

# **Mesh simplification (part three): simplification algorithms**

**Leila De Floriani and Enrico Puppo  
Department of Computer Science  
University of Genova**

- Schemes for incremental simplification algorithms
- Examples of incremental simplification algorithms for scalar fields

# Paradigms for incremental simplification

- *Incremental refinement*: refinement of a coarse mesh through iterative application of *vertex insertion/ vertex split*
- *Incremental decimation (or coarsening)*: decimation of the mesh at full resolution through iterative application of *vertex removal/ edge collapse*
- All algorithms make use of heuristics techniques
- Choice where to apply next modification can be driven by several criteria, e.g.:
  - *approximation error* of the current approximation
  - shape of the simplexes
  - number of simplexes involved in a modification
- Approximation error is the most common criterion

# Incremental refinement algorithm

- **Input:**
  - mesh  $\square$  at the coarsest resolution
  - stopping criterion, which can be:
    - error threshold  $\epsilon$  on the output mesh, or
    - size  $s$  of the output mesh
- **Output:** mesh  $\square'$  such that
  - $|\square'| = s$  or
  - $E(\square', \square^*) \leq \epsilon$  and  $\square'$  of minimal sizewhere
  - $\square^*$  = mesh at full resolution (reference mesh)
  - $E(\square', \square^*)$  = error performed when approximating  $\square^*$  with  $\square'$

# Scheme of a refinement algorithm

- **Initialization**  $\square' = \square$
  - **Loop**
    - *select* a refinement modification;
    - *update*  $\square'$  through the refinement modification;
    - evaluate the *approximation error*  $E(\square', \square^*)$  associated with  $\square'$
- until**  $|\square'| = s$  or  $E(\square', \square^*) \leq \epsilon$

# Characteristics of different refinement algorithms

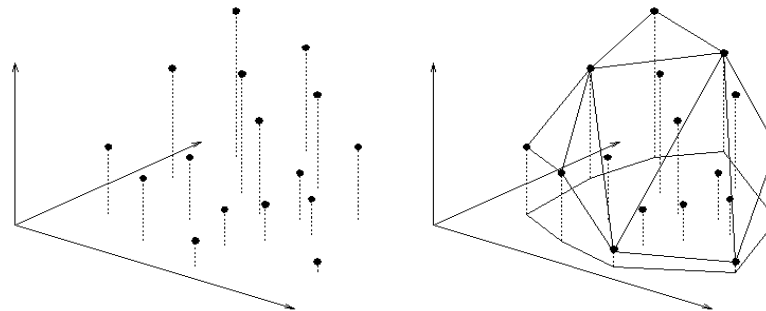
- *Type of refinement modification applied:*
  - vertex insertion (common for scalar fields)
  - vertex split
- *Heuristics used to select a refinement modification:*
  - usually, the one that best improves the approximation
  - random selection
- *Construction of the initial mesh (mesh  $\square$  at coarse resolution)*

## Construction of the initial mesh

- *For a scalar field:*
  - Detection of the boundaries of mesh  $\square^*$  at full resolution
  - Projection of such boundaries on the domain
  - Triangulation of the region surrounded by projected boundaries (e.g., Constrained Delaunay Triangulation)
  - Morphological features such as points (minima, maxima, saddle points), or line features (ridges and valleys) can be also included in the initial approximation of the scalar field (for 2D scalar fields)
- *For a free-form surface:*
  - Difficult in general
  - If the surface has genus zero, the convex hull of the vertices of mesh  $\square^*$  can be used

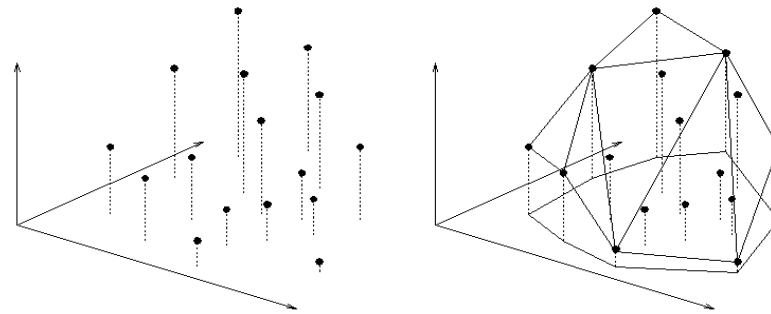
# Refinement in a scalar field by vertex insertion: notation

