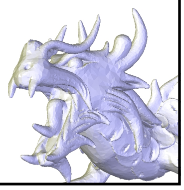
Lyon 1

Mesh and Computational Geometry

Raphaëlle Chaine
Université Claude Bernard Lyon 1

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



1

Non incremental Delaunay

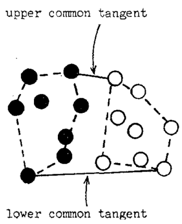
- Divide and Conquer
  - 2D algorithm
    - Split the set of points into 2 subsets `L` and `R` using their median by ascending `x` if the 2 resulting subsets have more than 2 points
    - Delaunay Triangulation `Del(L)` of `L`
    - Delaunay Triangulation `Del(R)` of `R`
    - Merge `Del(L)` and `Del(R)`
      - 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`)

233

233

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)



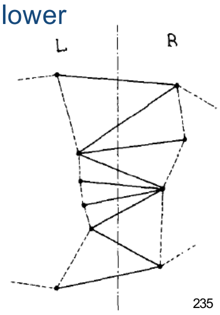
Lee and Schachter

234

234

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

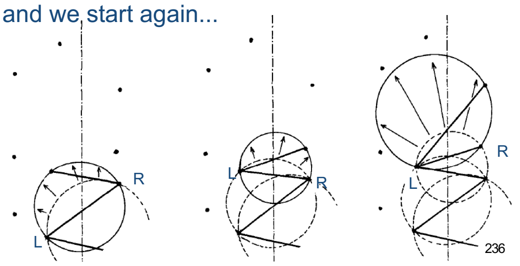


235

235

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...

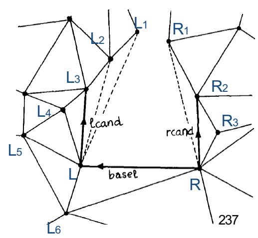


236

236

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



237

237

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)` counterclockwise around `L`

238

238

Delaunay Fusion

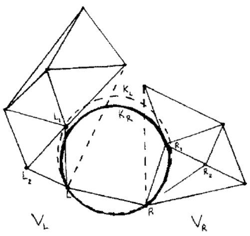
- Construction of the next **LR** edge from a previously constructed `LR` edge
  - Initialization `i=1`
  - As long as there is no RR_i edge to be kept
    - The edge RR_i is removed if `L` conflicts with triangle $RR_{i+1}R_i$ (ie. `L` inside its circumcircle)
  - Symmetric process of edge removal in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

239

239

Delaunay Fusion

- Suppressed edges in dotted lines



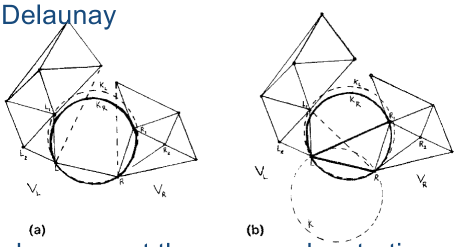
(a)

240

240

Delaunay Fusion

- Once the edges have been deleted, we look which of the two edges RL_1 and LR_1 is Delaunay



(a) (b)

