Lyon 1

Mesh and Computational Geometry

Raphaëlle Chaine
Université Claude Bernard Lyon 1

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

2024-25

1

Incremental Delaunay Algorithm

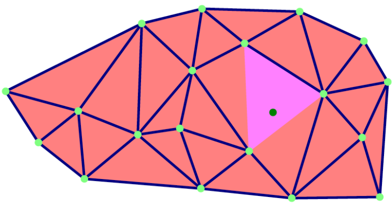
- Complexity analysis
- In average :
  - Complexity dependent on the strategy used to locate the triangle containing the point to be inserted

163

163

Location strategies

- Exhaustive search among all the triangles



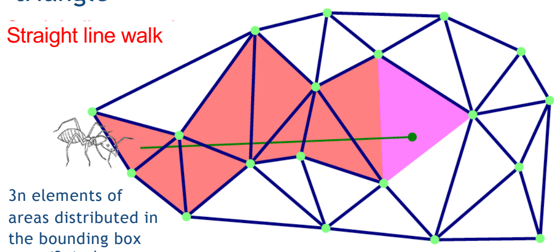
$O(n)$ O. Devillers

164

164

Location strategies

- Search from a first randomly selected triangle



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

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

165

165

Location strategies

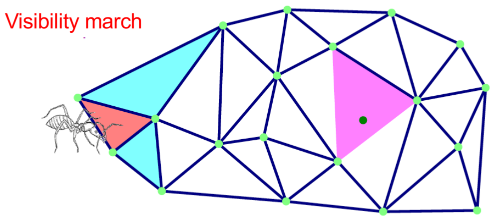
- Search from a first randomly selected triangle
  - Straight walk
  - Requires a predicate of segments intersection

166

166

Location strategies

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

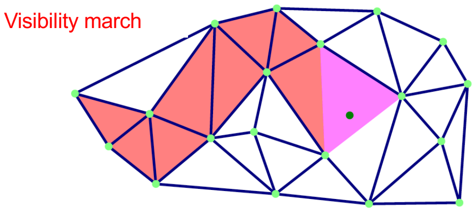


167

167

Location strategies

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



168

168

Location strategies

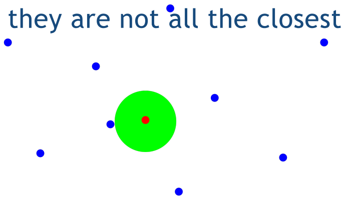
- Search from a first randomly selected triangle
  - Visibility walking
  - Only requires an orientation predicate to find the next triangle to walk in

169

169

Delaunay and proximity in space

- Delaunay triangulation allows to model the notion of proximity between points
- Each point is thus connected to nearby points around it
- Be careful, they are not all the closest!

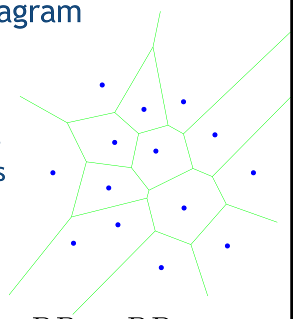


170

170

Voronoi diagram

- Voronoi's cell of a site P_i is the set of points closer to this site than to other sites



$$V_i = \{P \in \mathbb{R}^k \ t.\ que\ PP_i < PP_j \ pour\ tout\ j \neq i\}$$

171

171

Voronoi diagram

- Given a set E of points in $\mathbb{R}^k$, the partitioning of $\mathbb{R}^k$ into cells composed of points having the same nearest neighbour in E is called a Voronoi diagram of E

172

172

Voronoi diagram

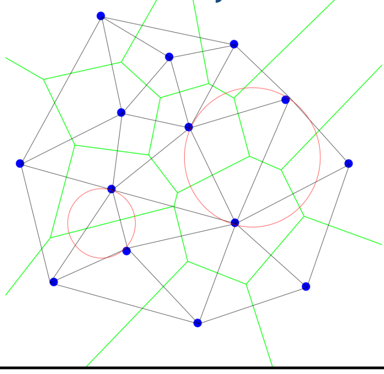
- Possible construction:
  - V_i : intersection of half-spaces h_{ij}^{i} where h_{ij} is the mediator of segment P_iP_j and h_{ij}^{i} is the half-space delimited by h_{ij} containing P_i

In practice we will proceed differently!

