EUROGRAPHICS 2016

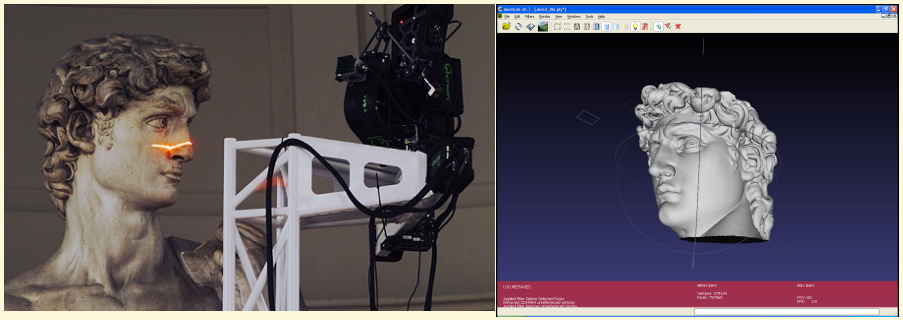
THE 37th ANNUAL CONFERENCE OF THE
EUROPEAN ASSOCIATION FOR COMPUTER GRAPHICS

Structure aware mesh decimation

David Salinas, Florent Lagarge, Pierre Alliez (Inria)

Size of acquired data

The size of acquired data has grown exponentially over the last decades



Resolution increase

Size of object increase

Currently, recent automatic acquisition workflows (MVS, lidar) are able to capture dense meshes with billion of vertices

Around 6 millions vertices for 1 km² of a city with reasonable accuracy
With same density, around 10^15 points to represent the whole land world area

Dire need for **simplification algorithms** to use this data
(online visualization, numeric simulation, ...).

A (very) brief history of geometrical simplification

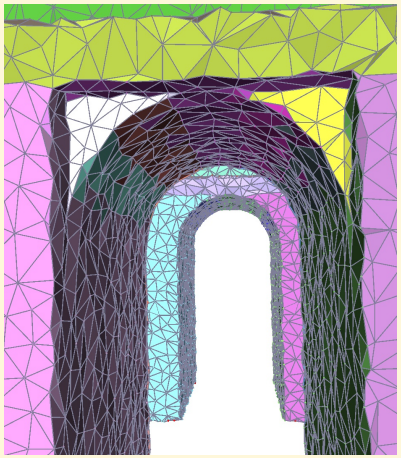
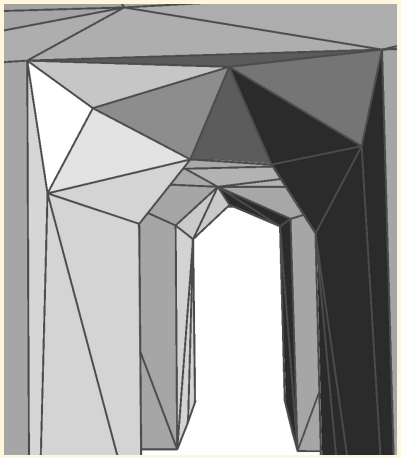
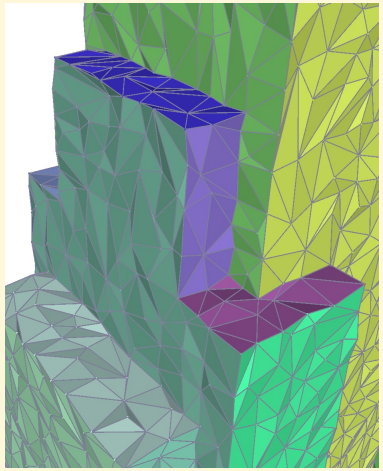
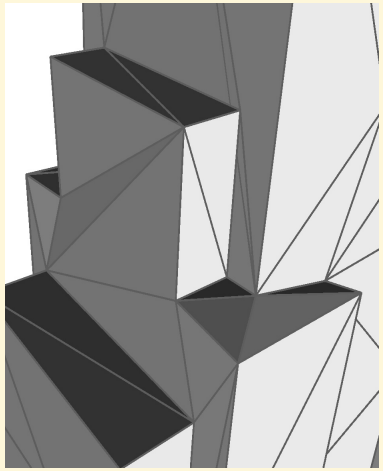
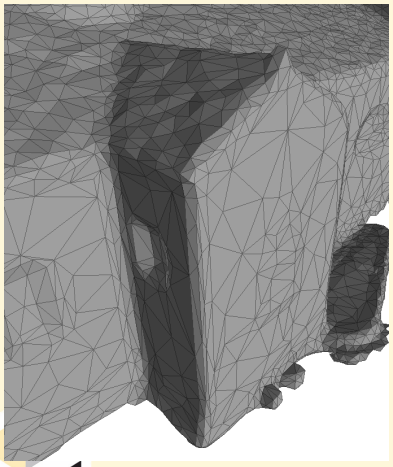
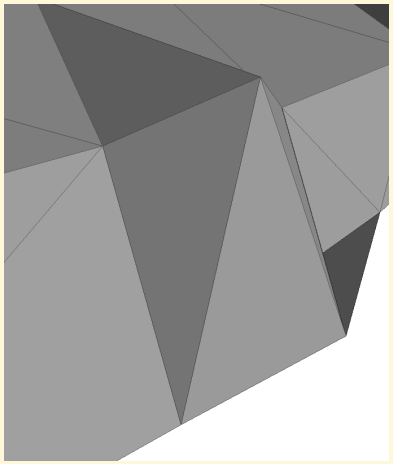
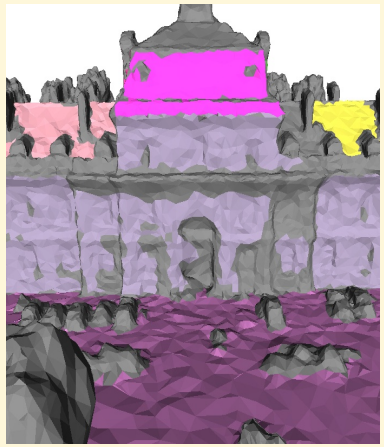
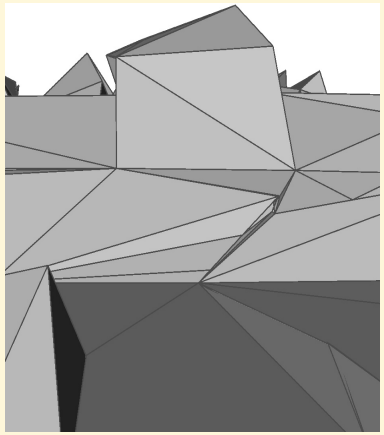
Geometrical simplification can be seen as lossy compression

Goal : preserve geometry of simplified shape while reducing its complexity

- Quadric error metric [Garland Heckbert 98]
- Memoryless simplification [Lindstrom Turk 99]
- Variational approach VSA [Cohen-Steiner Alliez Desbrun 04]

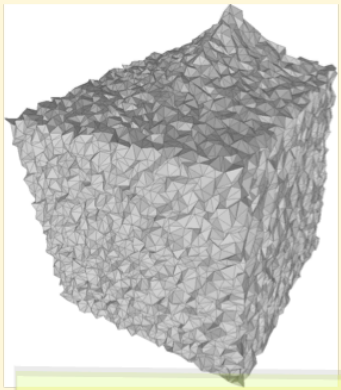
Common problem : structure alteration at **coarse** complexities

