CAD & Computational Geometry

Course plan

- Introduction
- Segment-Segment intersections
- Polygon Triangulation
- Intro to Voronoï Diagrams & Delaunay Triangulations
- Geometric Search
- Sweeping algorithm for Voronoï Diagrams

Delaunay Triangulation

- Delaunay triangulation algorithms

  Two alternatives

  1- Generate a triangulation for a given set of points, known in advance

  - Using Fortune’s algorithm (for Voronoï diagrams) and “dualization”
  - Ad-hoc algorithms, not necessarily incremental

  2- Points are generated “in line”, at the same time the triangulation is updated

  - Mesh generation / mesh adaptation
  - Algorithms must be incremental

Delaunay Triangulation

- Principles behind incremental algorithms for Delaunay triangulations
  - Insertion of a point p_n in a triangulation of $n–1$ points

    - Find the entity “containing” the new point

    - Insert (topologically) the point in the triangulation

    - Modify the triangulation in order to keep the “Delaunay” characteristic

      There exist two close algorithms (equivalent since the Delaunay triangulation is unique)

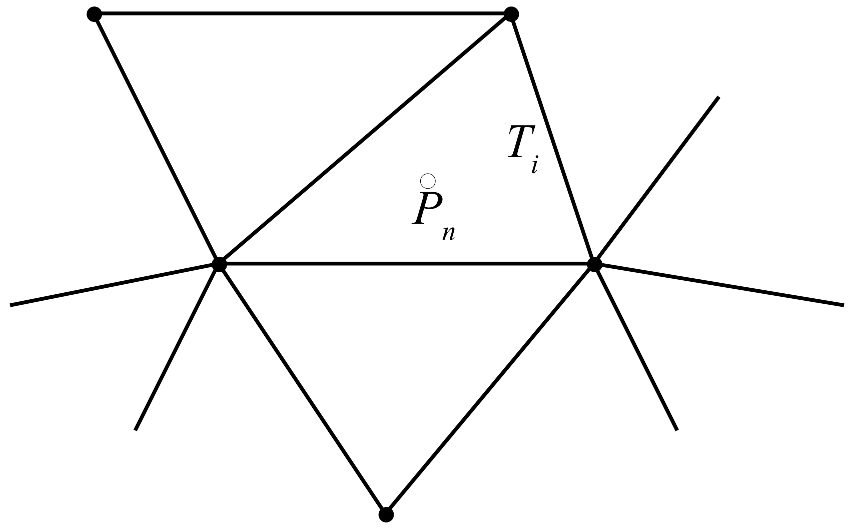
      - Lawson – edge swapping
        a valid (and better) triangulation is guaranteed at each step.
      - Bowyer-Watson – Find every triangle that violate the empty sphere criterion, delete them and build a new cavity. Mesh the cavity (star-shaped cavity), i.e. link every boundary edge with the new point.
        Drawback: it is possible to build non star-shaped cavities (even unconnected) if predicates are evaluated in finite numerical precision.

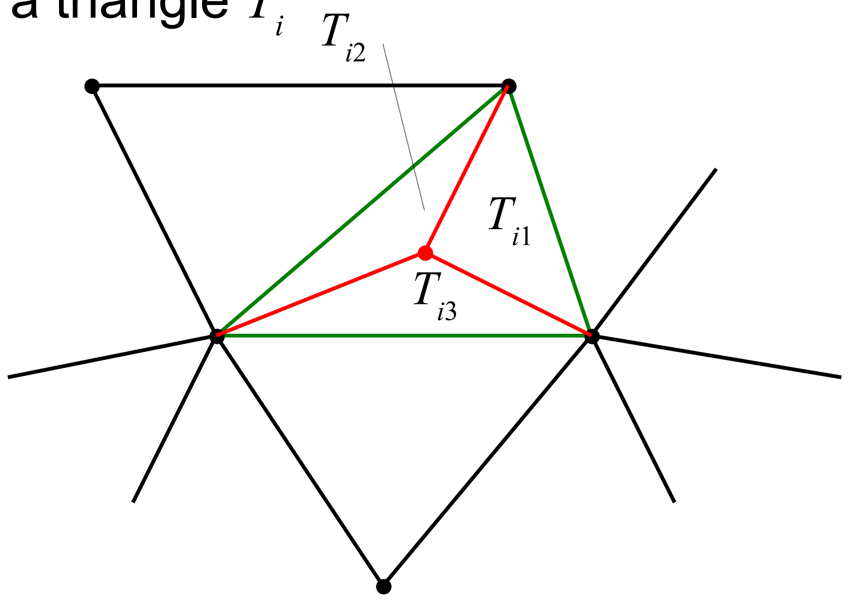
CAD & Computational Geometry

Delaunay Triangulation

- Where is the entity containing the new point ? Geometric search in general – see next course, sometimes we know already where to insert the new point
- Insertion of the point in the existing triangulation.

Case 1 : the point p_n is located in a triangle T_i



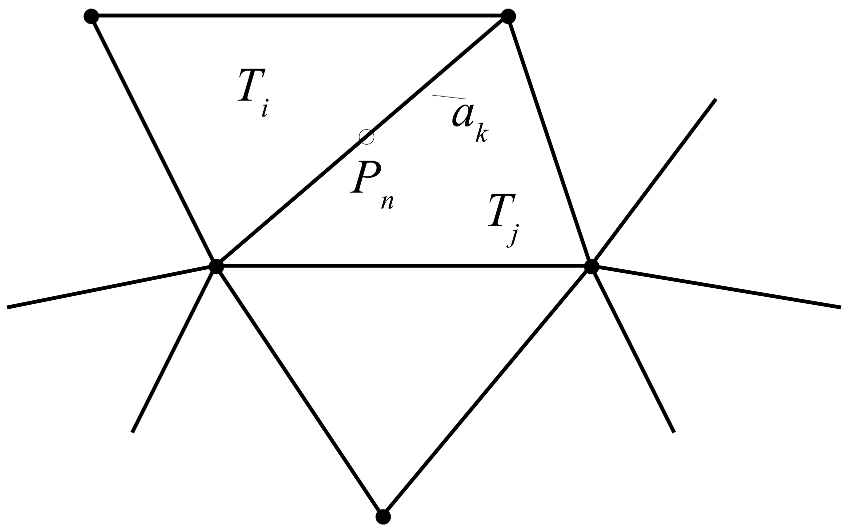


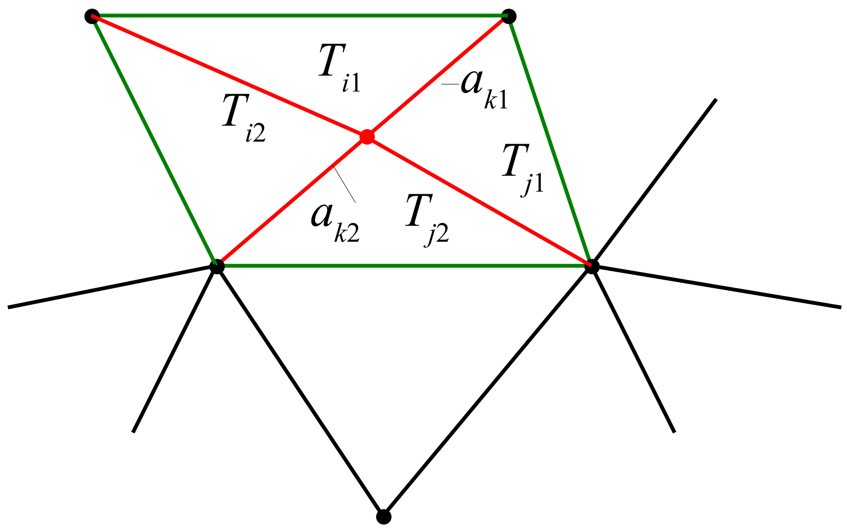
- The red edges are legal by construction.

Delaunay Triangulation

- Insertion of the point in the existing triangulation.

Case 2 : the p_n is located on an edge a_k



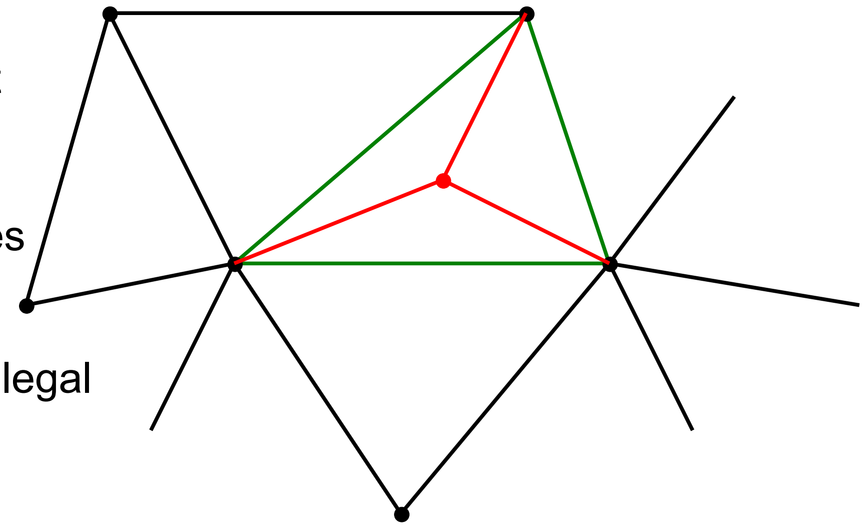


- The new edges in red are also legal by construction.

- What about the other triangles / edges (the green ones in particular) ?