173

173

Duality between Voronoi and Delaunay



174

174

Duality between Voronoi and Delaunay

- Each Voronoi vertex is located at the center of the circumscribed circle of a Delaunay triangle
- Two Voronoi vertices are connected if they are associated with adjacent triangles

175

175

Coordinates of the centre of the circle circumscribed to a triangle ABC

- Useful for displaying the Voronoi diagram
- 1st possibility:
  - Write the equation of the mediator for each edge
    ex: For the edge AB, set of points M such that $MA^2=MB^2$
  - Solving a system of 2 equations with 2 unknowns (it is enough to take 2 mediators)
  - Numerically unstable

176

176

Coordinates of the centre of the circle circumscribed to a triangle ABC

- 2nd possibility :
  - Let's consider the angles
    $$\hat{A} = \widehat{CAB}\ \hat{B} = \widehat{ABC}\ \hat{C} = \widehat{BCA}$$
  - Then the barycentric coordinates of the center H of the circumscribed circle with respect to A, B and C are elegantly expressed :

$$\begin{aligned} Barycenter(&A(\tan\hat{B}+\tan\hat{C}),\\ &B(\tan\hat{C}+\tan\hat{A}),\\ &C(\tan\hat{A}+\tan\hat{B})) \end{aligned}$$

177

177

Coordinates of the centre of the circle circumscribed to a triangle ABC

$$H = Barycenter((A, \tan\hat{B}+\tan\hat{C}), (B, \tan\hat{C}+\tan\hat{A}), (C, \tan\hat{A}+\tan\hat{B}))$$

- 2 Reminders :

$$\tan(\widehat{ABC}) = \frac{\sin(\widehat{ABC})}{\cos(\widehat{ABC})} = sign((\overrightarrow{BC}\times\overrightarrow{BA})\cdot\vec{k})\frac{\left\|\overrightarrow{BC}\times\overrightarrow{BA}\right\|}{\overrightarrow{BC}\cdot\overrightarrow{BA}}$$

$$\begin{aligned} &Barycenter((A,\alpha a),(B,\alpha b),(C,\alpha c))\\ &\quad = Barycenter((A,a),(B,b),(C,c)) \end{aligned}$$

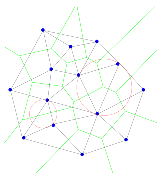
- Ensure to have no more denominators in the expression of your barycentric coordinates (normalization performed afterwards)

178

178

Duality between Voronoi and Delaunay

- Each Delaunay vertex is dual to one Voronoi cell
- Each Delaunay edge is dual to a Voronoi edge
- Each Voronoi vertex is dual to a Delaunay triangle



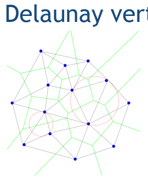
- What about Delaunay, Voronoi and their duality in 3D?

179

179

Duality between Voronoi and Delaunay

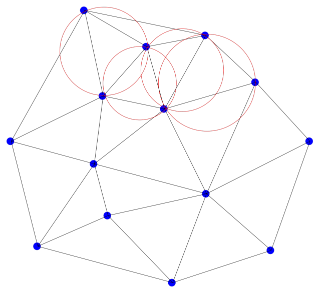
- Which data structure for Voronoi?
  - Walking around a Voronoi face is performed by walking through the faces/edges incident at a Delaunay vertex.
  - To move from one Voronoi cell to an adjacent cell is like moving from a Delaunay vertex to an adjacent vertex.