Structure alteration

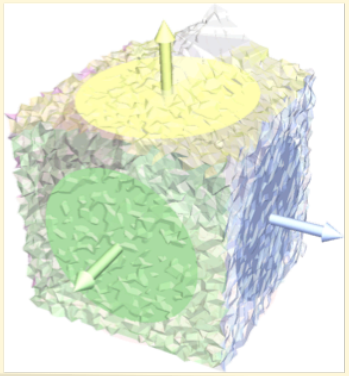









Our method – overview

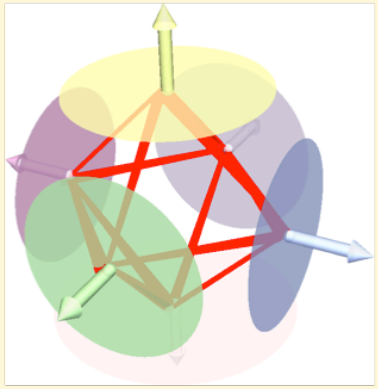
Our method



Initial mesh
6,500 vertices

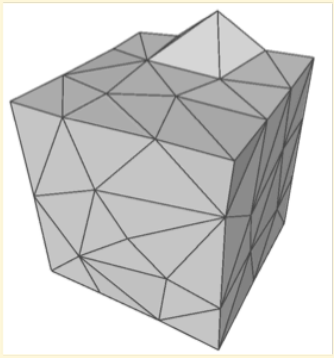


Detection of proxies

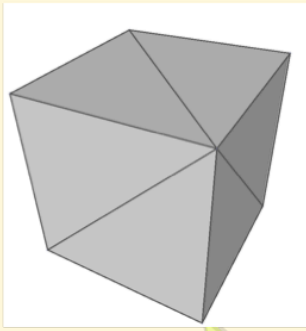


Graph of proxies

65 vertices



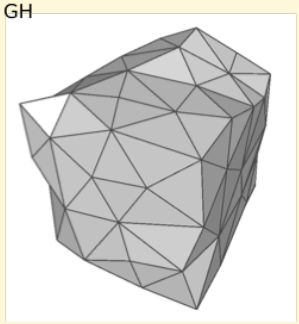
8 vertices
(end of decimation)



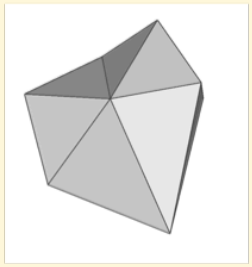
Decimation

Structure unaware

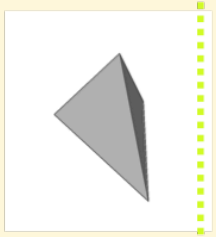
GH



65 vertices

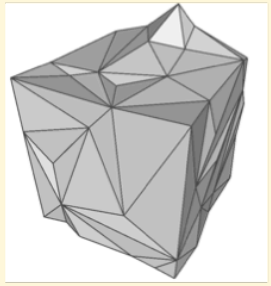


8 vertices

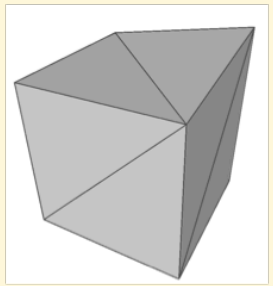


end

LT



65 vertices

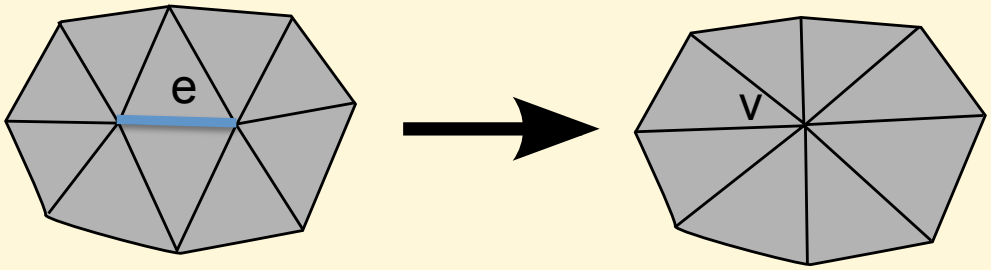


8 vertices

end

General decimation algorithm

Edge collapse: merge two vertices of an edge *e* into a vertex *v*



Decimation algorithm:

- Edges <- All edges of the mesh to be simplified
- While Edges not empty
  - Extract e in Edges where *cost(e)* is minimum
  - If valid(e) is true then
    - Collapse(e) (place the new point where the cost is minimized)
    - Update cost of edges in the neighborhood of e

Parameterized by a **cost function *cost(e)*** and a **validity function *valid(e)***

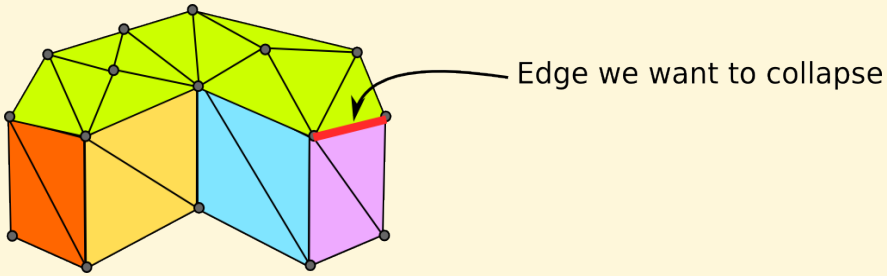
Decimation algorithm – Cost function

Cost function :
Given an edge *e* and a new position *v* specify a (geometric) cost for this alteration.

Placement function :
Given an edge *e* specify where the new vertex must be placed after the edge collapse.
Usually placed at the point minimizing the cost.

Our cost function :
Takes into account squared distances to :

- planes of neighbors triangles
- orthogonal planes of neighbors border triangles
- planes of neighbors proxies
- orthogonal planes of neighbors border proxies triangles



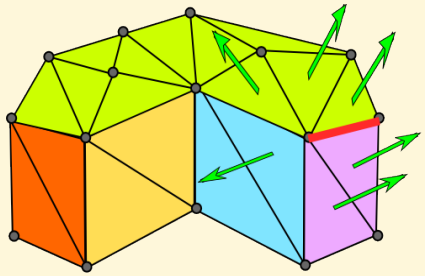
Decimation algorithm – Cost function

Placement function :
Given an edge *e* specify where the new vertex must be placed after the edge collapse.
Usually placed at the point minimizing the cost.

Cost function :
Given an edge *e* and a new position *v* specify a (geometric) cost for this alteration.

Our cost function :
Takes into account squared distances to :

- planes of neighbors triangles
- orthogonal planes of neighbors border triangles
- planes of neighbors proxies
- orthogonal planes of neighbors border proxies triangles



orthogonal planes
of border triangles

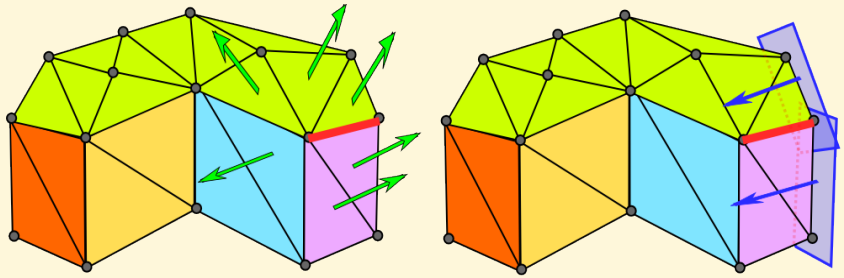
Decimation algorithm – Cost function

Placement function :
Given an edge *e* specify where the new vertex must be placed after the edge collapse.
Usually placed at the point minimizing the cost.