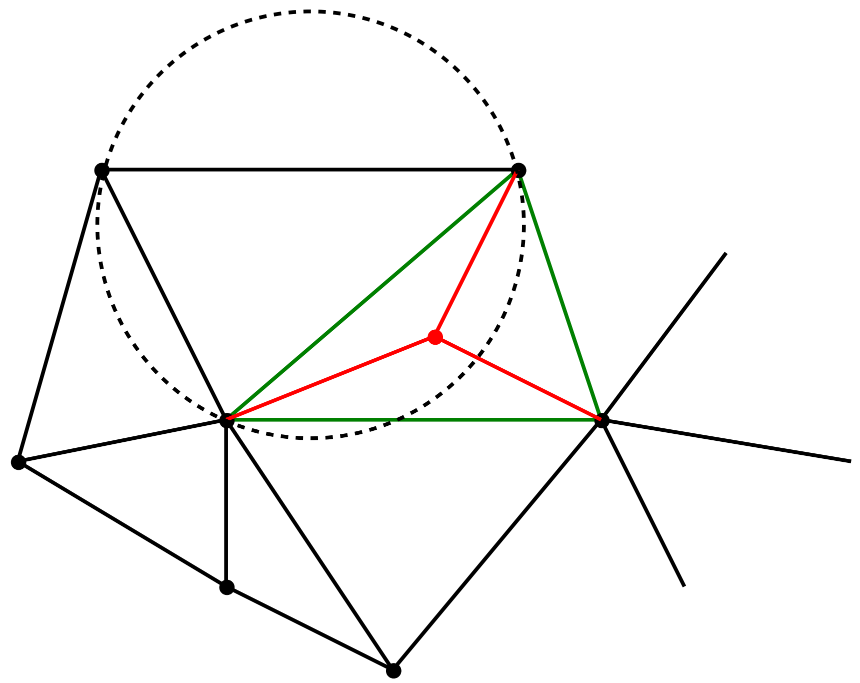
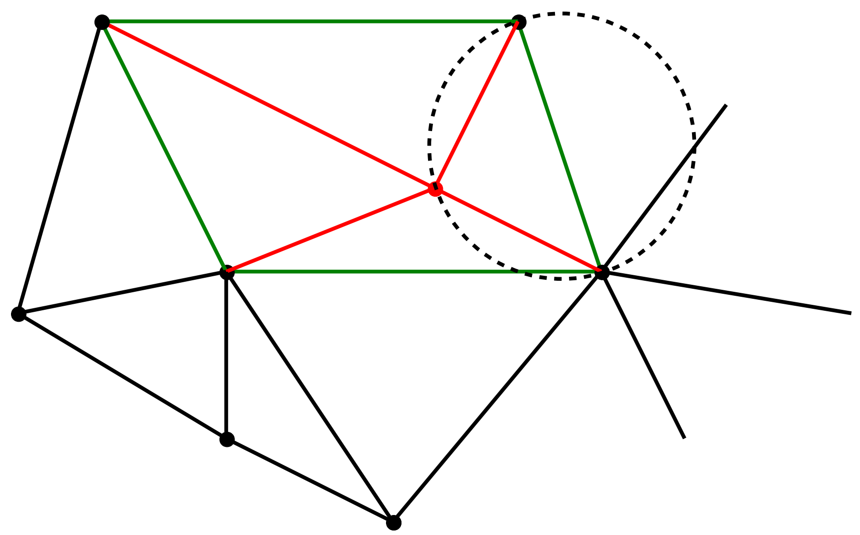
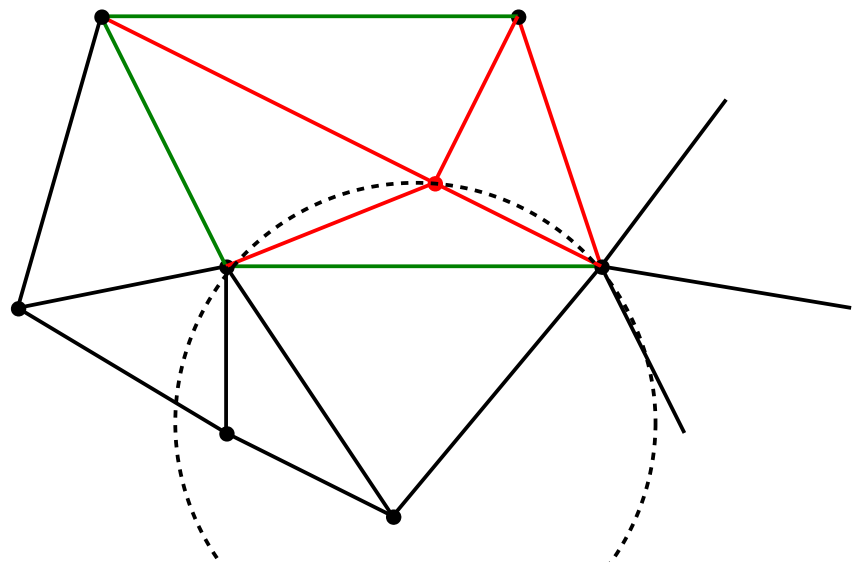
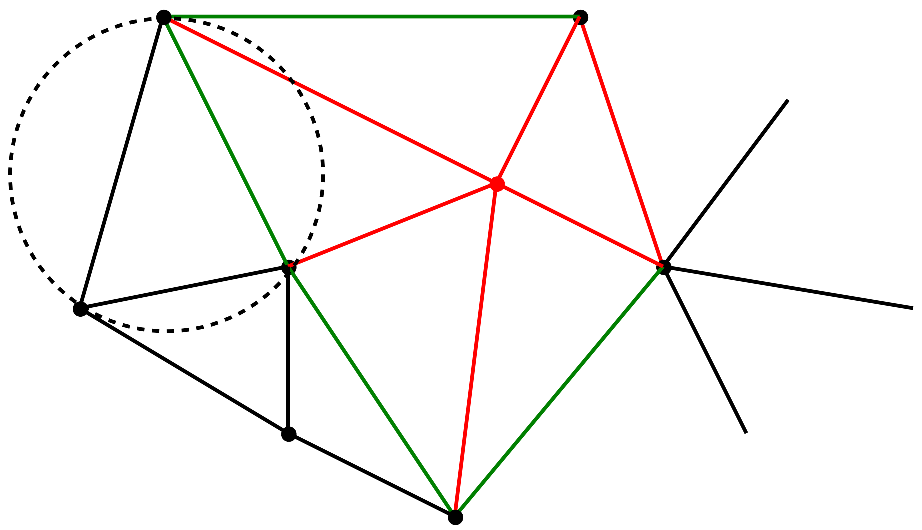
Delaunay Triangulation

- Modify the triangulation so that the “Delaunay” characteristic is kept.
  - It amounts to legalize every edge that became illegal following the insertion of the new vertex int the triangulation
  - What are these edges ?
    - These are the ones that have new neighboring triangles following the insertion. (here in green)
    - All the other edges (black) are still legal for the time being, because nothing changed in their neighborhood..
    - We should therefore check if these green edges are still legal, and perform edge swapping if this is not the case.
    - When an edge-swap happens, one must check the neighboring triangles in order to make sure all edges are still legal. If not, swap the next layer of illegal edges, until no new illegal edge is found