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

241

241

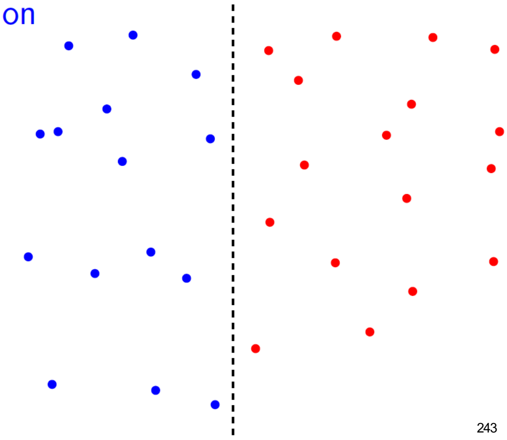
Delaunay Fusion

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

242

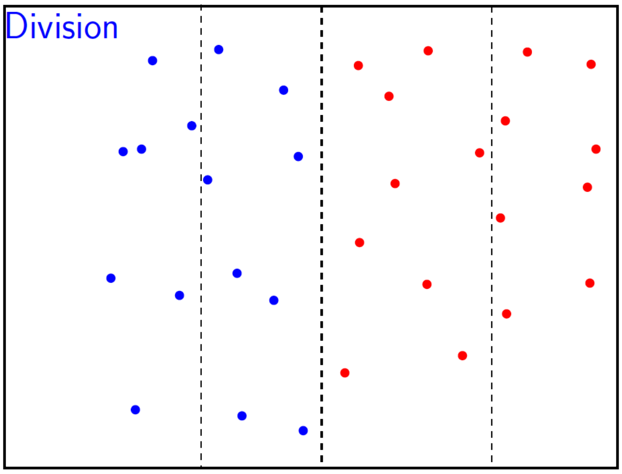
242

Division

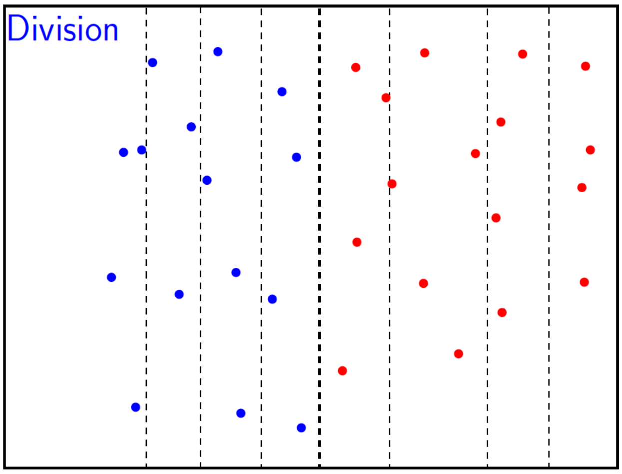


243

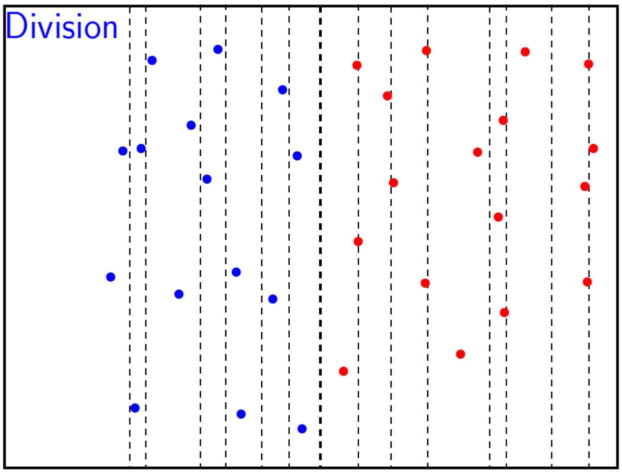
243



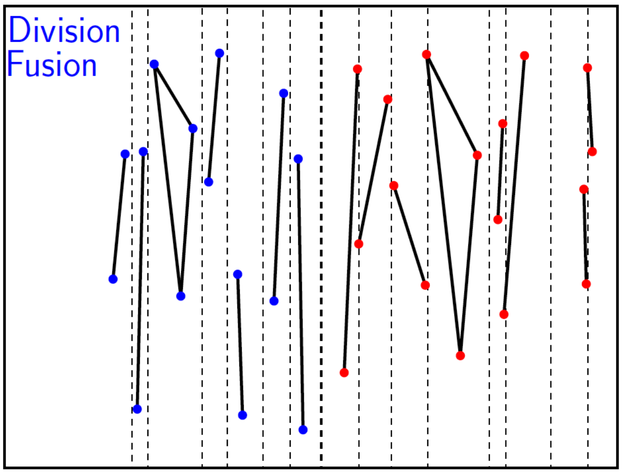
244



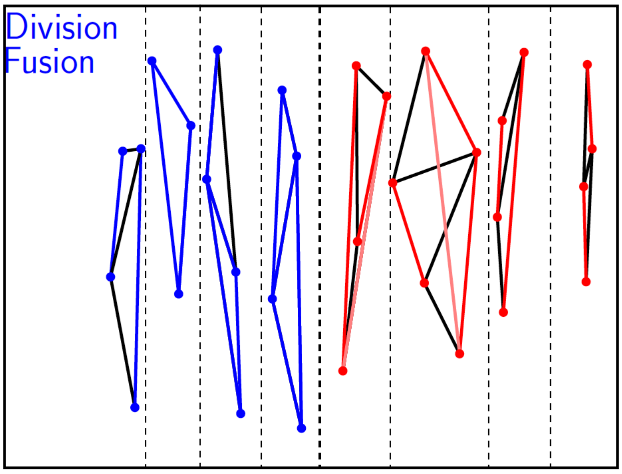
245



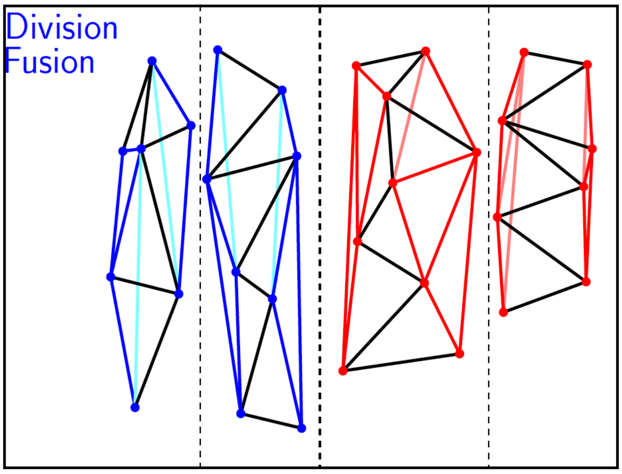
246