- $\square$  = mesh at the coarsest resolution covering the domain D of the scalar field (initial mesh)
- $\square^*$  = mesh at full resolution (reference mesh) covering the domain D of the scalar field ( $\square^*$  is the corresponding mesh in d dimensions)
- $V^*$  = vertices of mesh  $\square^*$
- $S^*$  = set of points in d-dimensional space corresponding to the points of  $V^*$  (a point  $p$  in  $S^*$  is defined by  $(q, f(q))$ , where  $q$  is a point in  $V^*$  and  $f(q)$  is the scalar field value at  $q$ )



# Example: refinement algorithm in a 2D scalar field through vertex insertion

- INPUT:
  - mesh  $\square$  at the coarsest resolution
  - sets  $V^*$  and  $S^*$
  - error threshold  $\epsilon$  on the output mesh
- OUTPUT:
  - mesh  $\square'$  covering the domain  $D$  of the scalar field such that  $E(\square', S^*) \leq \epsilon$  and  $\square'$  has minimal sizewhere  $E(\square', S^*)$  is the error associated to the 3D mesh  $\square'$  obtained by lifting  $\square'$  to the 3D space and computed with respect to the given data set of points  $S^*$



## Example: refinement algorithm in a 2D scalar field through vertex insertion (cont'd)

- *Initialization step:*
  - $\square' := \square$
  - $V' := V$
- *Initialization of error:*
  - for each vertex  $v$  in  $V^* - V$ :
    - let  $t$  be the triangle of  $\square'$  containing  $v$
    - compute the vertical distance  $d(v,t)$  from the point of  $S^*$  corresponding to  $v$  to the surface patch corresponding to  $t$
    - store  $v$ ,  $t$  and  $d(v,t)$  together in a data structure
  - $E(t) :=$  maximum of  $d(v,t)$  for all vertices in  $V^* - V$  falling inside  $t$  gives
  - $E(\square', S^*) := \max E(t)$  over the triangles in  $\square'$

## Example: refinement algorithm in a 2D scalar field through vertex insertion (cont'd)

- *Main Loop:*
  - select the point  $v$  of  $V^*-V'$  with the maximum error;
  - insert  $v$  in  $\mathcal{T}'$  by performing a vertex insertion: the set of triangles  $\mathcal{T}_1$  to be replaced in  $\mathcal{T}'$  with the new set of triangles  $\mathcal{T}_2$  are defined by the Delaunay criterion;
  - $V' := V' \cup \{v\}$
  - compute the error associated with the triangles in  $\mathcal{T}_2$
  - update the error associated with the points of  $V^*-V'$  which fall inside the triangles of  $\mathcal{T}_2$
  - update  $E(\mathcal{T}', S^*)$  as the maximum of the errors on the triangles in the new  $\mathcal{T}'$  (we do not need to re-compute the error for the triangles in  $\mathcal{T}' - \mathcal{T}_1$ )
- The *main loop* is repeated while  $E(S^*, \mathcal{T}') > \epsilon$

## **Example: refinement algorithm in a 2D scalar field through vertex insertion (cont'd)**

- *Implementation:*
  - An auxiliary data structure is necessary to maintain relations between each vertex in  $V^*-V'$  and the triangle of  $\square'$  containing it
  - Usually, a priority queue of the vertices to be inserted is maintained sorted by decreasing error values
  - A topological data structure for storing the mesh is necessary to support updates
- *Time complexity:*
  - To be evaluated on a specific implementation
  - It depends on the computation of the update and on the data structures used