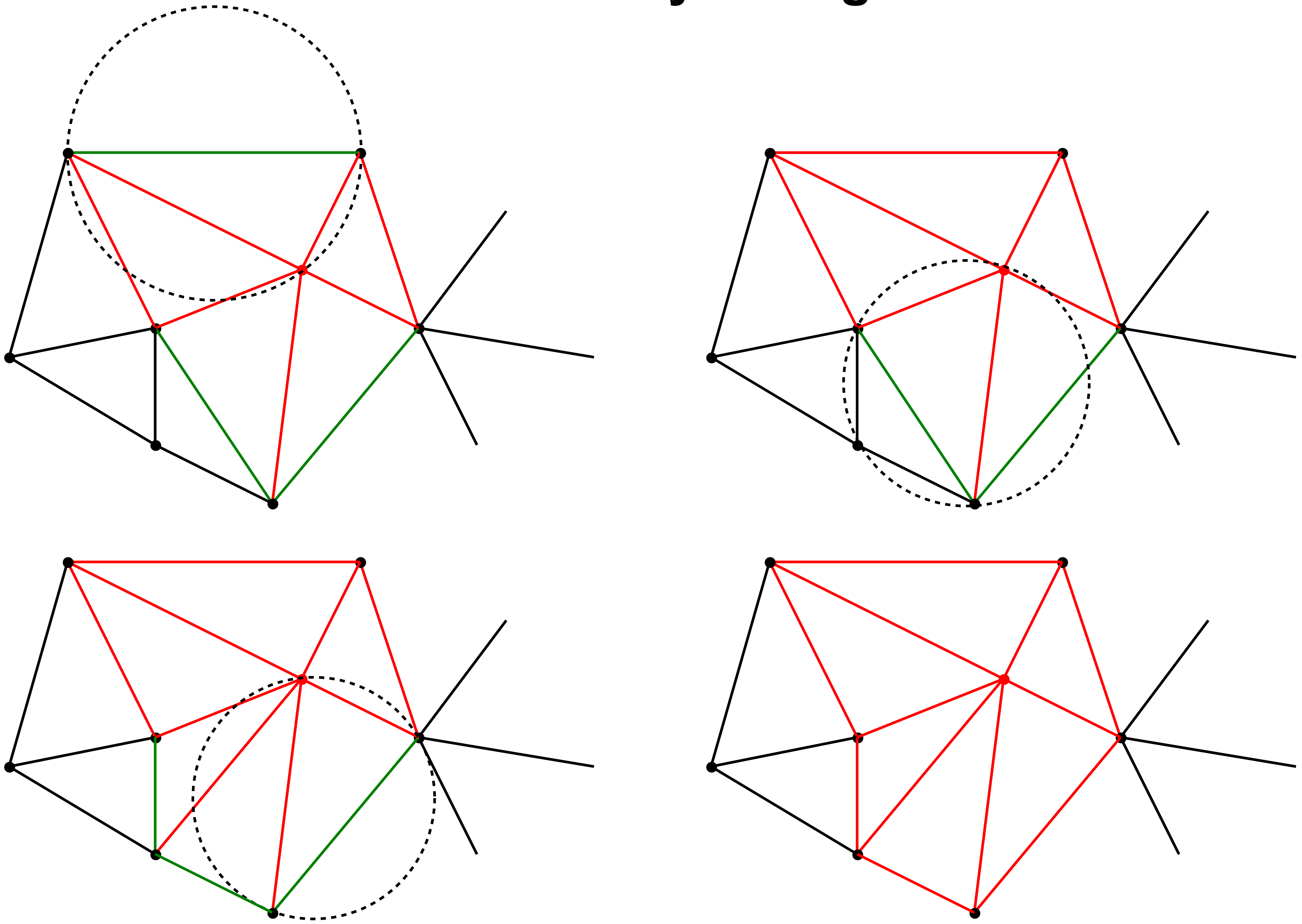
CAD & Computational Geometry

Delaunay Triangulation

CAD & Computational Geometry

Delaunay Triangulation



Delaunay Triangulation

- Global algorithm

```
InsertPoint($p_r$,$T$)
Input : a point $p_r$ to be inserted, and a Delaunay triangulation
Output : A Delaunay triangulation containing $p_r$.
{
 Find the triangle $T_i$ ($p_i$, $p_j$, $p_k$) containing $p_r$
 If $p_r$ is inside $T_i$
 {
  Cut $T_i$ in 3 and add the edges linking $p_r$ and $p_i$,$p_j$ and $p_k$
  LegalizeEdge($p_r$, $p_i p_j$, $T$)
  LegalizeEdge($p_r$, $p_j p_k$, $T$)
  LegalizeEdge($p_r$, $p_k p_i$, $T$)
 }
 Else ($p_r$ is on an edge , e.g. $p_i p_j$ , $p_k$ and $p_l$ are the opposite points to $p_i p_j$)
 {
  Cut both triangles neighbors to $p_i p_j$ in 4 and insert the edges linking $p_r$ and $p_i$, $p_j$, $p_k$, $p_l$.
  LegalizeEdge($p_r$, $p_i p_l$, $T$) ;    LegalizeEdge($p_r$, $p_l p_j$, $T$) ;
  LegalizeEdge($p_r$, $p_j p_k$, $T$) ;    LegalizeEdge($p_r$, $p_k p_i$, $T$) ;
 }
}
```

Delaunay Triangulation

- Swap an edge and make it legal

```
LegalizeEdge($p_r$, $p_ip_j$, $T$)
{
 If ($p_ip_j$) is an illegal edge
  {
   Let $p_ip_jp_k$ the triangle adjacent to $p_rp_ip_j$ through edge $p_ip_j$
   Flip $p_ip_j$ and replace it by $p_rp_k$
   LegalizeEdge($p_r$, $p_ip_k$, $T$)
   LegalizeEdge($p_r$, $p_kp_j$, $T$)
  }
}
```

Delaunay Triangulation

- Proof that the algorithm is correct

  We have to prove that no illegal edge remains after the insertion of point p_r.

  - One may notice that :

  - Every new edge (by swapping) is linked to p_r

  - An edge ought to become illegal only if an adjacent triangle is modified (by the insertion of p_r or by edge swapping) – it will necessarily be checked later on because of the recursion

  - Every new edge obtained by edge swapping is therefore legal and belong to the Delaunay graph (proof follows)

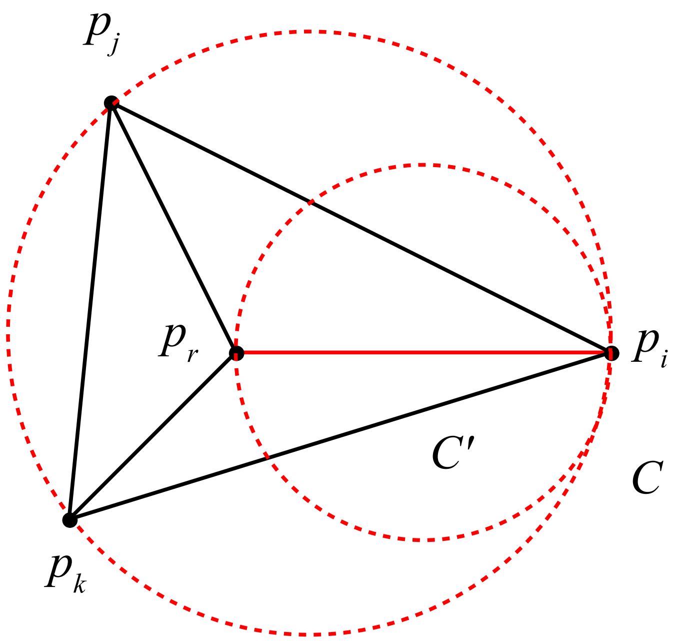
  QED.

Delaunay Triangulation

- The new edges belong the the Delaunay graph

  Case of the insertion of p_r.
  Lets consider the edges $p_r p_i$, $p_r p_j$ and $p_r p_k$. As $p_r p_j p_k$ is a triangle from before the insertion of p_r, its circumscribed circle C does not contain any point other than p_r. One can therefore shrink C, such that one gets a new circle C' going through p_r and p_i and entirely contained in C. It therefore does not contain any other point. This means that $p_r p_i$ belongs to the Delaunay graph (the conclusion is obviously identical for $p_r p_j$ and $p_r p_k$).

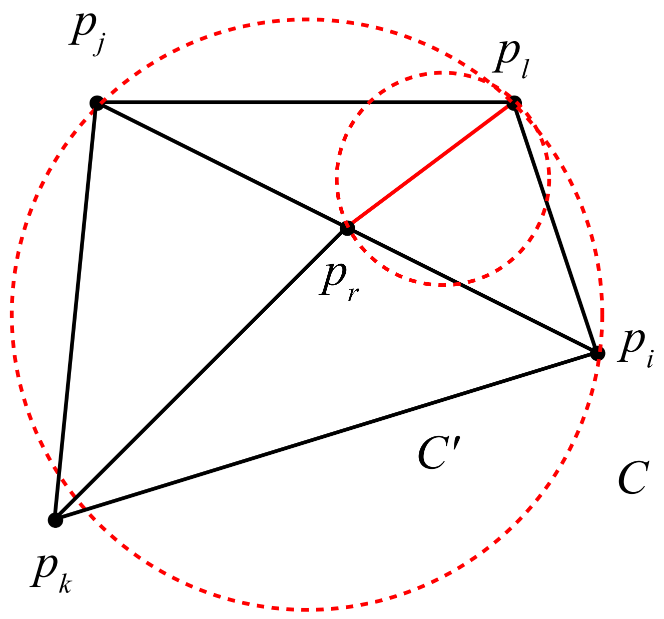
- Proof is similar if p_r is located on an edge.



CAD & Computational Geometry

Delaunay Triangulation

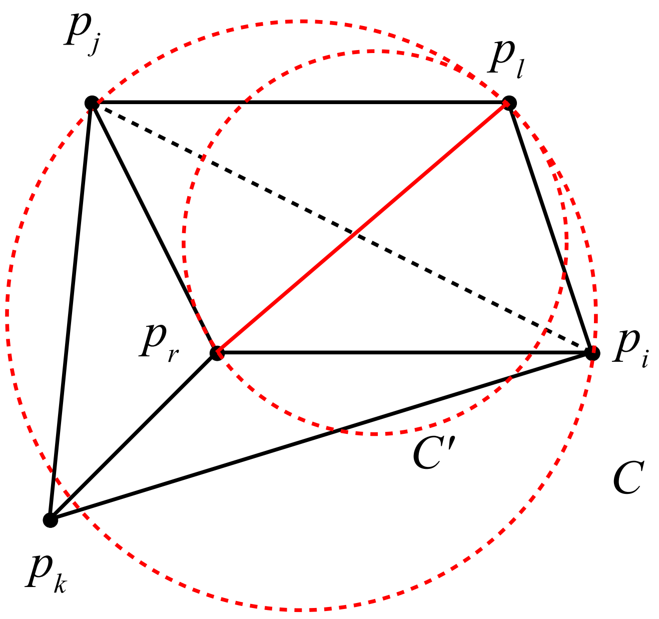
- Proof is similar if p_r is located on an edge.



Delaunay Triangulation

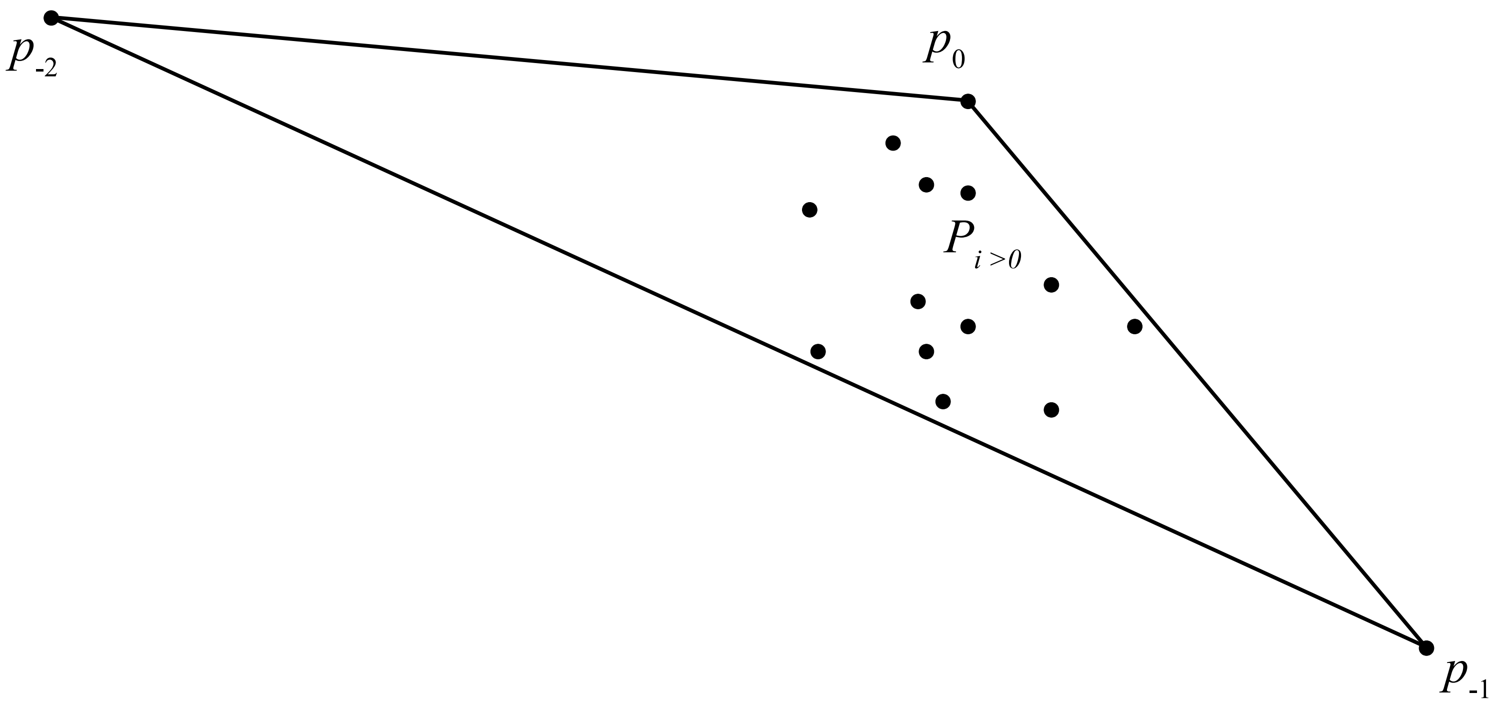
- Do the new edges obtained by swapping belong the the Delaunay graph ?