180

180

Delaunay : Divide and conquer algorithm



181

181

Non incremental Delaunay

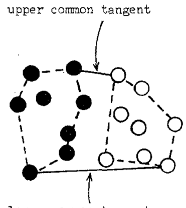
- Divide and Conquer
  - 2D algorithm
    - Split the set of points into 2 subsets `L` and `R` using their median by ascending `x`
    - if the 2 resulting subsets have more than 2 points
      - Delaunay Triangulation `Del(L)` of `L`
      - Delaunay Triangulation `Del(R)` of `R`
      - Merge `Del(L)` and `Del(R)`
        - Removal of some `Del(L)` (resp. `Del(R)`) edges
        - Adding edges joining points of `L` to points of `R`
        - No addition of edges joining points of `L` (resp. `R`)

182

182

Delaunay Fusion

- Determination of the common lower (or upper) tangent
  - Edge joining a vertex of the left (resp. right) convex hull to a vertex of the right (resp. left) convex hull
  - Leaving all the others points above (resp. below)



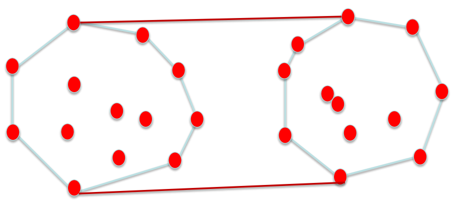
Lee and Schachter

183

183

Delaunay Fusion

- First merge Convex Hulls
  - Find the upper and lower extreme tangents of the 2 polygons

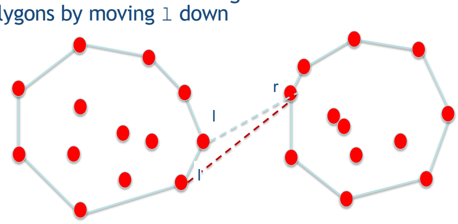


184

184

Convex Hull

- Merger
  - Starting from a segment `lr`
  - Find the lower extreme tangent of the 2 polygons by moving `l` down

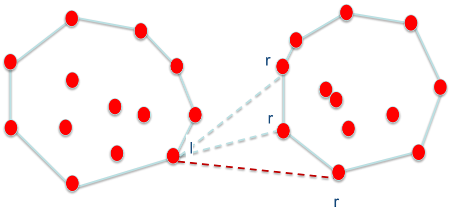


185

185

Convex Hull

- Merger
  - Find the lower extreme tangent of the 2 polygons by moving `l` down, and `r` …

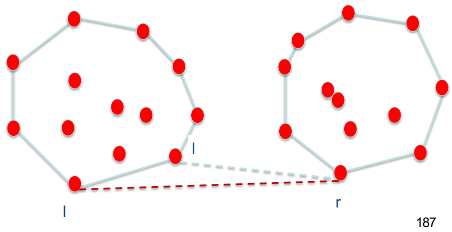


186

186

Convex Hull

- Merger
  - Find the lower extreme tangent of the 2 polygons by moving `l` down, and `r`, and `l`, alternately



187

187

Convex Hull

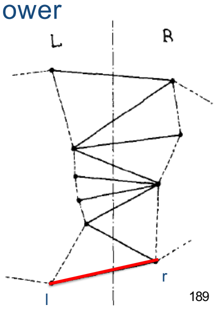
- Merger
  - Find the upper and lower extreme tangents of the 2 polygons
  - Extreme lower tangent
    - Segment `lr` initialized with `l=right(`**`L`**`)` and `r=left(`**`R`**`)`
    - Move `l` down (resp. `r`) to the adjacent vertex positioned lower on **`L`** (resp. **`R`**) as long as it can be seen from `r` (resp. `l`)
    - Repeat alternately on **L** and **R**
- Complexity
  - Fusion in O(n)