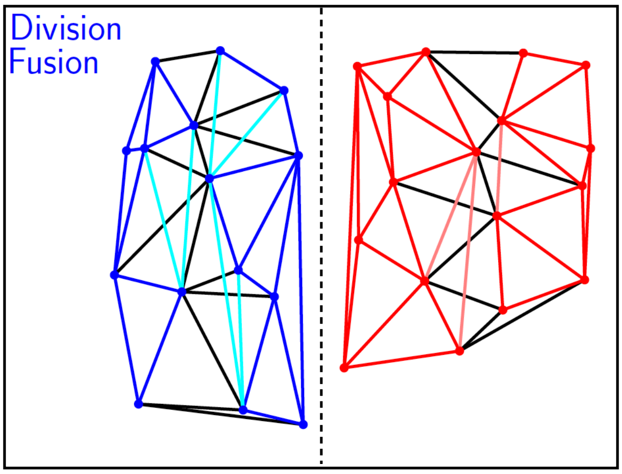
247



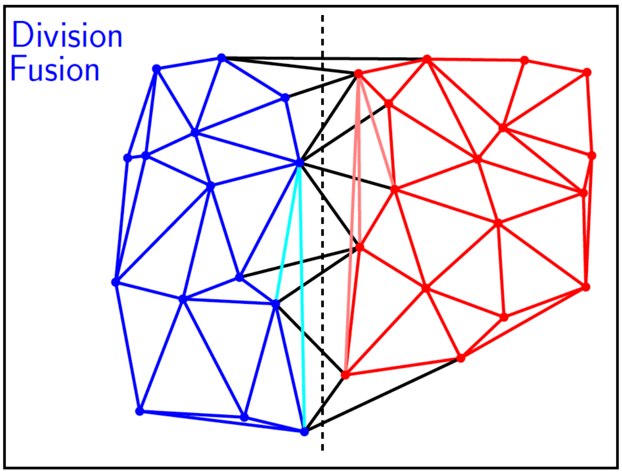
248



249



250



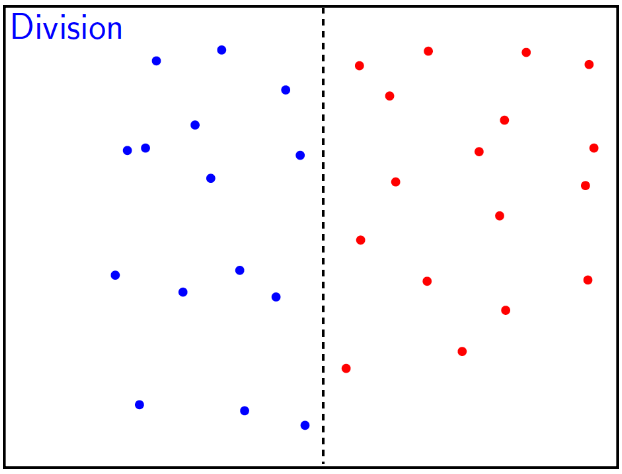
251

Delaunay Divide and Conquer Using a kd-tree

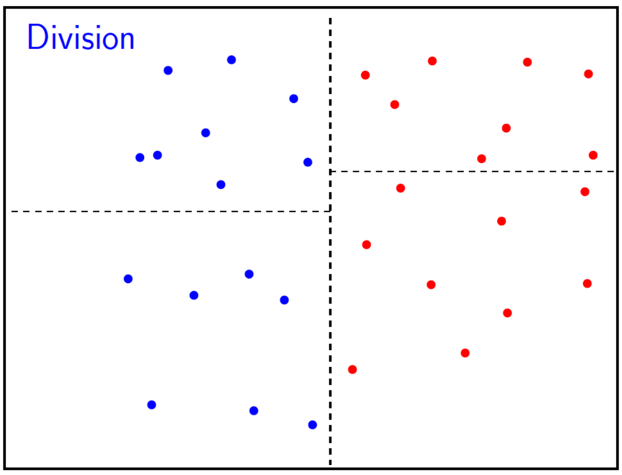
- Easily derecursifiable algorithm :
  - The construction of the connectivity is only performed at the recursive return (« Remontée récursive »)
  - The recursive split can be replaced by a prior sorting
- The split can be performed on x and y alternately using a kd-tree