Cost function :
Given an edge *e* and a new position *v* specify a (geometric) cost for this alteration.

Our cost function :
Takes into account squared distances to :

- planes of neighbors triangles
- orthogonal planes of neighbors border triangles
- planes of neighbors proxies
- orthogonal planes of neighbors border proxies triangles



orthogonal planes of border triangles

orthogonal planes of border triangles

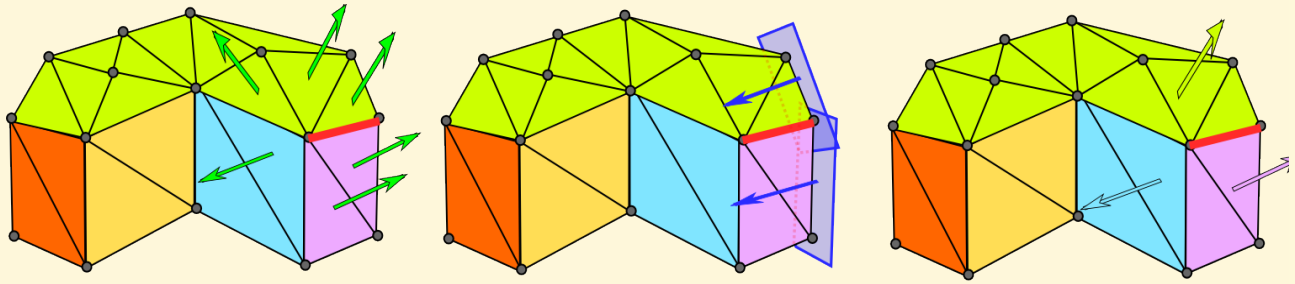
Decimation algorithm

Placement function :
Given an edge *e* specify where the new vertex must be placed after the edge collapse.
Usually placed at the point minimizing the cost.

Cost function :
Given an edge *e* and a new position *v* specify a (geometric) cost for this alteration.

Our cost function :
Takes into account squared distances to :

- planes of neighbors triangles
- orthogonal planes of neighbors border triangles
- planes of neighbors proxies
- orthogonal planes of neighbors border proxies triangles



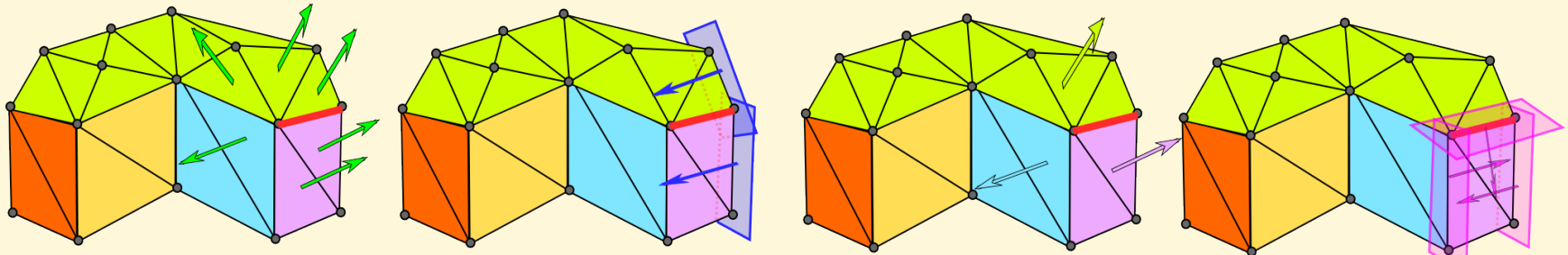
Decimation algorithm – Cost function

Placement function :
Given an edge *e* specify where the new vertex must be placed after the edge collapse.
Usually placed at the point minimizing the cost.

Cost function :
Given an edge *e* and a new position *v* specify a (geometric) cost for this alteration.

Our cost function :
Takes into account squared distances to :

- planes of neighbors triangles
- orthogonal planes of neighbors border triangles
- planes of neighbors proxies
- orthogonal planes of neighbors border proxies triangles



Decimation algorithm – Cost function

Placement function :
Given an edge *e* specify where the new vertex must be placed after the edge collapse.
Usually placed at the point minimizing the cost.

Cost function :
Given an edge *e* and a new position *v* specify a (geometric) cost for this alteration.

Our cost function :
Takes into account squared distances to :

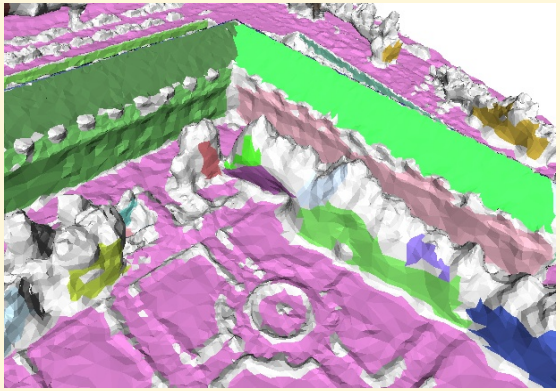
- planes of neighbors triangles
- orthogonal planes of neighbors triangles (if edge on the border)
- planes of neighbors proxies
- orthogonal planes of neighbors proxies (if edge on the border of a proxy)

Two parameters in (0,1) :

- ***Border*** parameter : trade border preservation for inner surface preservation
- ***Abstraction*** parameter : trade planar proxy proximity to initial surface proximity

Decimation algorithm – abstraction parameter

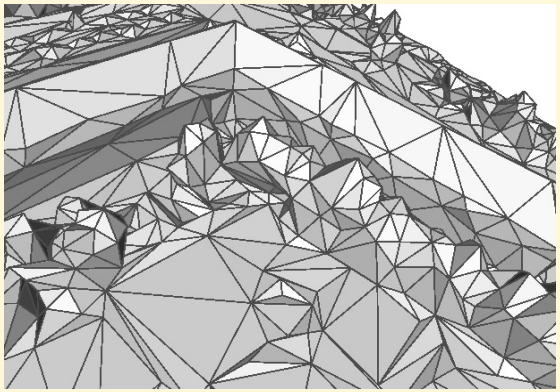
Abstraction parameter : trade planar proxy proximity to initial surface proximity



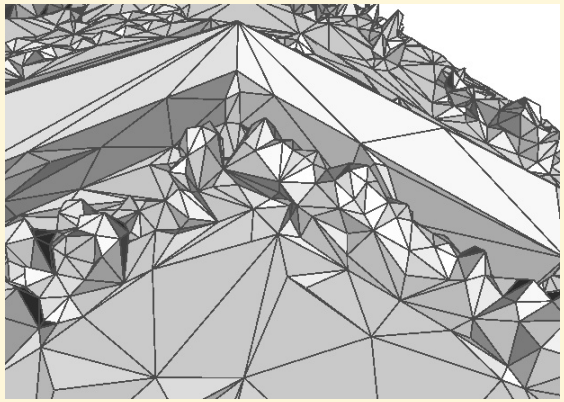
Initial mesh with detected planar proxies