188

188

Delaunay Fusion

- Build a sequence of « **LR** edges » starting from the common lower tangent of **`LR`**
- The merge is done by incremental sewing between the two triangulations, until it reaches the upper tangent

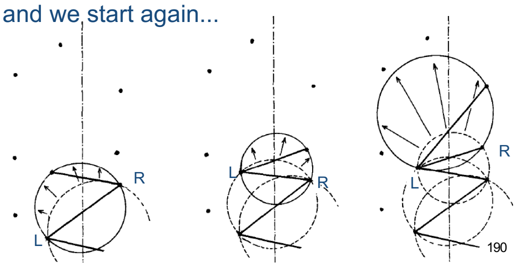


189

189

Delaunay Fusion

- Idea behind the construction of the edge sequence
  - We inflate a ball passing through the vertices of the last LR edge, until we meet a point on **`L`** or **`R`**, and we start again...

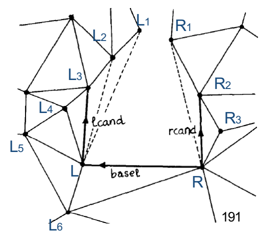


190

190

Delaunay Fusion

- Idea behind the construction of the edge sequence
  - A ball is inflated while remaining centered on the `LR` edge mediator, until it meets a point on **`L`** or **`R`**
  - Depending on the reached point, some edges of `Del(`**`L`**`)` and `Del(`**`R`**`)` will be removed



191

191

Delaunay Fusion

- Construction of the next **LR** edge from a previously constructed `LR` edge
  - Let denote R_1, R_2, R_3... the vertices adjacent to `R` in `del(R)`, clockwise around `R`
  - The vertices L_1, L_2, L_3 adjacent to `L` in `del(L)` counterclokwise around `L`

192

192

Delaunay Fusion

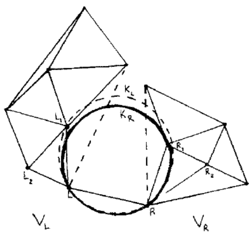
- Construction of the next **LR** edge from a previously constructed `LR` edge
  - Initialization `i=1`
  - As long as there is no RR_i edge to be kept
    - The edge RR_i is removed if `L` conflicts with triangle $RR_{i+1}R_i$ (ie. `L` inside its circumcircle)
  - Symmetric process of edge removal in `Del(L)`
    - An LL_i edge is deleted if `R` conflicts with triangle LL_iL_{i+1}

193

193

Delaunay Fusion

- Suppressed edges in dotted lines



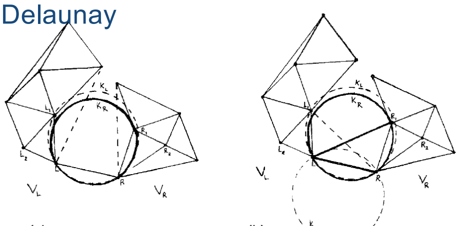
(a)

194

194

Delaunay Fusion

- Once the edges have been deleted, we look which of the two edges RL_i and LR_i is Delaunay



(a) (b)

- And we repeat the process by starting from the chosen edge

195

195

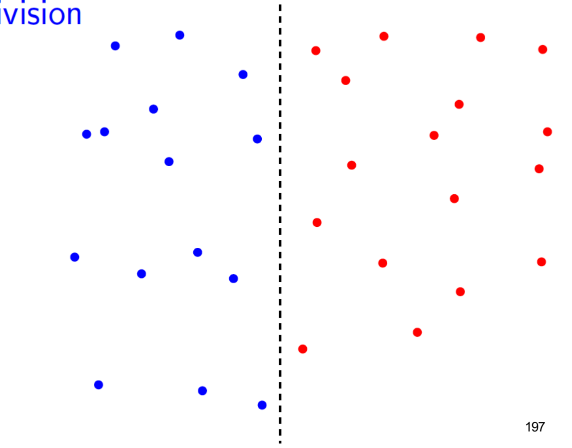
Delaunay Fusion

- Fusion in O(n)
- If the split is well balanced (by using the median):
  algorithm in O(nlog(n))

