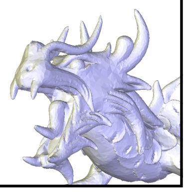
Lyon 1

Mesh and Computational Geometry

Raphaëlle Chaine
Université Claude Bernard Lyon 1

M2 ID3D
Image, Développement
et Technologie 3D
et 3A Centrale



1

Properties

- There are interesting interpretations of the Delaunay triangulation, lifting the points into a space of higher dimension
  - What is Delaunay in 1D?
    - Delaunay is like sorting and linking each point with the next one

    OR
    - Projecting the 1D points on the lower side of a 2D convex and computing the convex envelop of the 2D lifted points
  - Delaunay = lower envelope of points

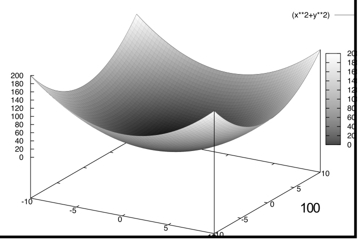
99

99

Properties

- Nice interpretation of Delaunay in the space (« space of spheres »)
  - Correspondence between the Delaunay triangulation and the lower convex envelope of the points lifted on the paraboloid

$$z = x^2 + y^2$$

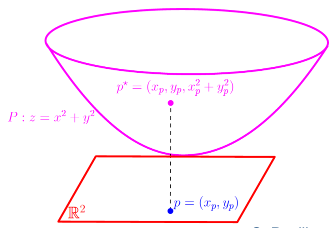


100

100

- Each 2D point x_P, y_P is lifted to $x_P, y_P, x_P^2+y_P^2$ on the paraboloid

$$z = x^2 + y^2$$



O. Devillers

101

101

Demonstration

- Let us consider a circle of the plan

$$(M - C)^2 = R^2$$
$$(x - x_C)^2 + (y - y_C)^2 = R^2$$
$$x^2 + y^2 - 2xx_C - 2yy_C + x_C^2 + y_C^2 - R^2 = 0$$
$$x^2 + y^2 - 2ax - 2by + c = 0$$

  - Where are the points of the circle lifted on the paraboloid?

102

102

Demonstration

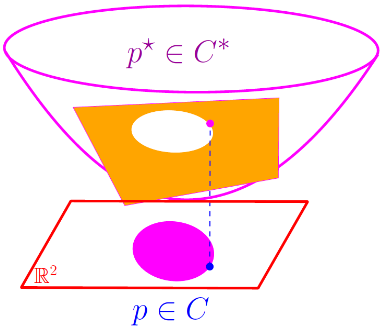
- Circle $x^2 + y^2 - 2ax - 2by + c = 0$
  - all the points belonging to this circle are lifted to points belonging simultaneously to the paraboloid and the plane

$$z - 2ax - 2by + c = 0$$

103

103

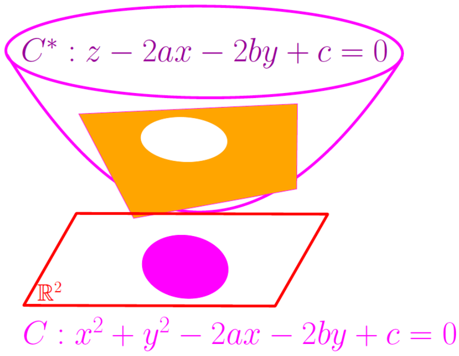
Demonstration



104

104

Demonstration

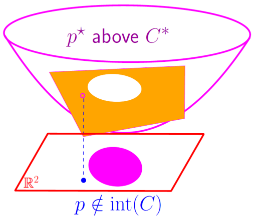


105

105

Demonstration

- What about the points outside the circle C when they are lifted?
  - They are lifted above the plane C*

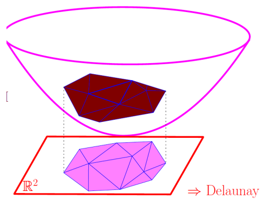


106

106

Demonstration

- Thus, if 3 points correspond to a Delaunay triangle, none of the other points can be lifted under the plane corresponding to the 3 points :
  - Correspondance between the Delaunay triangulation and the lower enveloppe of the points lifted to the paraboloid.