  One replaces an edge $p_i p_j$ by another edge $p_r p_l$. As $p_i p_j p_l$ is a triangle from before inserting p_r, its circumscribed circle C contains only p_r (and if it did not contain p_r, then $p_i p_j$ would be legal). It is also possible to find another circle C' going through p_r and p_l , completely contained in C, thus containing no point. This proves that $p_r p_l$ indeed belong the the Delaunay graph.



Delaunay Triangulation

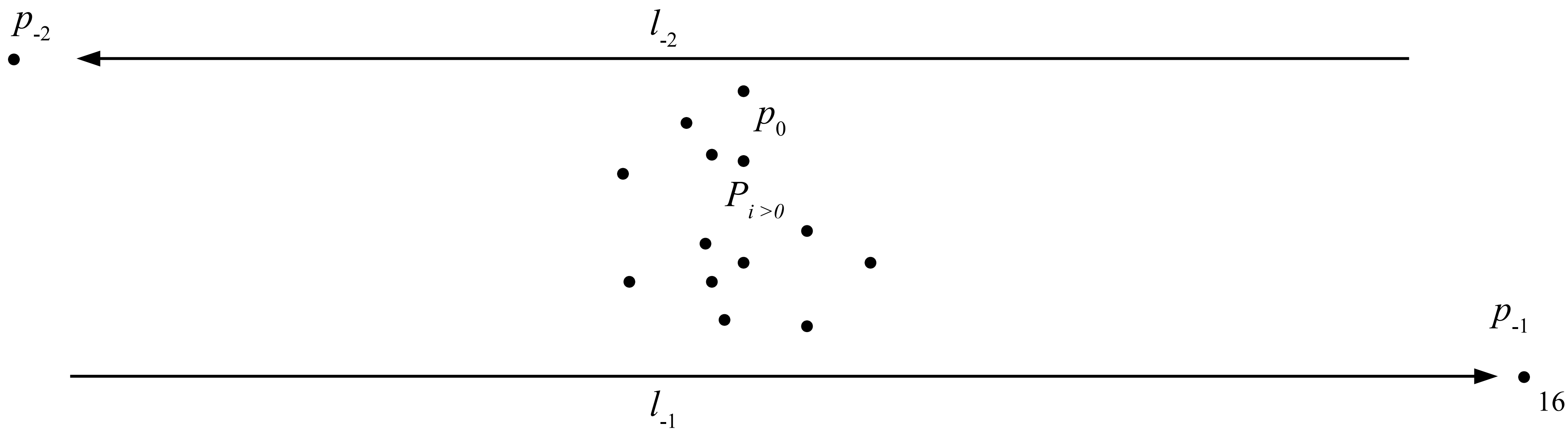
- What about initialization ?
  - In fact, one starts with a trivial triangulation (one triangle !). The initial triangle is made of one of the extremal vertices of the convex hull of the points to be inserted, and two fictive vertices
    - The fictive vertices are located far enough so that they are not in any of the circumscribed circles of the final triangulation
    - However, for numerical reasons, they cannot be set as far as one would like



CAD & Computational Geometry

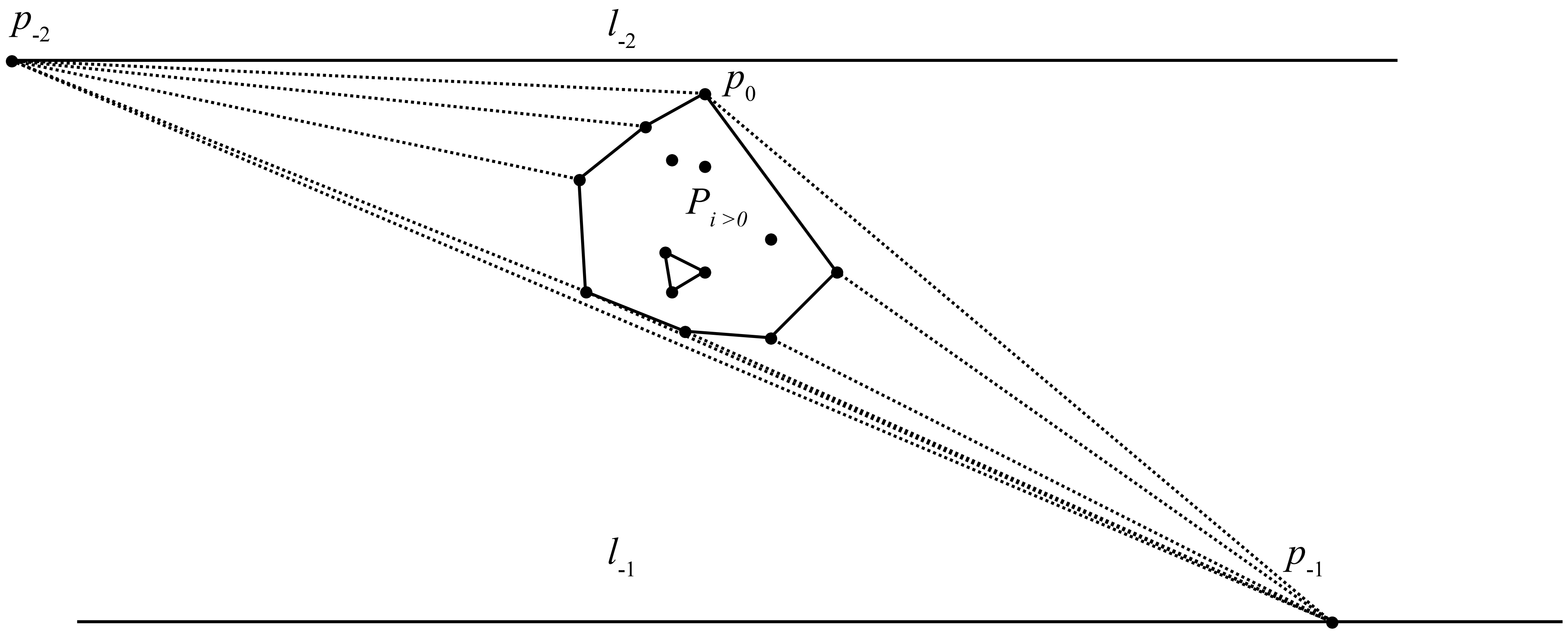
Delaunay Triangulation

- In fact, we won't assign coordinates to p_{-1} and p_{-2}, instead we will modify every predicate in which they are involved so the the behavior is as if those were located at the infinite.
    - Let's assume the following ordering : p is above q if $y_p > y_q$ or if $y_p = y_q$ and $x_q > x_p$.
    - Let $l_{-1}$and l_{-2} be two horizontal lines (see drawing). p_{-1} is on l_{-1} and such that the ordering as stated above is the same as the one induced by a clockwise ordering around p_{-1}. p_{-2} is on l_{-2} such that the ordering as stated above is the same as the one induced by a counterclockwise ordering around p_{-2} – for every point p_i AND p_{-1}.



Delaunay Triangulation

- The Delaunay triangulation of $\{p_{-2}, p_{-1}, p_0 \ldots p_n\}$ is the triangulation of $\{p_0 \ldots p_n\}$ with additional edges joining the right side of the convex hull with p_{-1}, the left side to p_{-2} and $p_{-2}p_{-1}$.



- Once the triangulation has been computed, it is easy to take out every triangle connected to p_{-2} and/or p_{-1}.

Delaunay Triangulation

- Modifications to the predicate of legality
  - Let $p_i p_j$ an edge for which the legality is to be tested. The apex points on neighboring triangles are p_k and p_l .

1 – If $p_i p_j$ is an edge of the triangle $p_0 p_{-1} p_{-2}$, then it is legal.

2 – If the indexes i,j,k,l are all positive : classical test

3 – In all the other cases : $p_i p_j$ is legal iff $\min(k,l)<\min(i,j)$.
Explanation :

3a – If only one of i,j,k,l is negative, then the edge joining the two points with positive indexes is the only legal one, either ij or kl.