Simplification when changing abstraction parameter λ

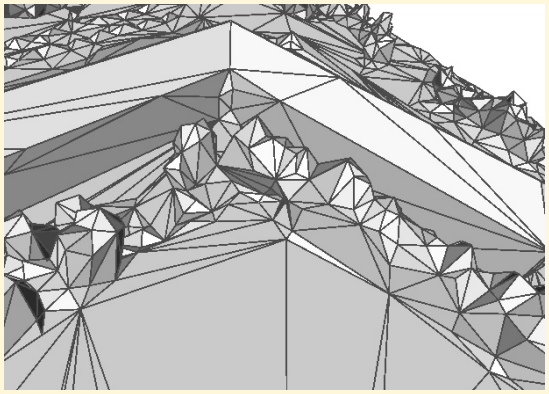
λ=0



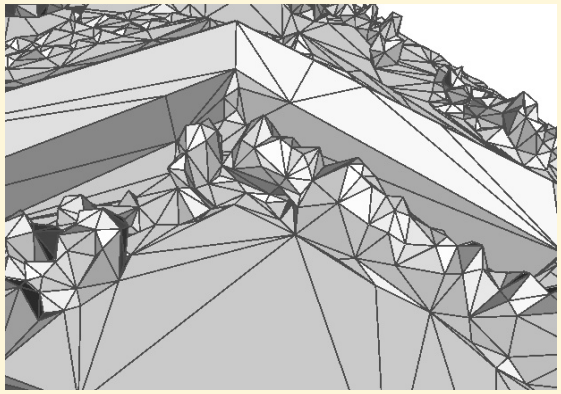
λ=0.5



λ=0.8



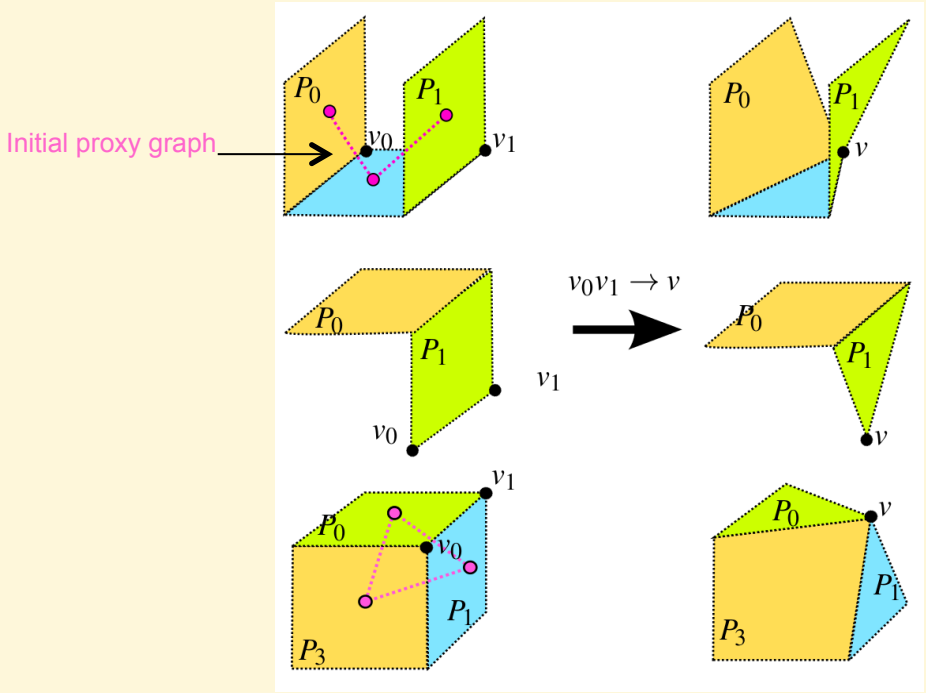
λ=0.99



Decimation algorithm – validity function

Validity criterion :

Reject an edge collapse if it violates one the three structure preserving rules :



Forbidden proxy join:

Two proxies can't be join if they are not linked in the initial proxy graph.

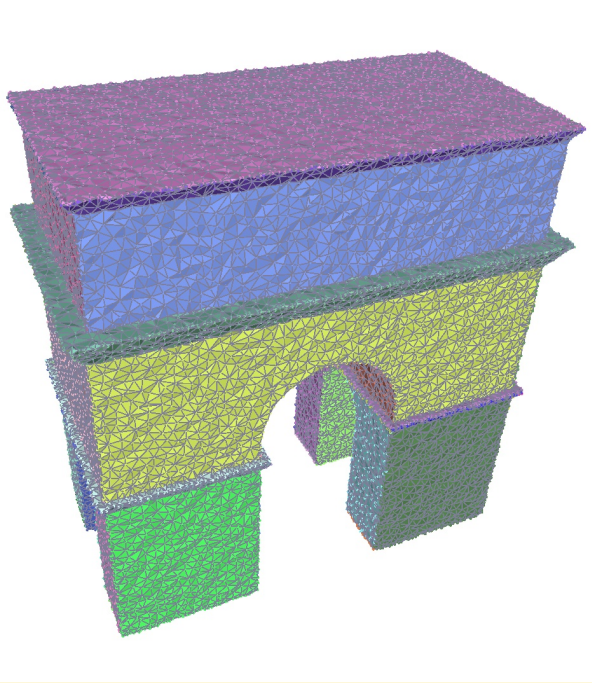
Proxy degeneration :

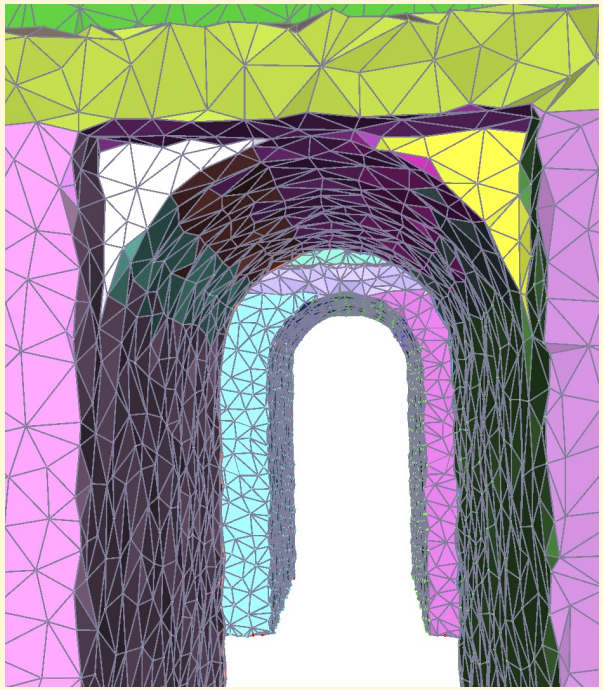
A proxy had less than 4 vertices after an edge collapse.

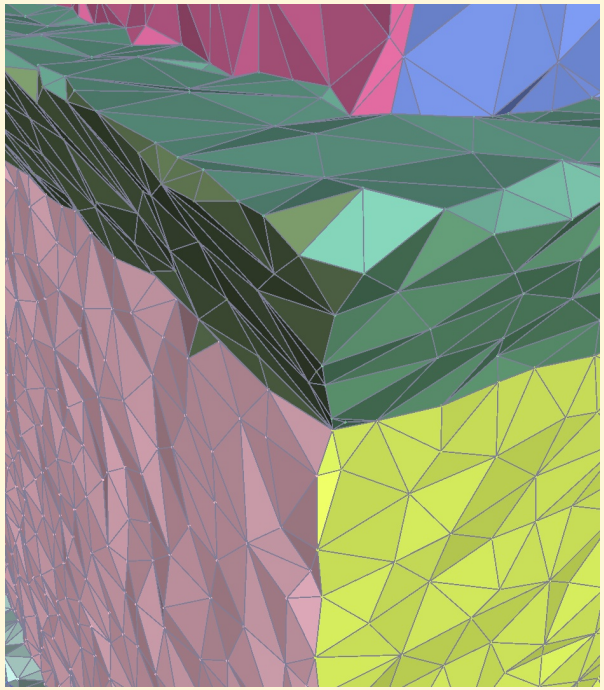
Corner alteration :

A corner is destroyed after an edge collapse.