107

107

Back to Lawson's algorithm

- Transformation of a 2D triangulation into a Delaunay triangulation
  - As long as there is a non-locally Delaunay edge (ie a non Delaunay subtriangulation of 4 points)
    - Replace the subtriangulation by the alternative subtriangulation (flip)
- What complexity?

108

108

Back to Lawson's algorithm

- By using Delaunay interpretation in the spheres space, it is possible to show that **an edge cannot appear more than once in the queue** (even with other incident triangles)

109

109

Back to Lawson's algorithm

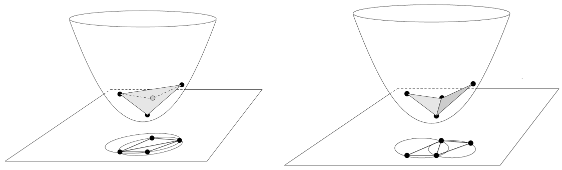
- Interpretation of the *flip* in the “space of spheres”
  - Before the *flip*, each of the two triangles has the 4th point in its circumscribed circle
  - The lifting of each triangle on the paraboloid is located above the 4th point.

110

110

Back to Lawson's algorithm

- Interpretation of the *flip* in the “space of spheres”



- Each *flip*, allows the lifted surface to descend locally

111

111

Back to Lawson's algorithm

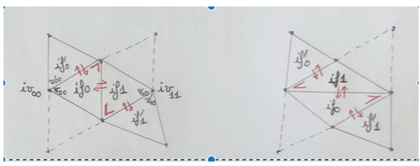
- This means that a flipped edge cannot reappear a second time in the flow of the algorithm
- The number of edges that can be formed with n points is n(n-1)/2
- Lawson's algorithm is therefore in $O(n^2)$
- Much more efficient in practice!

112

112

Operations for triangulating and flipping edges

- Updating the Mesh data structure
  - Division of a triangle into 3 triangles (Split)
  - Flip of an edge (Flip)



113

113

Delaunay triangulation in any dimension n

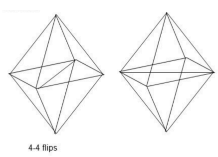
- Paving the convex-hull of the points with n-simplices whose circumscribed sphere is empty
- Warning: Lawson's algorithm is only valid in 2D, because the notion of edge flip is more complicated in larger dimensions

114

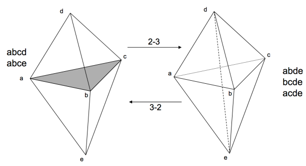
114

Flip in 3D

- 4-4 flip



- 2-3 and 3-2 flips



115

115

Other use of the flip operation

- Insertion of a point P outside the convex-hull of a triangulation
  - New triangles should be created using P and the boundary edges that are visible from P
  - The corresponding infinite faces should be destroyed

116

116

Other use of the flip operation

- Insertion of a 2D point P outside the convex-hull of a 2D triangulation
- A possible implementation using the infinite vertex and the flip operation
  - Split InfF, one of the infinite face to be destroyed into 3
  - Iteratively perform flips on the infinite edges bounding that modified area if they are incident to an other infinite face that should disappear (starting from InfF)

117

117

Geometric algorithms implementation

- Distinction between exact, combinatorial vs. approximate objects
- Input data (considered as exacts) used to construct exact non combinatorial objects
  - points x,y or x,y,z
- Construction of combinatorial objects from the exact ones
- Approximate objets should be constructed for visualisation purpose only

118

118

Algorithmic using predicates

- The progress of an algorithm should only depend on the sign of predicates evaluated accurately
  - Use of a controlled arithmetic
  - No use of inexact objects in the evaluation of predicates

119

119

Algorithmic using predicates

- Without these precautions, there is a risk of aberrant behaviour of a geometric algorithm
- Example: How to express the simple insertion algorithm in a 2D triangulation according to these criteria?