3b – If two indexes i,j,k,l are negative, then only those are necessarily shared among (i,j) and (k,l) (otherwise it is case 1, note than in every case the index $r>0$ of the inserted point belongs to i,j,k,l)
Then, the legal edge is the one containing p_{-1} (this is obvious because we have decided that p_{-2} is located such that no circumscribed circle may contain it, including triangles formed with p_{-1})

CAD & Computational Geometry

Delaunay Triangulation

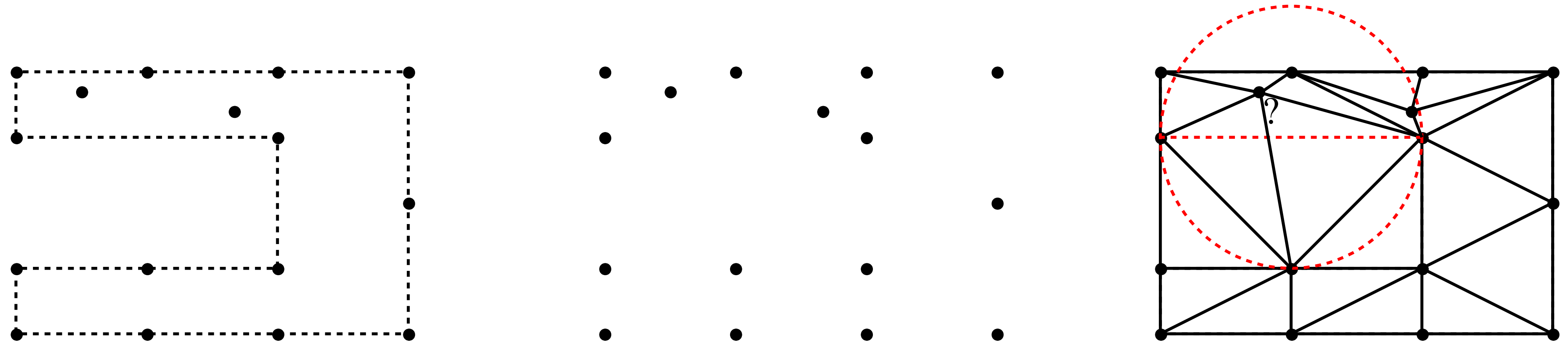
- Performance of the algorithm is in $n \log n$ (Optimal !)
  - This depends on the fact that the lookout of the triangle containing the point to insert is in $\log n$ (see next course)

CAD & Computational Geometry

Constrained Delaunay Triangulation

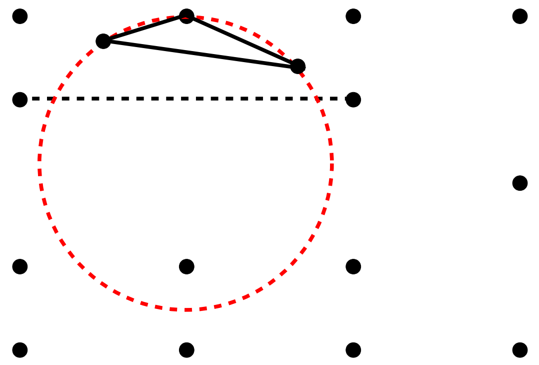
- The previous algorithm gives us a triangulation of the convex hull of the set of points.

  One often have a discretization of the boundaries of a polygon, possibly non convex, and possibly containing holes.
  - Hence, the Delaunay triangulation may miss some of the boundary edges. We must therefore allow them to appear in the final triangulation, although they are not necessarily legal.



Constrained Delaunay Triangulation

- A triangulation in which some edges are constrained is a constrained triangulation.
  - If it respects the "constrained" empty sphere criterion, in the meaning that a sphere circumscribed to any triangle $p_i p_j p_k$ does not contain any visible point from either p_i, p_j and p_k, then it is a Constrained Delaunay Triangulation.

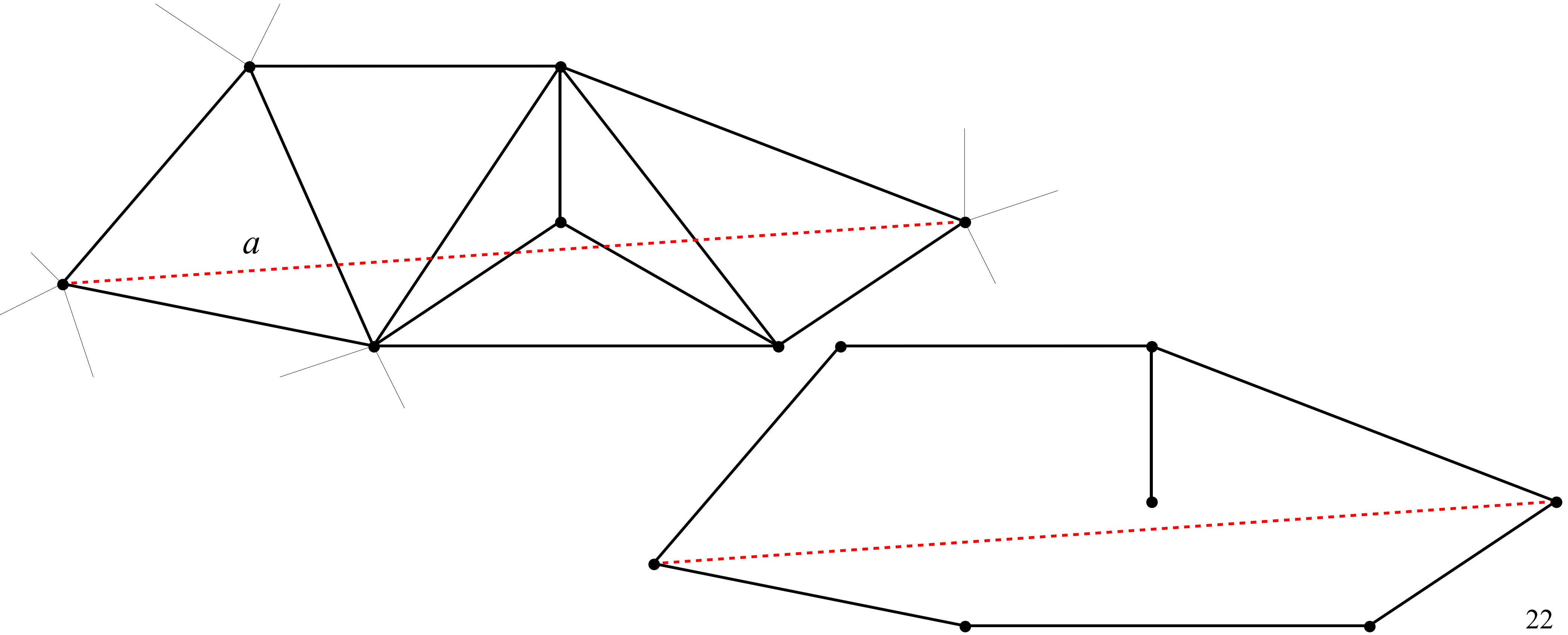


  - Definition of the visibility : two points a and b are visible if the segment ab does not intersect any constrained edge..

Constrained Delaunay Triangulation

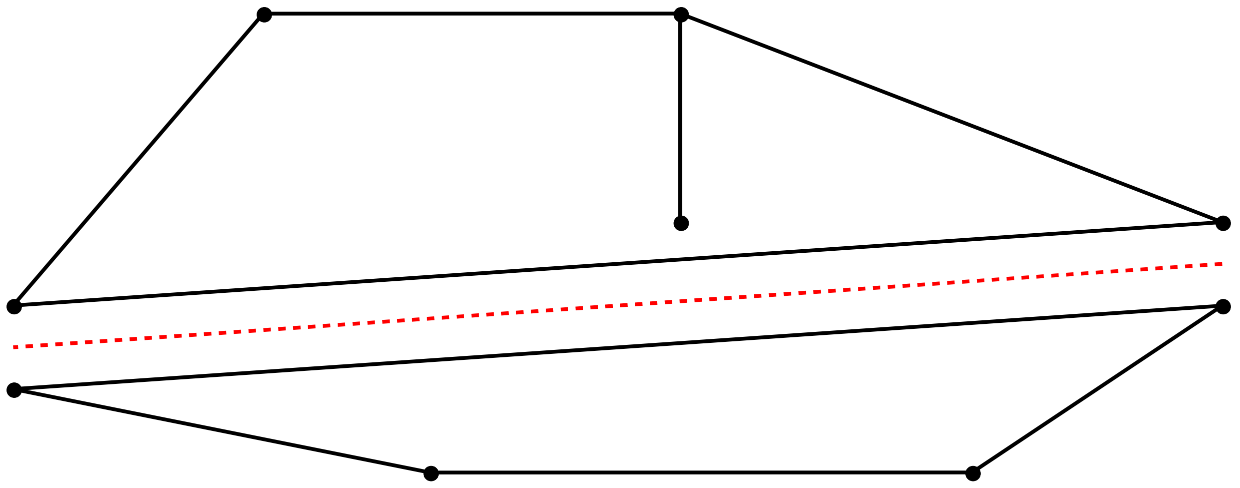
- How to constraint a Delaunay triangulation ?
  - Algorithm :

    Look for the set of triangles intersecting the constrained edge a, delete them, keep the pseudo-polygon that bound the void that has just been created.