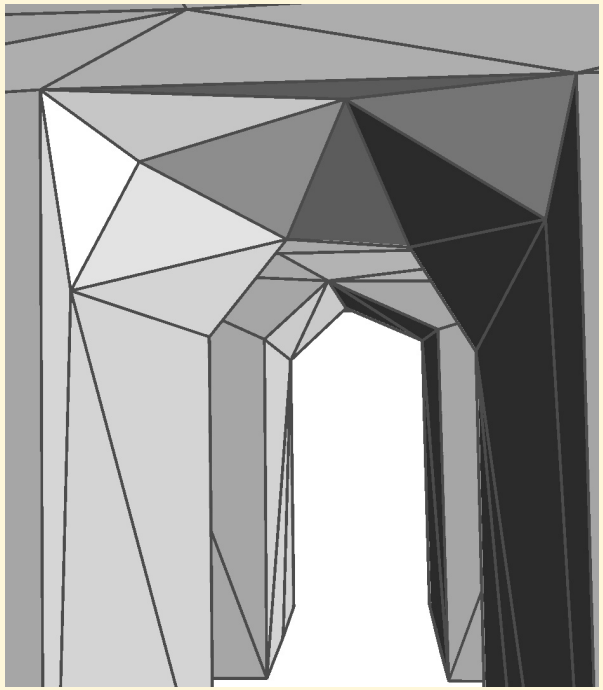
Some results

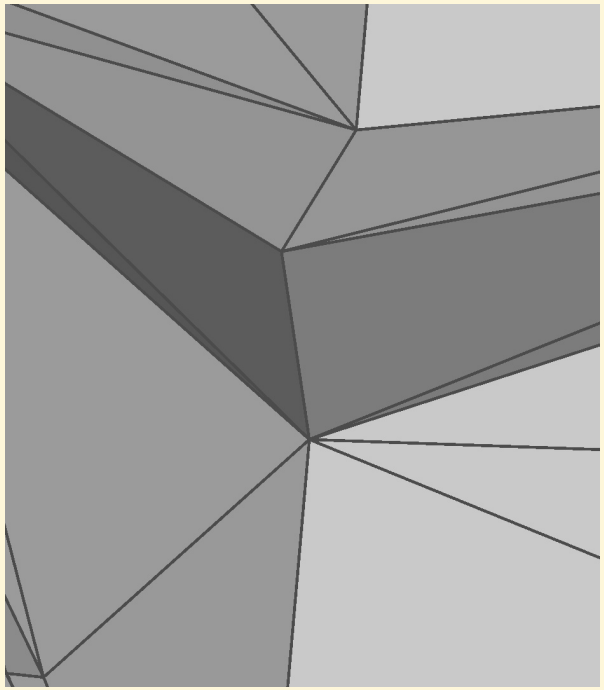




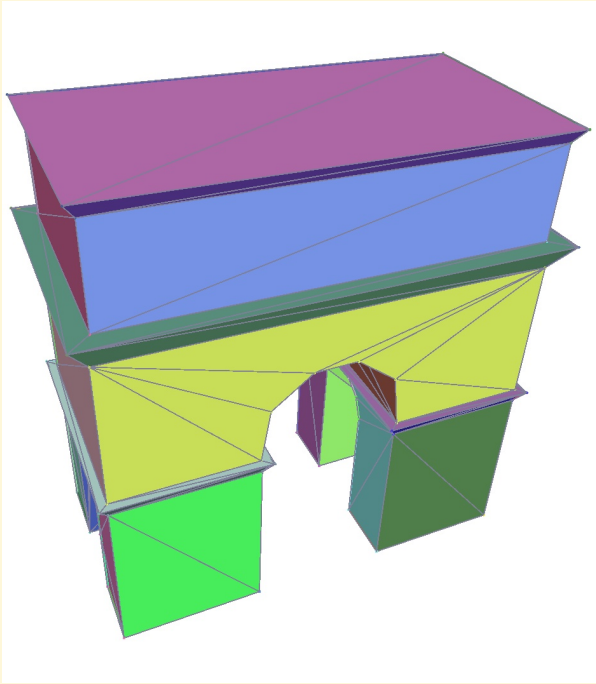


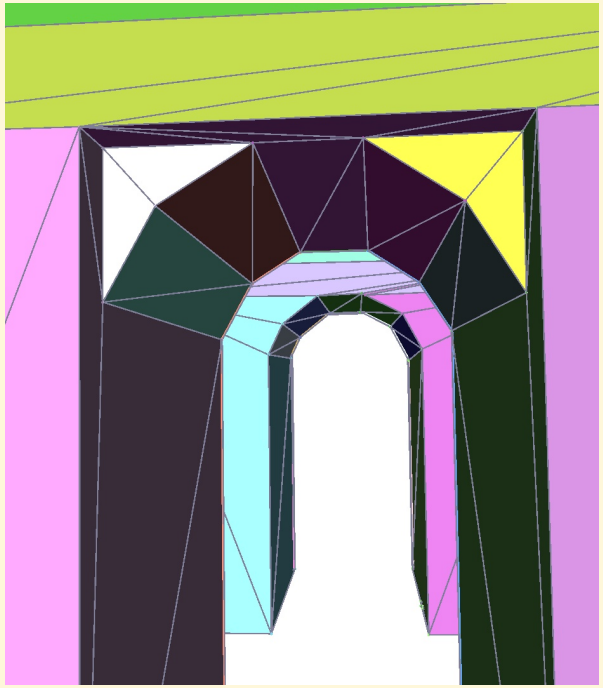
Some results

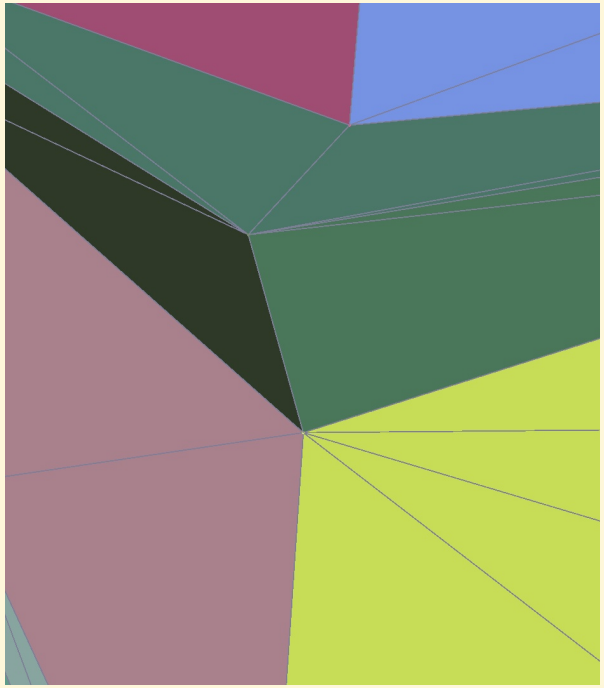




Some results