196

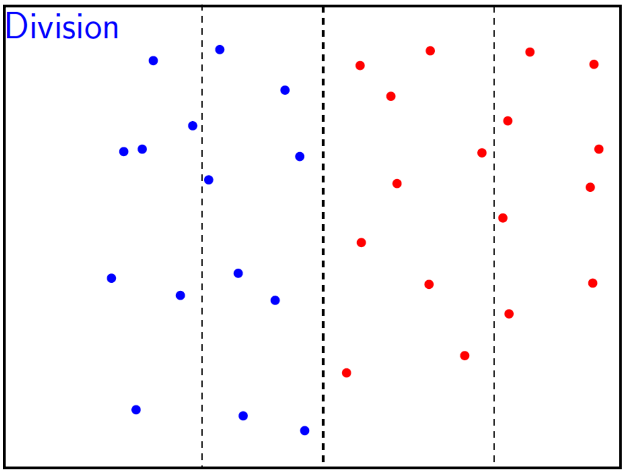
196

Division

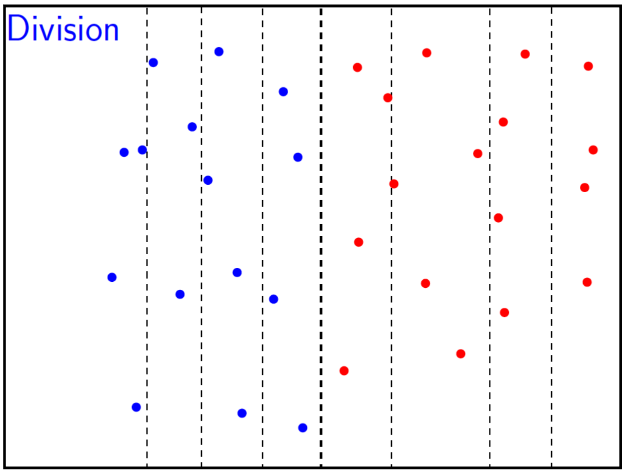


197

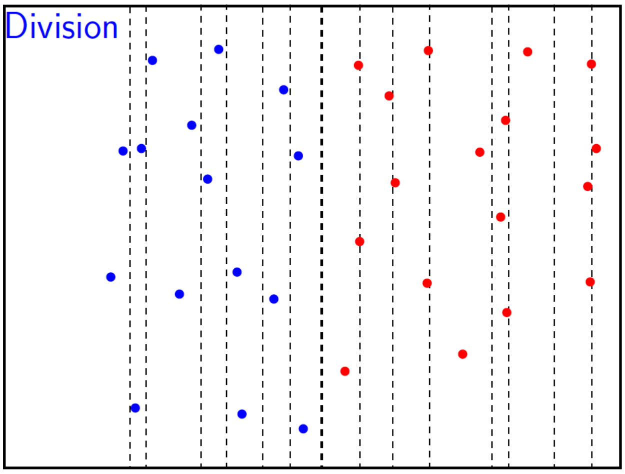
197



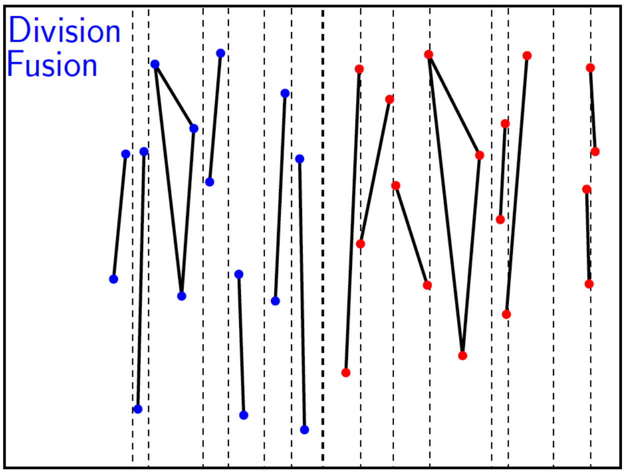
198



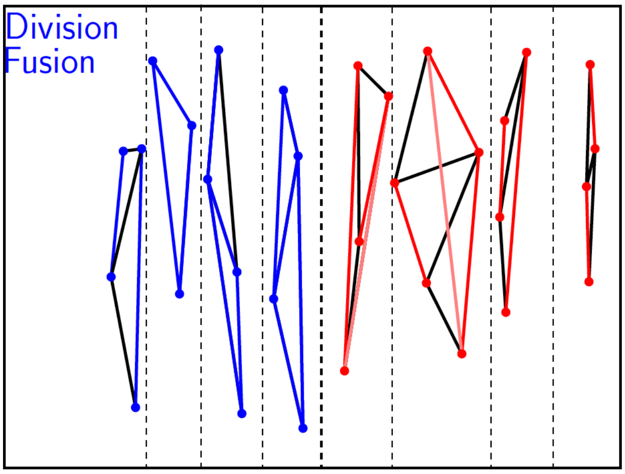
199