120

120

Algorithmic using predicates

- Algorithm of simple insertion in a 2D triangulation:
  - Using the three points orientation predicate
    - To perform inclusion tests in a triangle
    - To perform visibility tests on an edge of the convex envelope

121

121

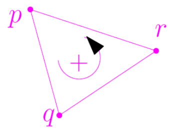
Algorithmic using predicates

- Three 2D points orientation predicate:

$$orientation(p,q,r) = sign(((q-p)\times(r-p)).Oz)$$

$$orientation(p,q,r) = sign((q_x - p_x)(r_y - p_y) - (q_y - p_y)(r_x - p_x))$$

$$= \begin{vmatrix} q_x - p_x & r_x - p_x \\ q_y - p_y & r_y - p_y \end{vmatrix}$$



$$orientation(p,q,r) = sign(\det \begin{bmatrix} 1 & p_x & p_y \\ 1 & q_x & q_y \\ 1 & r_x & r_y \end{bmatrix})$$

122

122

Algorithmic using predicates

- How to evaluate this orientation predicate?
  - In the case where the input coordinates belong to the regular grid of integers?
  - In the case where the input coordinates are rationals?

123

123

Algorithmic using predicates

- How to evaluate this orientation predicate?
  - In the case of the input coordinates belong tho the regular grid of integers?
  - In the case where the input coordinates are rationals?
  - In both cases the evaluation can be carried out accurately since we benefit from an exact multiplication, addition and subtraction for these types of numbers!

124

124

Algorithmic using predicates

- How to evaluate this orientation predicate?
  - In the case where input coordinates are double?

$$\pm m 2^e \qquad -1023 < e < 1024$$
$$m = 1.m_1 m_2 \dots m_{52} \ (m_i \in \{0,1\})$$

  - The result of the arithmetic operations is rounded to the nearest double

125

125

Algorithmic using predicates

- How to evaluate this orientation predicate?
  - In the case where input coordinates are double?
  - It is only the sign of the predicates that matters
  - The case where the three points are almost aligned can be error-prone

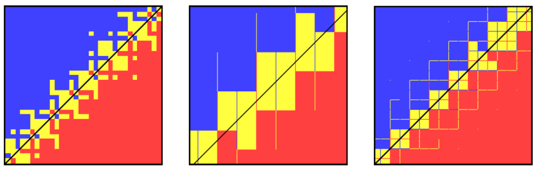
126

126

Algorithmic using predicates

- If the orientation predicate is evaluated using double arithmetic :
  - Result of orientation(p,q,r) with $p(p_x+Xu_x, p_y+Yu_y)$

$$0 \le X,Y \le 255 \qquad u_x = u_y = 2^{-53}$$



$p: \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$, $q: \begin{pmatrix} 12 \\ 12 \end{pmatrix}$, $r: \begin{pmatrix} 24 \\ 24 \end{pmatrix}$

$\begin{pmatrix} 0.50000000000002531 \\ 0.5000000000000171 \end{pmatrix}$, $\begin{pmatrix} 17.300000000000001 \\ 17.300000000000001 \end{pmatrix}$, $\begin{pmatrix} 24.00000000000005 \\ 24.0000000000000517765 \end{pmatrix}$

$\begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$, $\begin{pmatrix} 8.8000000000000007 \\ 8.8000000000000007 \end{pmatrix}$, $\begin{pmatrix} 12.1 \\ 12.1 \end{pmatrix}$

Images by S. Pion

127

Algorithmic using predicates

- If the orientation predicate is evaluated using double arithmetic :
  - There are still values for which the sign is unambiguously certified (need to control the threshold)
  - Otherwise :
    - Consider the coordinates of p, q and r as rational (with a finer precision than that usually considered) and make the exact calculation

Algorithmic unsing predicates

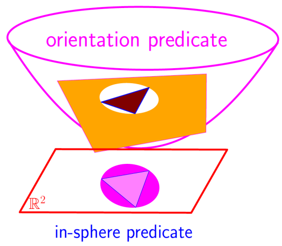
- Example : How to express Lawson algorithm using predicates?