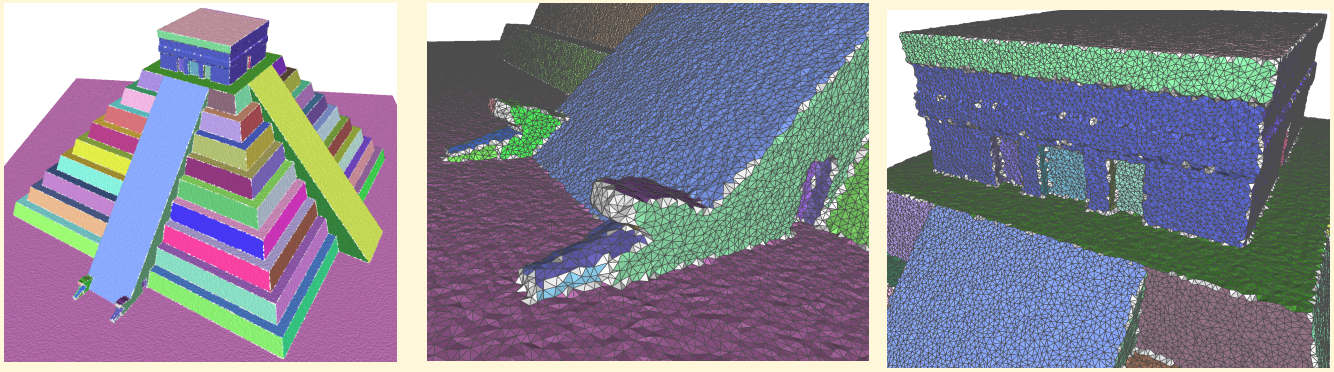




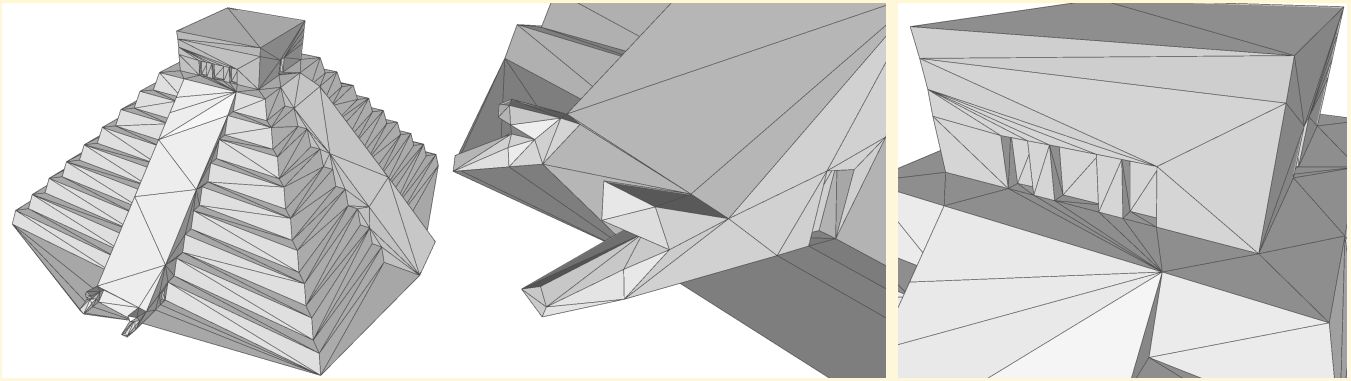


Some results

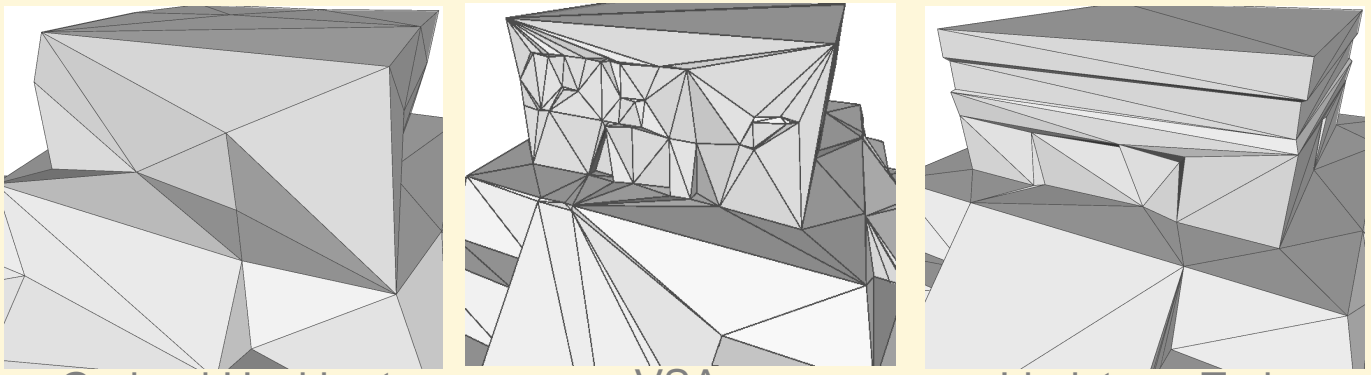
Input : 400,000 vertices



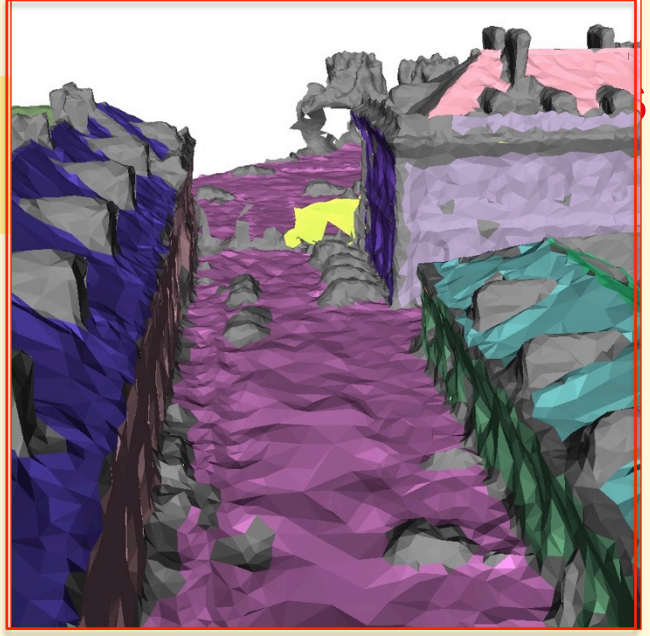
Our output when simplification **stops** (350 vertices)