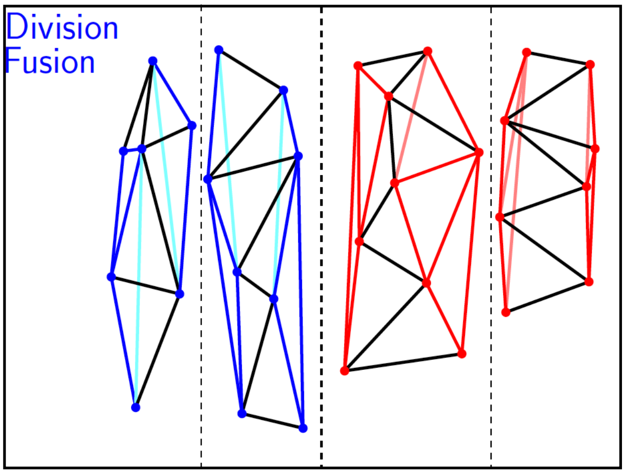
200



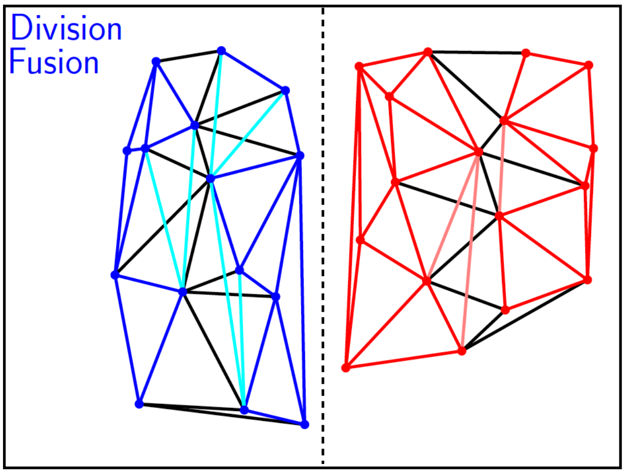
201



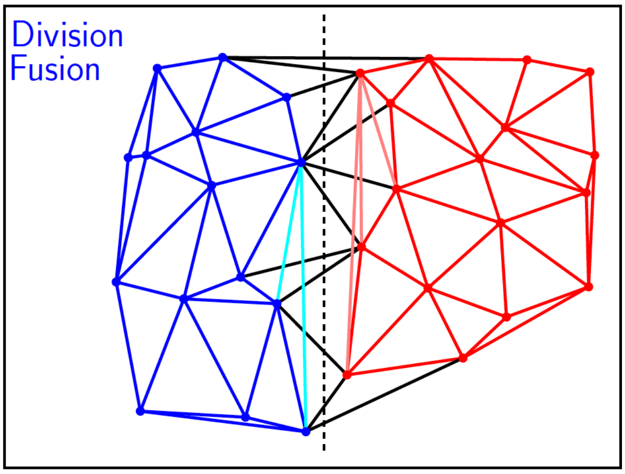
202



203



204



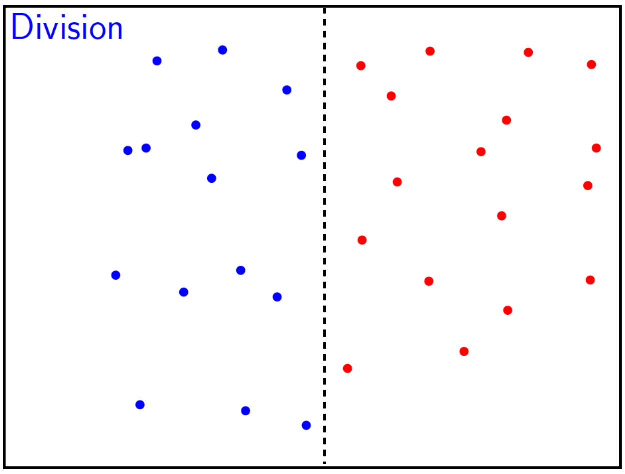
205

Delaunay Divide and Conquer Using a kd-tree

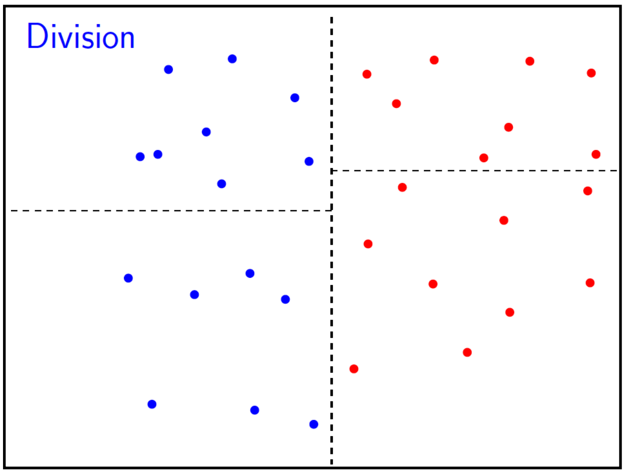
- Easily derecursified algorithm :
  - The construction of the connectivity is only performed at the recursive ascent (« Remontée récursive »)
  - The recursive split can be replaced by a prior sorting
- The split can be performed on x and y alternately using a kd-tree