Algorithmic using predicates

- Lawson Algorithm :
  - **Do not** base the evaluation of a predicate on the construction of an inaccurate temporary object (e. g. the centre of a circumscribed circle)
  - An AB edge should be flipped if the circle circumscribed to one of its 2 incident triangles ABC contains point D located on the other side

Algorithmic using predicates

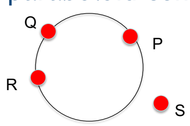
- Reminder: the in-circle inclusion test can be expressed as an orientation test in a space of higher dimension (space of spheres)



Images par O. Devillers

Algorithmic using predicates

- Predicate of inclusion of a point s in a circle circumscribed at p,q and r (orientation of the 4 points lifted on the paraboloid centered at p)

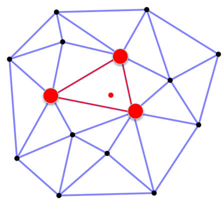


- In_cercle(p,q,r,s)
  =-signe(((Φ(q)-Φ(p))x(Φ(r)-Φ(p))).(Φ(s)-Φ(p))

$$= - \mathrm{sign}\begin{vmatrix} q_x - p_x & r_x - p_x & s_x - p_x \\ q_y - p_y & r_y - p_y & s_y - p_y \\ (q_x-p_x)^2+(q_y-p_y)^2 & (r_x-p_x)^2+(r_y-p_y)^2 & (s_x-p_x)^2+(s_y-p_y)^2 \end{vmatrix}$$

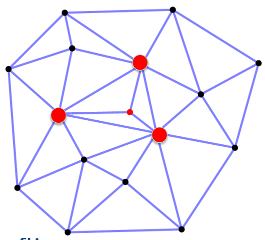
Back to Delaunay triangulation

- How to modify the incremental algorithm of insertion into a simple triangulation to obtain an incremental algorithm of insertion into a Delaunay triangulation?



Incremental insertion into a Delaunay triangulation

- First perform a simple insertion :



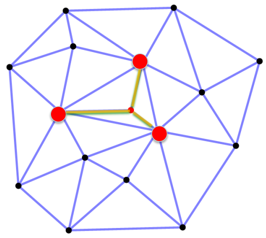
  - Use Lawson flips
  - Is it necessary to test all the edges?

134

134

Delaunay incremental insertion

- Let Δ be the triangle in which P was inserted
  - After the simple insertion, the triangle is star-shaped with respect to P

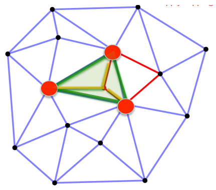


135

135

Incremental Delaunay insertion

- Let Δ be the triangle in which P was inserted
  - After the simple insertion, only the 3 edges of Δ are likely to be candidates for flipping
  - The 3 new edges incident to P could not be flipped
  - The others have not changed their pair of incident triangles

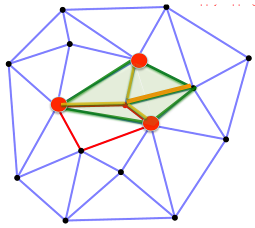


136

136

Incremental Delaunay insertion

- Each flip generates a new edge incident to P and two edges are added to the boundary of the modified area (in green)
- Green edges are likely to be flipped (since one of their incident triangle has been modified)