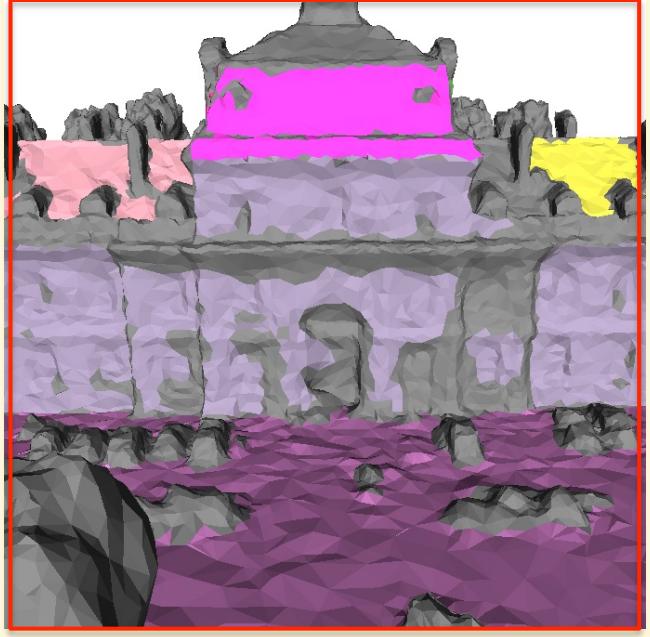


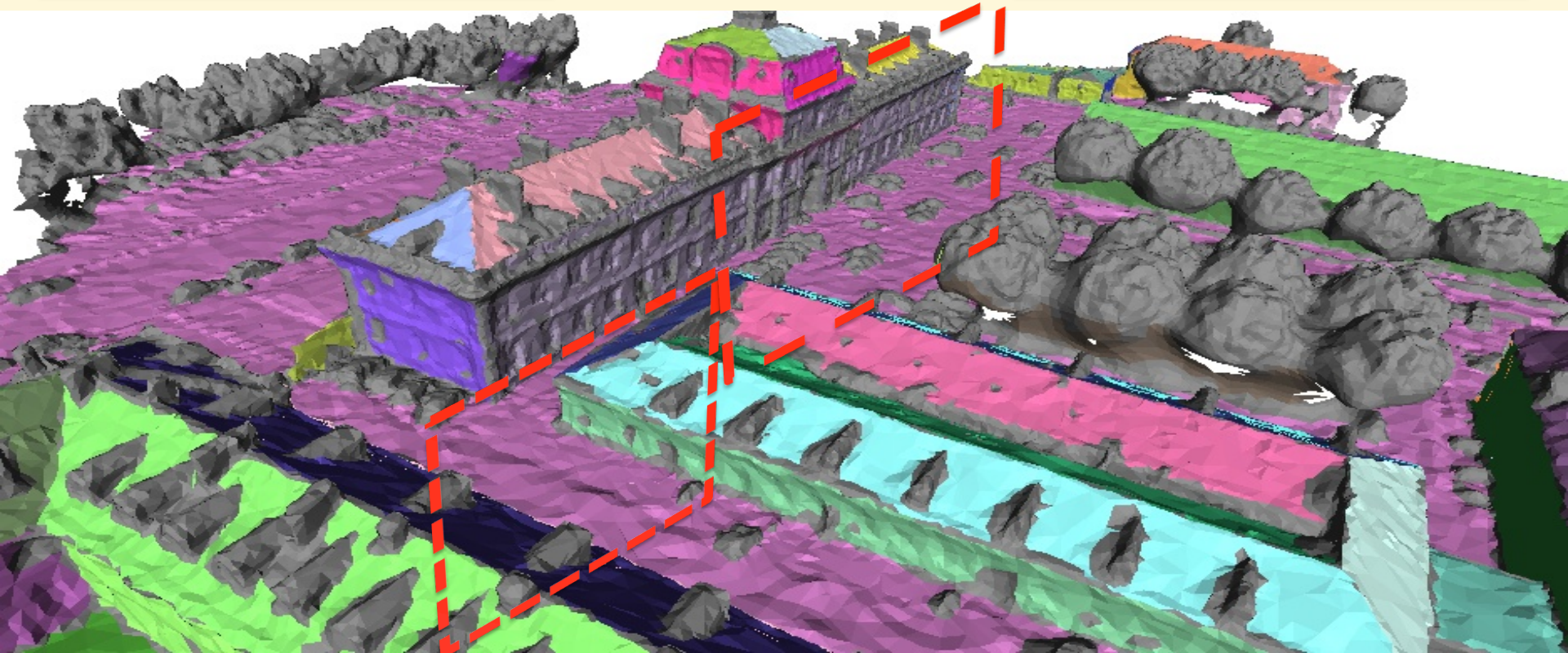
Structure unaware methods (350 or more vertices)