Constrained Delaunay Triangulation

- Cut the pseudo-polygon in two by edge a. That edge will obviously be part of the convex hull of each of the parts (independently)

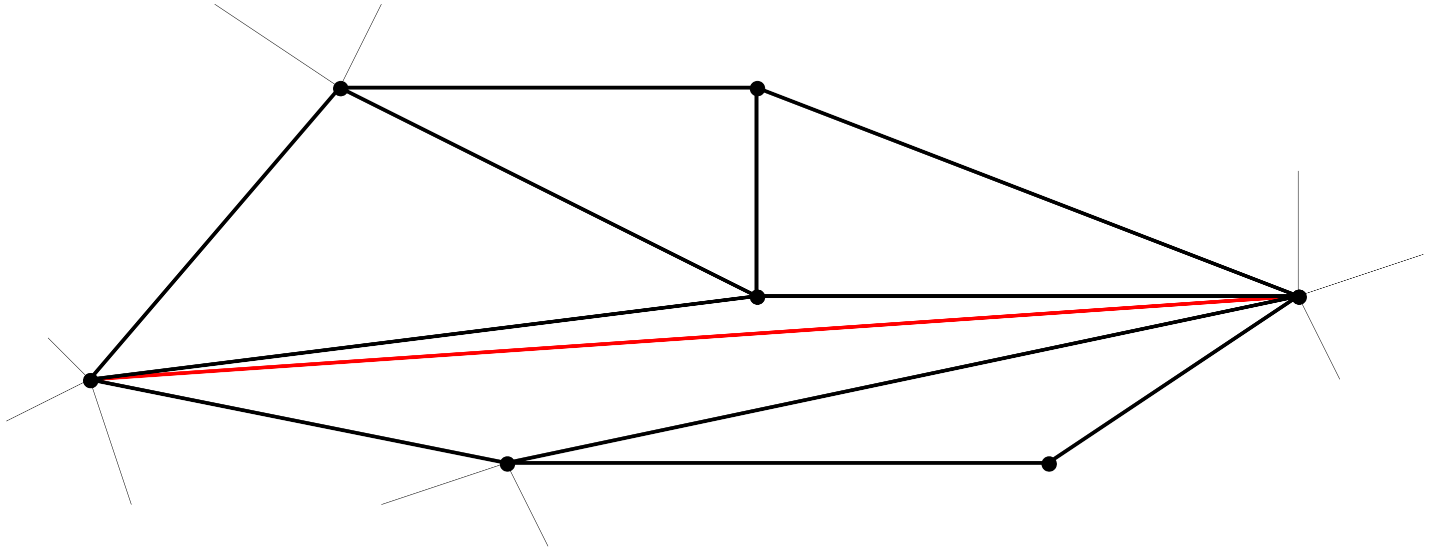


- Notice that the edges of both polygons are necessarily legal, and belong the the Delaunay triangulation.
  - Proof : those were legal before, and some vertices were taken out, so the edge remain part of the Delaunay graph of the subset of vertices taken out from the polygon. Moreover, edge a belongs also to the delaunay graph, because it is part of the convex hull.

CAD & Computational Geometry

Constrained Delaunay Triangulation

- Triangulate each sub-polygon separately, and merge them back to the original triangulation.



- Note that the pseudo-polygons are not convex and it is possible that “exterior” triangles appear. Those should be eliminated appropriately.

Mesh Generation

- Mesh generation
  - Let’s suppose we have made the triangulation of a given contour

    The aim here is to generate additional vertices so that the final triangulation respects at least two geometrical criteria :
    - Size of the triangles
    - Shape thereof
  - Where to insert these additional vertices in the triangulation ?
    - Idea : take advantage that we have a valid (possibly constrained) Delaunay triangulation at step n-1 .
    - One may use a measure of the geometrical “quality” of edges or triangles, and insert a new vertex at the right place to make this measure better, at least locally
    - At each step, the local connectivity is recomputed, using e.g. Lawson’s algorithm.
    - Some edges cannot be violated (imposed contours), thus Lawson’s algorithm must be modified so as to prevent swapping these edges.

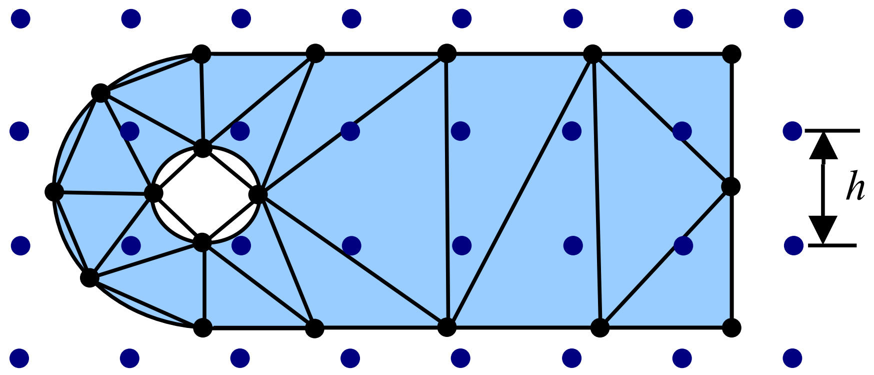
Mesh Generation

- Swap an edge and make it legal (with constraints)

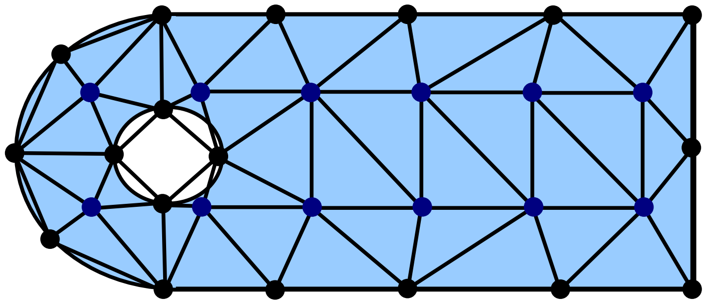
```
LegalizeEdge($p_r$, $p_ip_j$, $T$)
{
 If ($p_ip_j$) is an illegal edge & is not a constrained edge
 {
   Let $p_ip_jp_k$ the triangle adjacent to $p_rp_ip_j$ through edge $p_ip_j$
   Flip $p_ip_j$ and replace it by $p_rp_k$
   LegalizeEdge($p_r$, $p_ip_k$, $T$)
   LegalizeEdge($p_r$, $p_kp_j$, $T$)
 }
}
```

Node placement & Mesh generation

- Point placement strategy
  - Points are selected on a regular grid



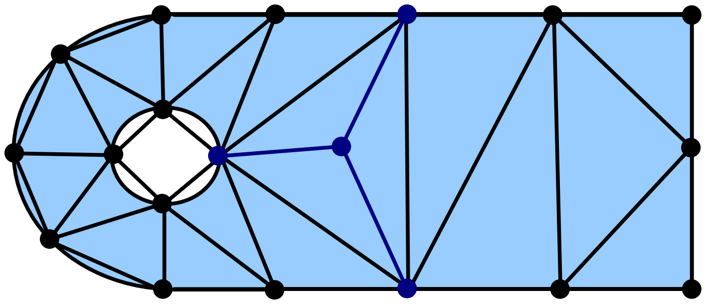
  - Vertices falling outside the triangulation are ignored.

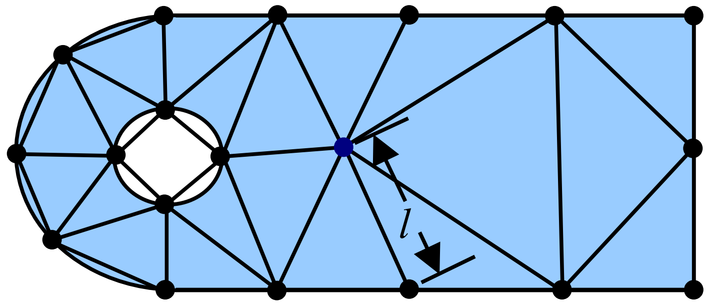


Node placement & Mesh generation

- Points are at centroïds of triangles

  One inserts new vertices until every edge has an acceptable length

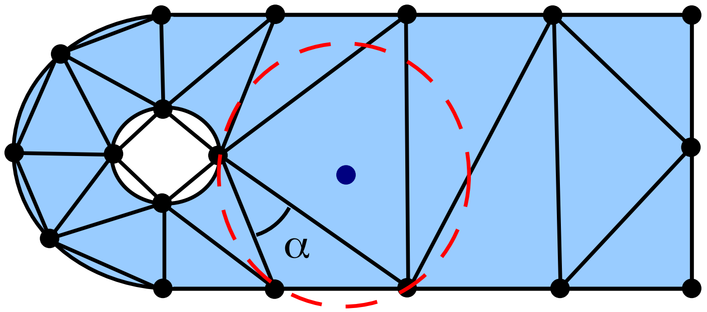




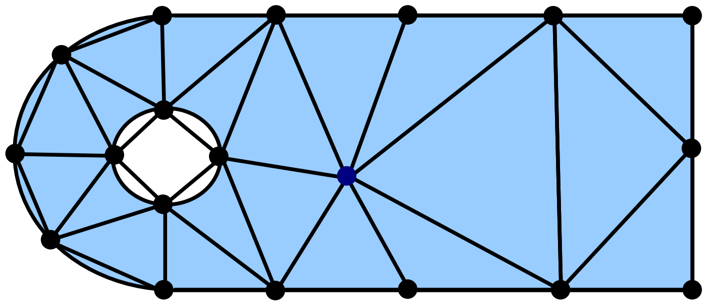
Node placement & Mesh generation

- Center of the circumscribed circle (P. Chew, J. Ruppert, J. Shewchuk)

  The order in which insertions are made is based on the minimal angle of a triangle, and one inserts new vertices until the minimal angle is above a certain threshold value (30° in principle)



- It is theoretically proven that it terminates for any threshold angle below 20.7 ° (cf J. Ruppert, 1995)
  In practice, works until
  ~ 33.8° (cf J. Shewchuk, 1996)



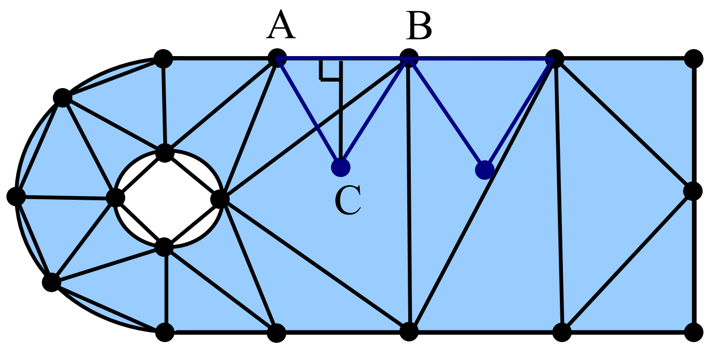
Jim Ruppert, *A Delaunay Refinement Algorithm for quality 2-Dimensional Mesh Generation*
Journal of Algorithms 18(3):548-585,1995

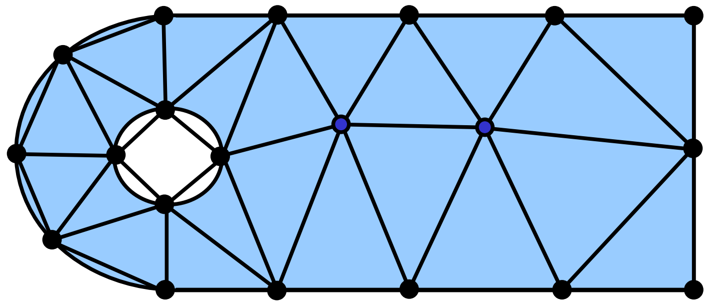
CAD & Computational Geometry

Node placement & Mesh generation

- By frontal advance (D.L. Marcum)

  Vertices are inserted from the boundaries, while maintaining a structure for an advancing front. The new vertices are located ideally, until the two front merge...