206

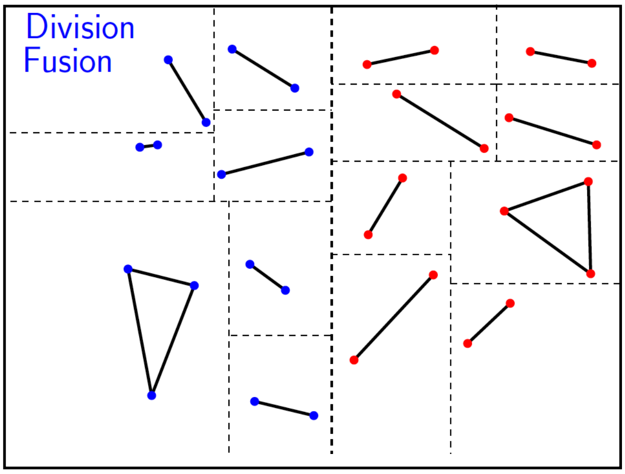
206



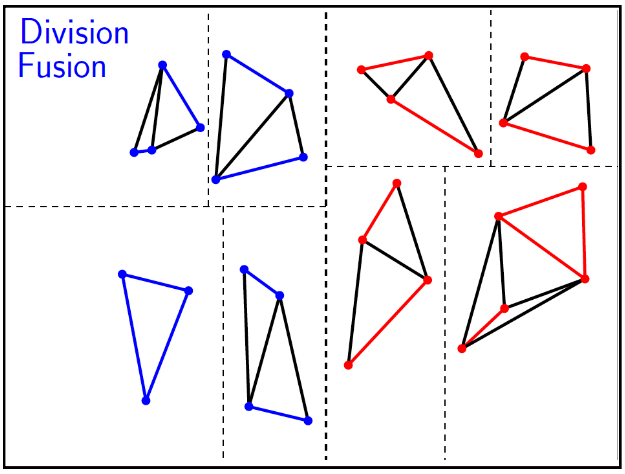
207



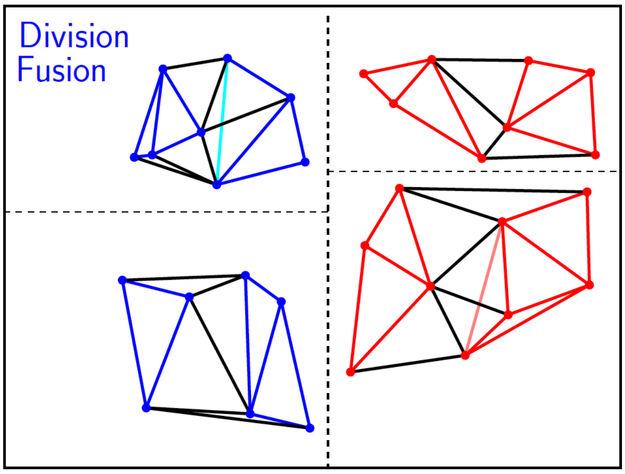
208



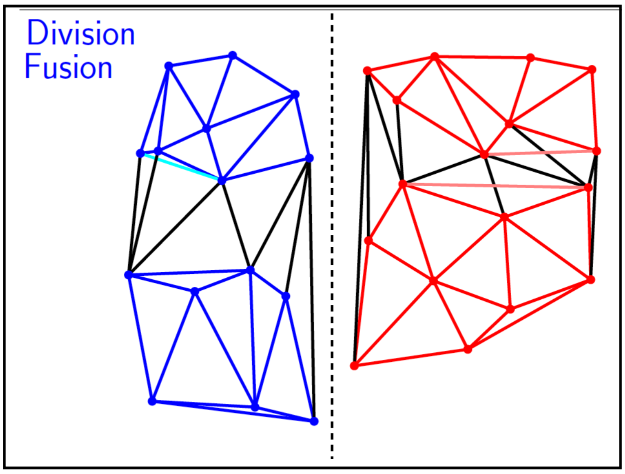
209



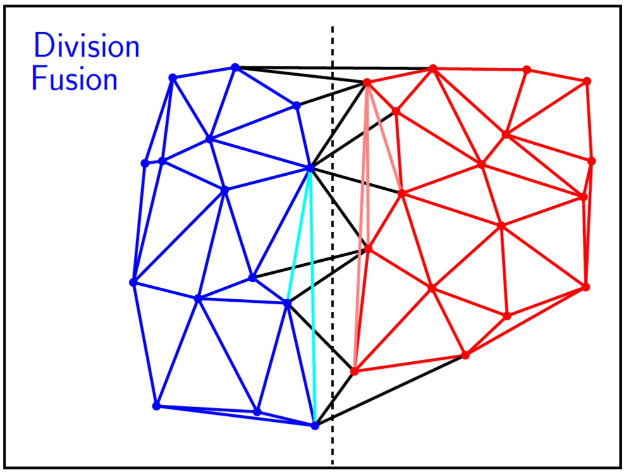
210



211



212



213