# Incremental coarsening algorithm

- **Input:**
  - mesh  $\mathcal{T}^*$  at full resolution
  - stopping criterion, which can be:
    - error threshold  $\epsilon$  on the output mesh, or
    - size  $s$  of the output mesh
- **Output:** mesh  $\mathcal{T}'$  such that
  - $|\mathcal{T}'| = s$  or
  - $E(\mathcal{T}', \mathcal{T}^*) \leq \epsilon$  and  $\mathcal{T}'$  has minimal size

# Scheme of a coarsening algorithm

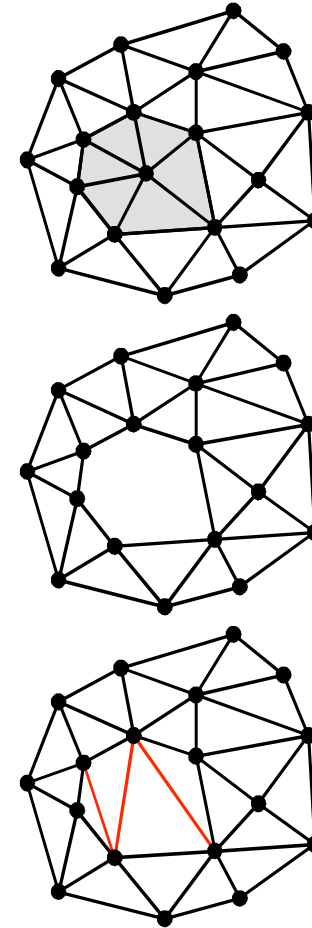
- **Initialization:**  $\Omega' = \Omega$ ;
  - **Loop**
    - *select* the vertex to be removed or the edge to be collapsed in  $\Omega'$ ;
    - *update*  $\Omega'$  though vertex removal or edge collapse;
    - evaluate the *approximation error*  $E(\Omega', \Omega^*)$  associated with  $\Omega'$
- until**  $|\Omega'| = s$  or  $\Omega'$  cannot be coarsened without violating the error condition  $E(\Omega', \Omega^*) \leq \epsilon$

# Characteristics of different coarsening algorithms

- *Type of coarsening modifications:*
  - Vertex removal
  - Edge collapse
  
- *Heuristics used to select a modification:*
  - usually, the one which causes the least increase in the approximation error
  - random selection

# Example: coarsening in a 2D scalar field by vertex removal

- INPUT:
  - mesh  $\mathcal{T}^*$  at full resolution covering the domain  $D$
  - error threshold  $\epsilon$  on the output mesh
- OUTPUT:
  - mesh  $\mathcal{T}'$  such that  $E(\mathcal{T}', S^*) \leq \epsilon$  and  $\mathcal{T}'$  has a minimal size



## Example: coarsening in a 2D scalar field by vertex removal

- *Initialization step:*
  - $\square' := \square^*$
  - $V' := V^*$
  - compute the error associated with the points in  $V'$ : for each vertex  $v$  in  $V'$ ,
    - simulate the removal of  $v$  from  $\square'$
    - compute  $E(v)$  by considering distance from the surface patch corresponding to the triangle containing  $v$  in the mesh resulting from the removal of  $v$
  - select the point  $v$  in  $V'$  with the minimum error  $E(v)$
  - $\text{current\_error} := E(v)$

## Example: coarsening in a 2D scalar field by vertex removal

- *Main Loop:*
  - while current-error  $\leq \epsilon$  do
    - delete  $v$  from  $\mathcal{T}'$  by performing a vertex removal:
      - the set of triangles  $\mathcal{T}'_2$  of  $\mathcal{T}'$  incident in  $v$  are replaced with a new set of triangles  $\mathcal{T}'_1$  defined by the Delaunay criterion
    - $V' := V' - \{v\}$
    - update the error associated with the points of  $V'$  which belong to the boundary of  $\mathcal{T}'_2$
    - select the point  $v$  of  $V'$  with the minimum error  $E(v)$
    - current\_error :=  $\max(\text{current\_error}, E(v))$

# Example: coarsening in a 2D scalar field by vertex removal

- Usually, a *priority queue* of the vertices to be removed is maintained sorted by increasing error values
- *Time complexity:*
  - To be evaluated on a specific implementation
  - It depends on the computation of the update and on the data structures used