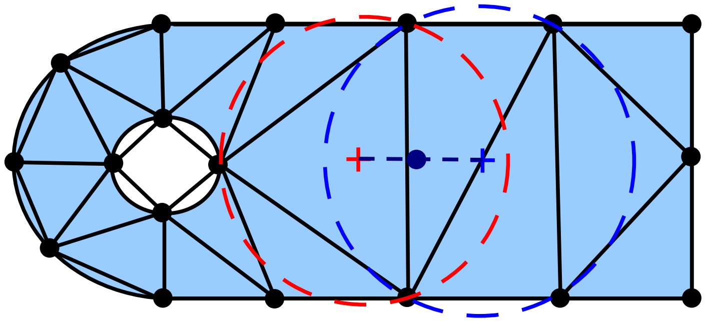


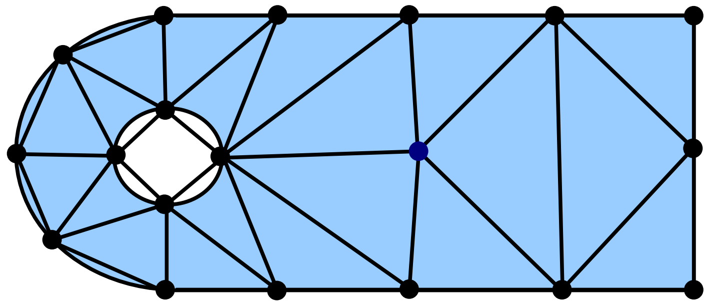


Marcum, D.L., and Weatherill, N.P., *A Procedure for Efficient Generation of Solution Adapted Unstructured Grids*, Computer Methods in Applied Mechanics and Engineering, Vol. 127, p. 259, 1995.

Node placement & Mesh generation

- Middle of a segment of the Voronoï diagram that links the center of the circumscribed circles of two triangles sharing an edge. (S. Rebay, 1993)



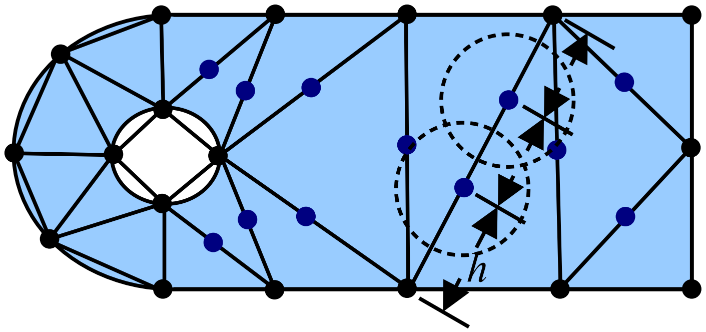


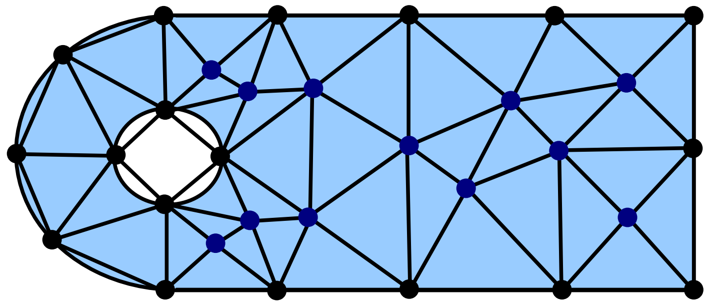
S. Rebay. *Efficient unstructured mesh generation by means of Delaunay triangulation and Bowyer-Watson algorithm*. Journal of Computational Physics, 106:25–138, 1993.

CAD & Computational Geometry

Node placement & Mesh generation

- Along existing edges – one should check that neighboring edges are not too close
  (P.L. George, 1991)



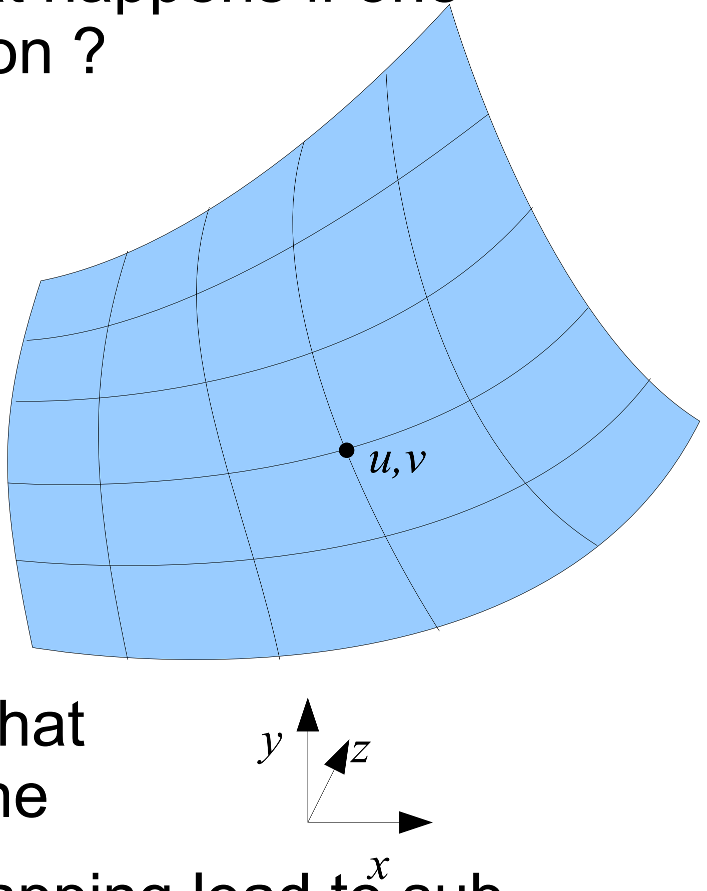


P.L.George, F.Hecht and E.Saltel. *Automatic mesh generator with specified boundary*. Computer Methods in Applied Mechanics and Engineering, 92:269–288, 1991.

Surface Delaunay Triangulation

- Instead of working in the 2D plane, what happens if one want to perform *surface* mesh generation ?

$$\vec{P}(u,v)=\begin{cases} x=f(u,v) \\ y=g(u,v) \\ z=h(u,v) \end{cases}$$



- It is possible to work in the parametric space of the surface, hence reuse all what has been said for the euclidean 2D plane
- However, deformations in the above mapping lead to sub-optimal results.
  - Impossible to control the size and shape of triangles – it will ultimately depend on the definition of the surface

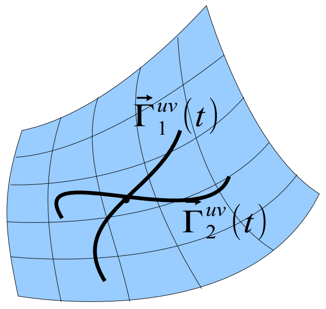
CAD & Computational Geometry

Surface Delaunay Triangulation

- We have to use the first fundamental form (metric tensor) of the surface, which allows us to measure real angles, lengths and areas using variables in the parametric space.

$$\vec{\Gamma}_1^{uv}(t):\begin{cases} u=u_1(t) \\ v=v_1(t) \end{cases} \qquad \phi_1(d\Gamma_1^{uv}, d\Gamma_2^{uv}) = \begin{pmatrix} du_1 & dv_1 \end{pmatrix} \begin{pmatrix} e & f \\ f & g \end{pmatrix} \begin{pmatrix} du_2 \\ dv_2 \end{pmatrix} = \begin{pmatrix} du_1 & dv_1 \end{pmatrix} \mathrm{M}_1 \begin{pmatrix} du_2 \\ dv_2 \end{pmatrix}$$

$$\vec{\Gamma}_2^{uv}(t):\begin{cases} u=u_2(t) \\ v=v_2(t) \end{cases} \qquad \text{with } e=P^u\cdot P^u \ , \ f=P^u\cdot P^v \ , \ g=P^v\cdot P^v , \ du_1=\frac{\partial u_1(t)}{\partial t}, dv_1=\frac{\partial v_1(t)}{\partial t} \ \ldots$$



$$L=\int_a^b \sqrt{\phi_1(d\Gamma^{uv}, d\Gamma^{uv})}\,dt \qquad A=\iint_D \sqrt{\det \mathrm{M}_1}\,du\,dv$$

$$\cos\alpha = \frac{\phi_1(d\Gamma_1^{uv}, d\Gamma_2^{uv})}{\sqrt{\phi_1(d\Gamma_1^{uv}, d\Gamma_1^{uv})\phi_1(d\Gamma_2^{uv}, d\Gamma_2^{uv})}}$$

Surface Delaunay Triangulation

- Back to our problem !!!
  - In our case, distances have to be measured along e.g. edges.