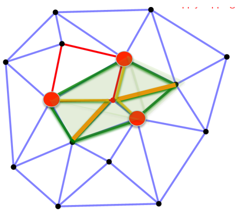


137

137

Incremental Delaunay insertion

- Each flip generates a new edge incident to P and two edges are added to the boundary of the modified area (in green)
- Green edges are likely to be flipped (since one of their incident triangle has been modified)

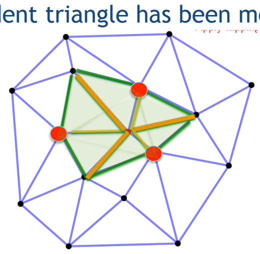


138

138

Incremental Delaunay insertion

- Each flip generates a new edge incident to P and two edges are added to the boundary of the modified area (in green)
- Green edges are likely to be flipped (since one of their incident triangle has been modified)



139

139

Incremental Delaunay Algorithm

- The complexity of each insertion directly corresponds to the final number i of edges incident at the new vertex.
  - Every flipped edge gave birth to an edge incident to P ->there are i-3 flips
  - The flipping test was checked on each edge that was effectively flipped, and also on the green boundary of the resulting modified area (i edges)

140

140

Incremental Delaunay Algorithm

- An insertion outside the convex envelope also starts as the simple insertion into a triangulation
  - Additional flips can be performed on the boundary of the modified area

141

141

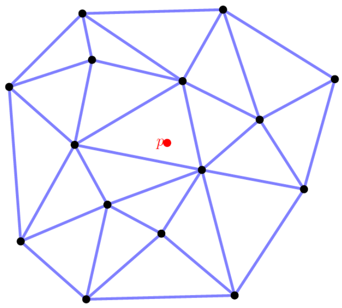
Delaunay incremental insertion (Alternative version)

- We just showed that the modified area of the triangulation is star-shaped around the inserted point P
- Alternative approach for incremental Delaunay :
  - Delete all the triangles whose circumscribed circle contains point P
  - Those triangles are said to be "in conflict" with P
  - Triangulate the conflict zone by star-shaping the conflict area around P

142

142

Incremental Delaunay Algorithm (Alternative Version)

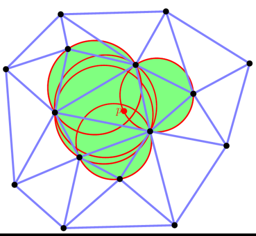


143

143

Delaunay incremental algorithm (Alternative Version)

- Determine the in-conflict triangles
  - Using breadth-first search (or deapth-first search) on the adajacency graph of triangles starting from Δ

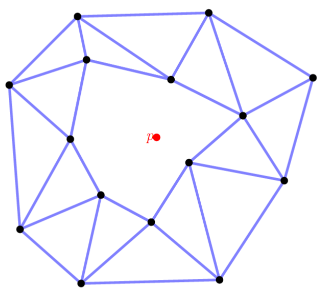


144

144

Delaunay incremental algorithm (Alternative Version)

- Conflict zone determination

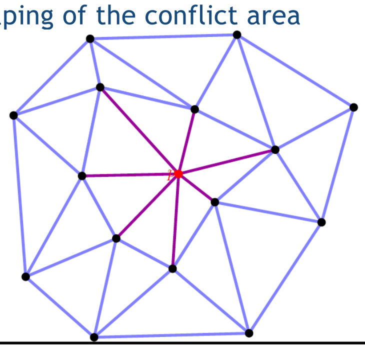


145

145

Delaunay incremental algorithm (Alternative Version)

- Star-shaping of the conflict area



146

146

Delaunay incremental algorithm (Alternative Version)

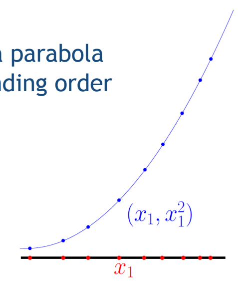
- Star-shaping of the conflict area
- Validity of this algorithm in higher dimension
  - In 2D, the **edges of the boundary** of the conflict zone get connected to the inserted point by constructing new triangles
  - In 3D, the Delaunay triangulation is composed of tetrahedrons. The **triangles of the boundary** of the conflict zone get connected to the inserted point by constructing new tetrahedrons

147

147

Incremental Delaunay Algorithm

- Complexity analysis
- Worst case :
  - Points distributed on a parabola and inserted in descending order of abscissa.

