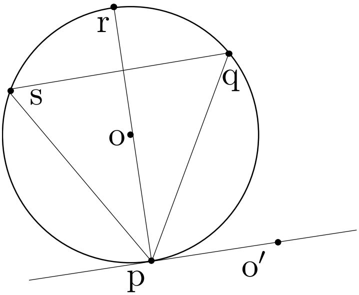
Orientation



Property expressed by an algebraic computation

$$\begin{vmatrix} x_p & y_p & 1 \\ x_q & y_q & 1 \\ x_r & y_r & 1 \end{vmatrix} > 0$$

Orientation and Flip



o is the center of the circumcircle

$p\ o\ q\ r\ s$ are in the configuration of a property known by Knuth

That property was proved formally in 2001 for convex hulls

Detecting illegal edges

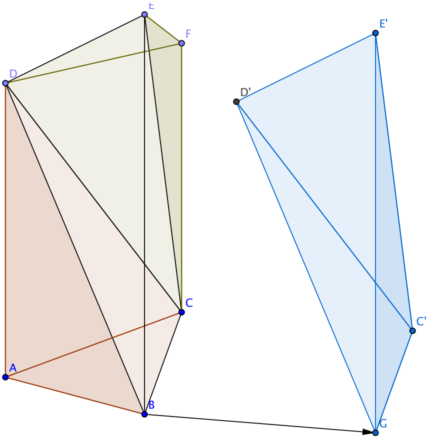
$$\begin{vmatrix} x_p & y_p & {x_p}^2 + {y_p}^2 & 1 \\ x_q & y_q & {x_q}^2 + {y_q}^2 & 1 \\ x_r & y_r & {x_r}^2 + {y_r}^2 & 1 \\ x_s & y_s & {x_s}^2 + {y_s}^2 & 1 \end{vmatrix} > 0$$

This determinant actually is a volume

Ensuring termination

- Project each point onto the revolution paraboloid $z = x^2 + y^2$
- Thus define a triangulated surface in space
- Consider the volume under this surface
- This volume decreases everytime one flips an illegal edge
- Termination argument: only a finite set of possible triangulations
- We developed a generic approach to describe this “finiteness” argument

Computing volumes



Conclusion

- Data structures closer to graphs
- Hypermaps actually represent efficient representations in memory
- Also adapted to handle more dimensions (add α_2 etcetera)
- Need to add a function to create the initial triangulation
- Need to also consider arbitrary external faces
- Wish: use this functional model as a basis to study an imperative implementation