    These segments are defined in the (u,v) parametric space, and have a parametrization (e.g. variable t taking values between 0 and 1)

    $$a^s(t)=(1-t)\cdot p_0^s+t\cdot p_1^s$$

    The length is therefore, using the first fundamental form:

    $$L=\int_a^b\sqrt{\begin{pmatrix}u'(t) & v'(t)\end{pmatrix}\begin{pmatrix}e & f\\ f & g\end{pmatrix}\begin{pmatrix}u'(t)\\ v'(t)\end{pmatrix}}\,dt$$

    $$L(a)=\int_0^1\sqrt{\left(p_1^s-p_0^s\right)^{\mathrm{T}}\begin{pmatrix}e & f\\ f & g\end{pmatrix}\left(p_1^s-p_0^s\right)}\,dt$$
  - If the metric tensor is deemed constant, the integral is easy to compute and the result is :

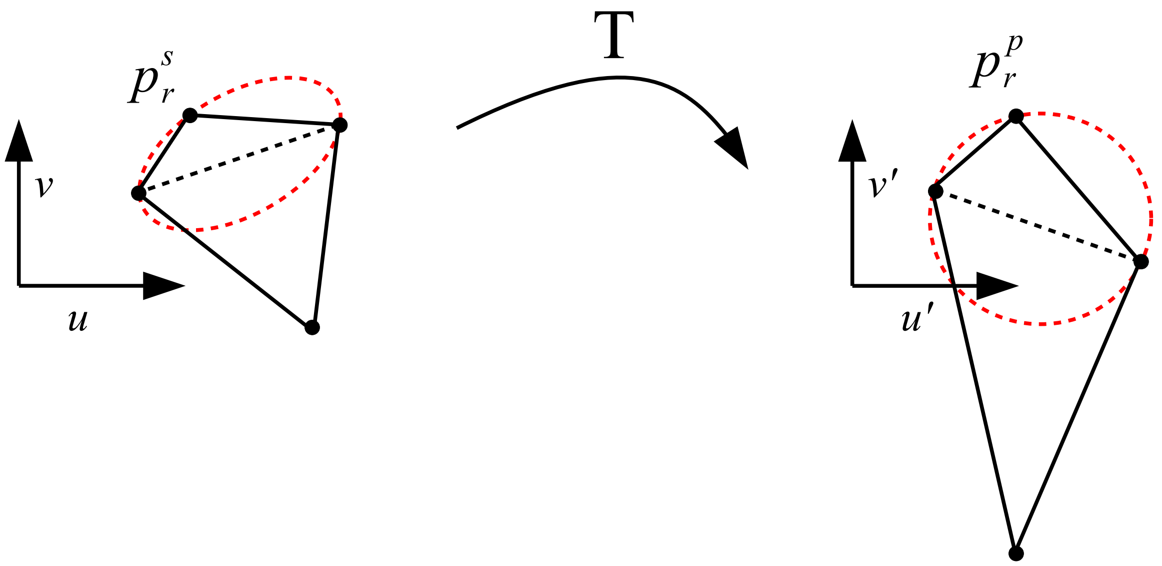
    $$L(a)=\sqrt{\left(p_1^s-p_0^s\right)^{\mathrm{T}}\begin{pmatrix}e & f\\ f & g\end{pmatrix}\left(p_1^s-p_0^s\right)}$$

Surface Delaunay Triangulation

- The modified edge legality check

  Amounts to determine the shape of the locus of points situated at a constant distance from a given point (the “center”), and going through three given points

  - In the general case, it is very costly, because the metric tensor is not constant.
  - If the metric tensor is constant, then the locus is an ellipse...
- Let us perform a change of variable T (which consists of a rotation followed by a scaling) to bring everything back in a planar coordinate system where the measures are euclidean...



Surface Delaunay Triangulation

- Definition of transformation T

$$L(a)=\sqrt{\left(p_1^s-p_0^s\right)^{\mathrm{T}}\begin{pmatrix} e & f \\ f & g \end{pmatrix}\left(p_1^s-p_0^s\right)}=\sqrt{\left(p_1^s-p_0^s\right)^{\mathrm{T}}\mathrm{M}_1\left(p_1^s-p_0^s\right)}$$

We expect such a distance measure to take place in a local euclidean space. In this space, the coordinates are p_0^p and p_1^p and the metric tensor is the identity I.

$$L(a)=\sqrt{\left(p_1^s-p_0^s\right)^{\mathrm{T}}\mathrm{M}_1\left(p_1^s-p_0^s\right)}=\sqrt{\left(p_1^p-p_0^p\right)^{\mathrm{T}}\mathrm{I}\left(p_1^p-p_0^p\right)}$$

Let us set $\mathrm{M}_1=\mathrm{J}^{\mathrm{T}}\cdot\mathrm{J}$, then we have

$$L(a)=\sqrt{\left(p_1^s-p_0^s\right)^{\mathrm{T}}\mathrm{J}^T\cdot\mathrm{J}\left(p_1^s-p_0^s\right)}=\sqrt{\left(p_1^s-p_0^s\right)^{\mathrm{T}}\mathrm{J}^T\cdot\mathrm{I}\cdot\mathrm{J}\left(p_1^s-p_0^s\right)}$$

$$=\sqrt{\left(\mathrm{J}\cdot p_1^s-\mathrm{J}\cdot p_0^s\right)^{\mathrm{T}}\mathrm{I}\left(\mathrm{J}\cdot p_1^s-\mathrm{J}\cdot p_0^s\right)}=\sqrt{\left(p_1^p-p_0^p\right)^{\mathrm{T}}\mathrm{I}\left(p_1^p-p_0^p\right)}$$

Therefore $p_0^p=\mathrm{J}\,p_0^s$ and $p_1^p=\mathrm{J}\,p_1^s$. Here, J is a transformation matrix of T, or **Jacobian matrix**.

CAD & Computational Geometry

Surface Delaunay Triangulation

- How to obtain J ?
  J has 4 independent scalars, however, M_1 is a symmetric matrix and therefore has only 3 degrees of freedom. The system $\mathrm{M}_1 = \mathrm{J}^T \cdot \mathrm{J}$ is therefore under-determined and has an infinite number of solutions. One need to find one suitable value for J.
- There exists one natural decomposition of a symmetric matrix M into the product of two transposed matrices, it is the Cholesky decomposition, which always exist for positive definite matrices : $\mathrm{M}=\mathrm{S}^\mathrm{T}.\mathrm{S}$ with S = a upper triangular matrix.

$$\mathrm{M}_1 = \begin{pmatrix} e & f \\ f & g \end{pmatrix} = S^T \cdot S = \begin{pmatrix} s_1 & 0 \\ s_3 & s_2 \end{pmatrix} \cdot \begin{pmatrix} s_1 & s_3 \\ 0 & s_2 \end{pmatrix} = \begin{pmatrix} s_1^2 & s_1 s_3 \\ s_1 s_3 & s_2^2 \end{pmatrix}$$

- We have therefore : $\mathrm{S} = \begin{pmatrix} \sqrt{e} & \dfrac{f}{\sqrt{e}} \\ 0 & \sqrt{g} \end{pmatrix} = \mathrm{J}$
  - To determine if a circle contains a given point, we will use this matrix J to transform the coordinates of every point (4 in total). It is advisable to bring back the coordinates as variations around a given reference, for instance the inserted point p_r (which becomes the origin of the local frame)

CAD & Computational Geometry

Surface Delaunay Triangulation

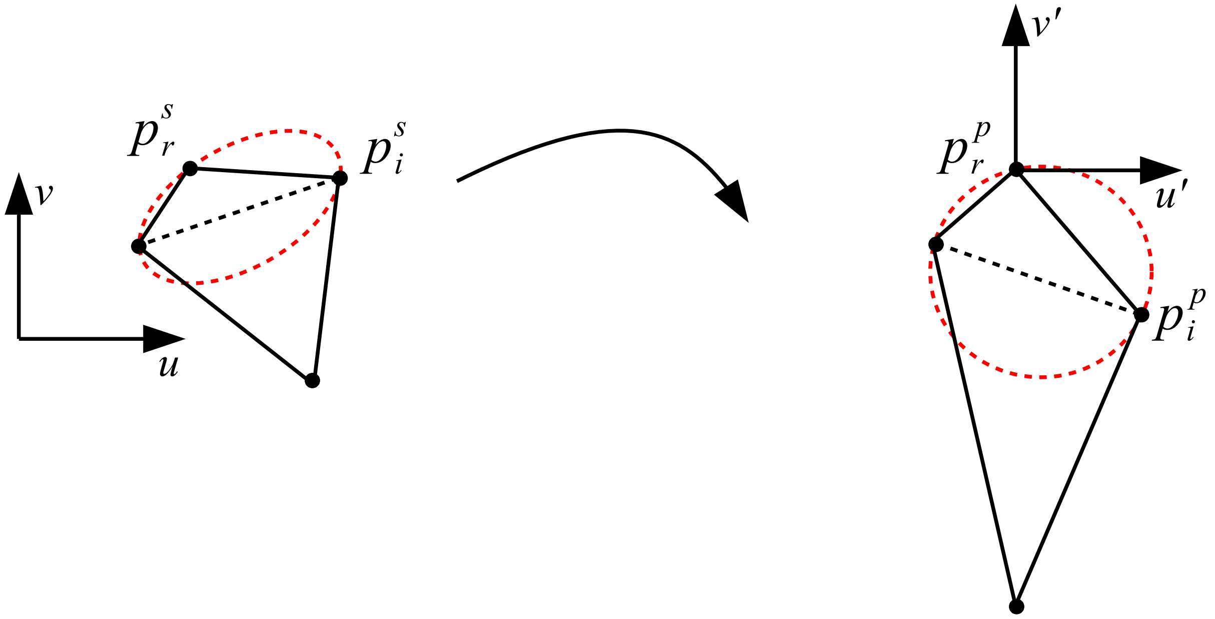
- Finally,

$$J = \begin{pmatrix} \sqrt{e} & \dfrac{f}{\sqrt{e}} \\ 0 & \sqrt{g} \end{pmatrix} \text{ with } e = P^u \cdot P^u \ , \ f = P^u \cdot P^v \ , \ g = P^v \cdot P^v$$

New coordinates :

$$p_i^p = J \cdot \left(p_i^s - p_r^s \right)$$

In this frame, the legality check is the same as in an euclidean 2D frame.



v
u
p_r^s
p_i^s
v'
u'
p_r^p
p_i^p