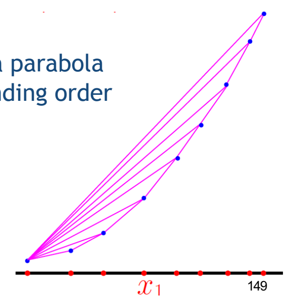


148

148

Incremental Delaunay Algorithm

- Complexity analysis
- Worst case :
  - Points distributed on a parabola and inserted in descending order of abscissa.



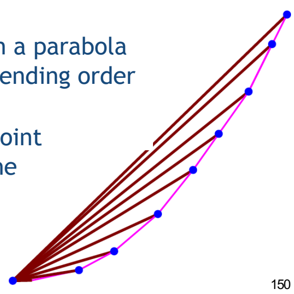
149

149

Incremental Delaunay Algorithm

- Complexity analysis
- Worst case :
  - Points distributed on a parabola and inserted in descending order of abscissa.
  - Each new inserted point conflicts with ALL the triangles

$$\Omega(n^2)$$



150

150

Incremental Delaunay Algorithm

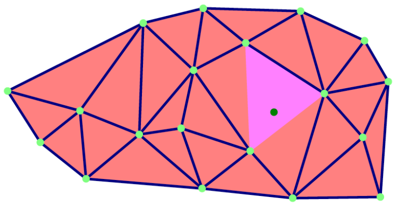
- Complexity analysis
- In average :
  - Complexity dependent on the strategy used to locate the triangle containing the point to be inserted

151

151

Location strategies

- Exhaustive search among all the triangles



$O(n)$

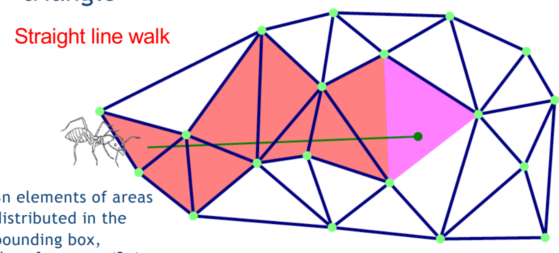
Images by O. Devillers

152

152

Location strategies

- Search from a first randomly selected triangle



3n elements of areas distributed in the bounding box, therefore squrt(3n) elements encountered on average on a straight line

$O(\sqrt{n})$ on average

153

153

Location strategies

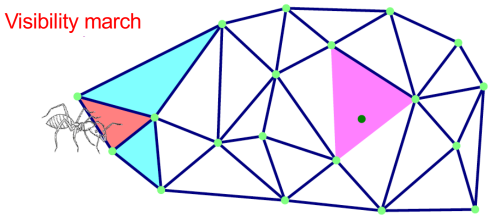
- Search from a first randomly selected triangle
  - Straight walk
  - Requires a predicate of segments intersection

154

154

Location strategies

- Search from a first randomly selected triangle
  - Some minor deviations from the straight line walk

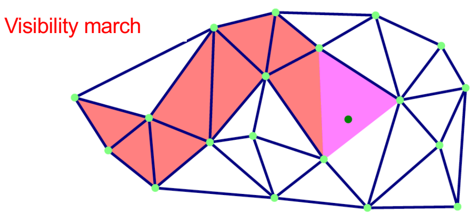


155

155

Location strategies

- Search from a first randomly selected triangle
  - Some minor deviations from the straight line walk



156

156

Location strategies

- Search from a first randomly selected triangle
  - Visibility walking
  - Only requires an orientation predicate to find the next triangle to walk in

157

157