252

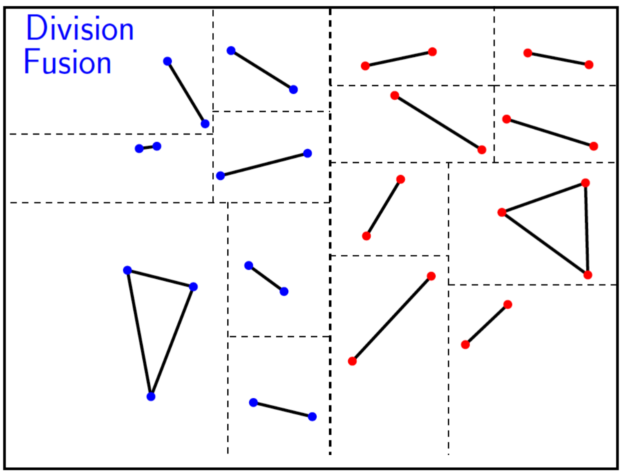
252



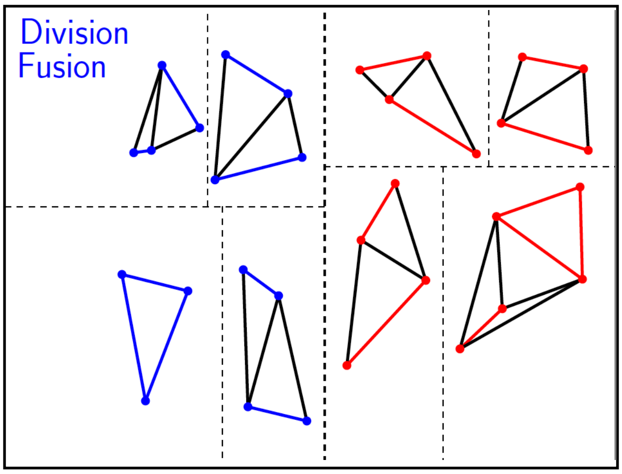
253



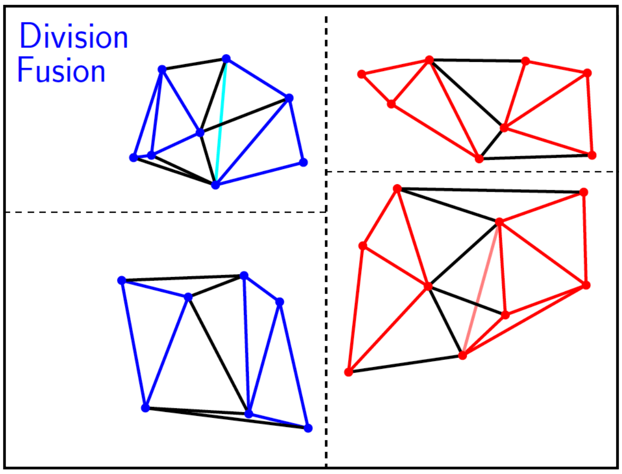
254



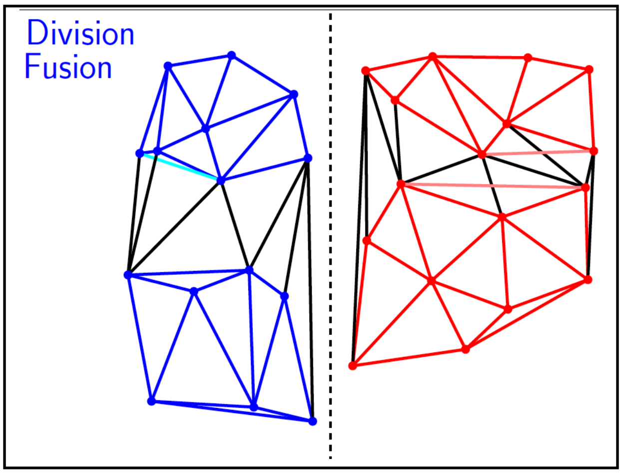
255



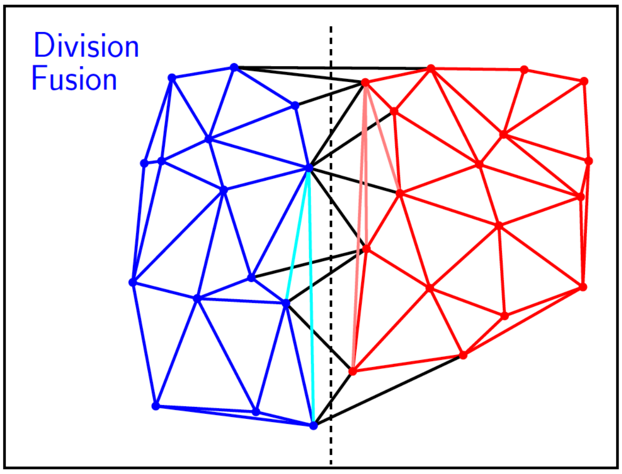
256



257



258



259