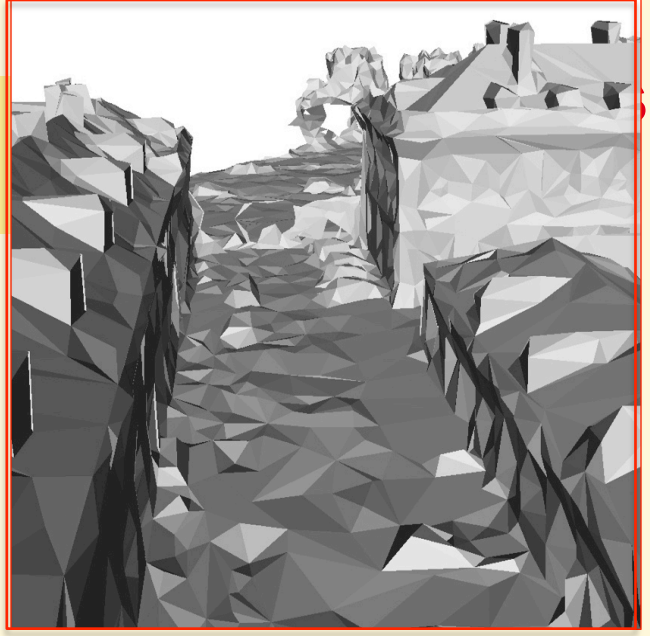


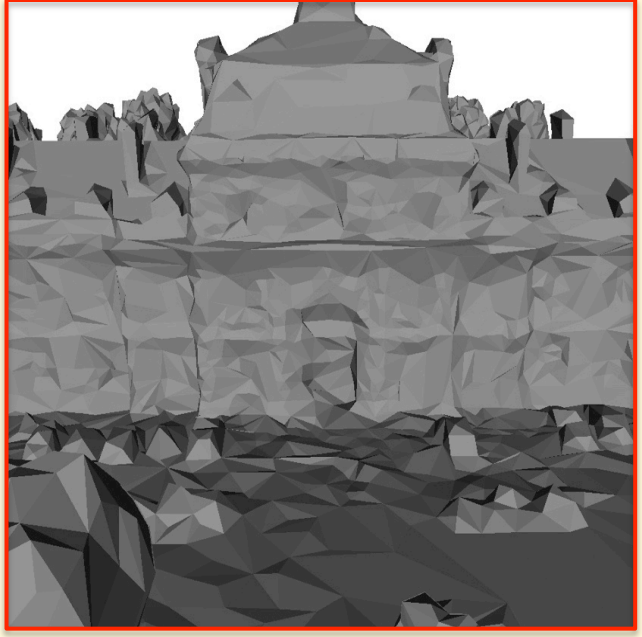
Garland Heckbert | VSA | Lindstrom-Turk

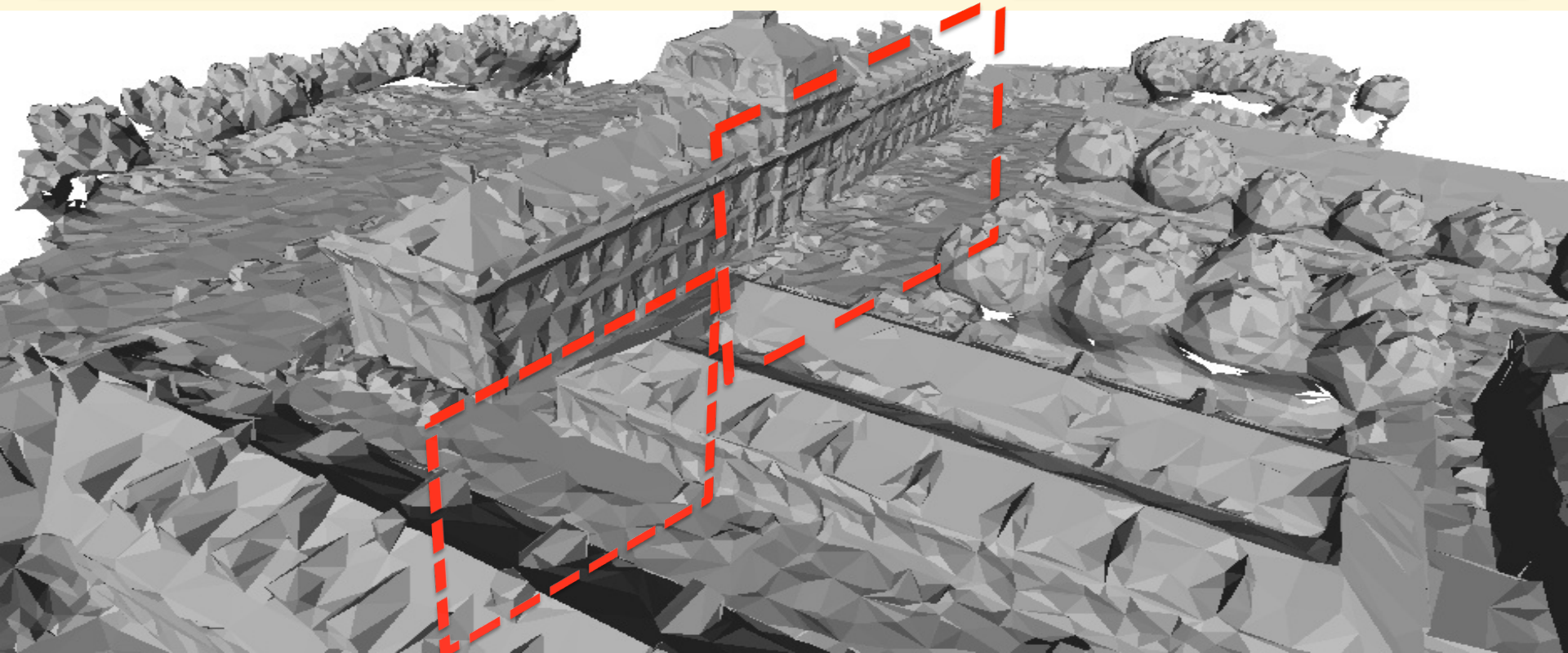


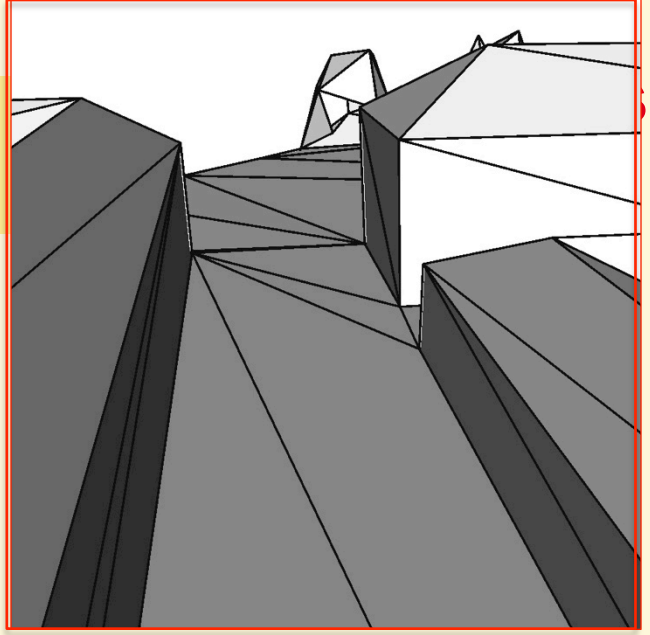


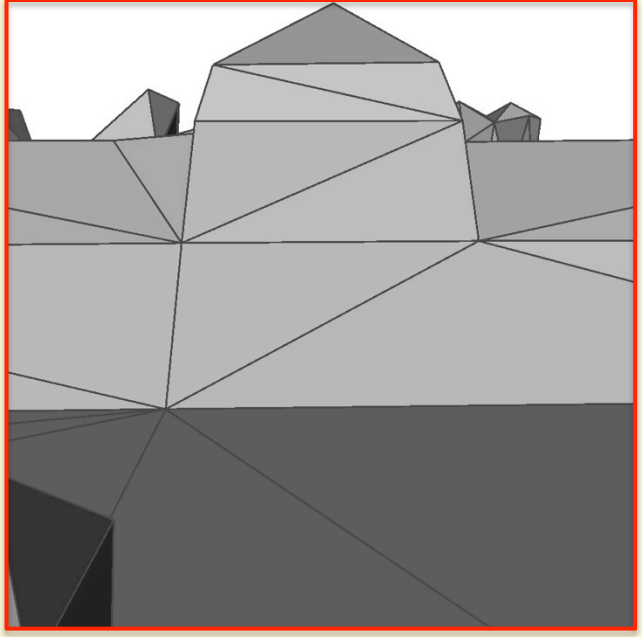


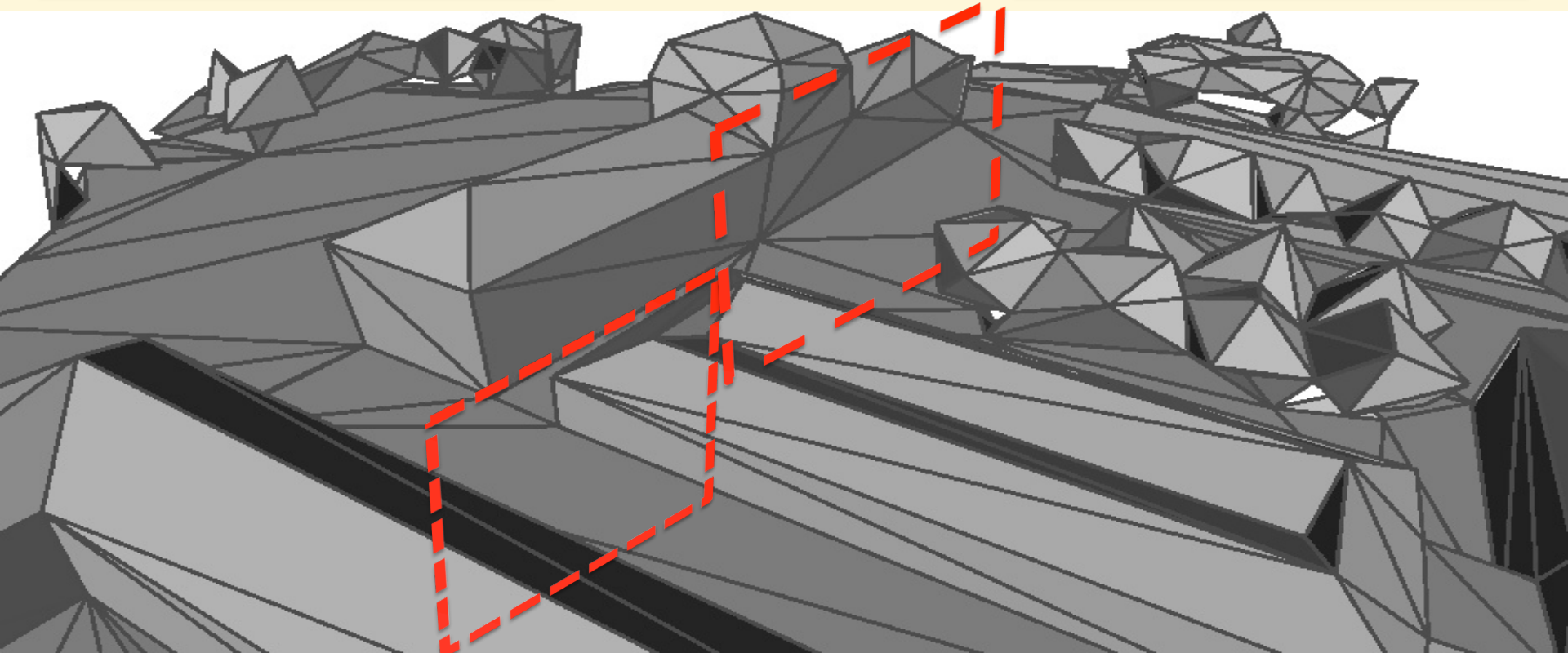


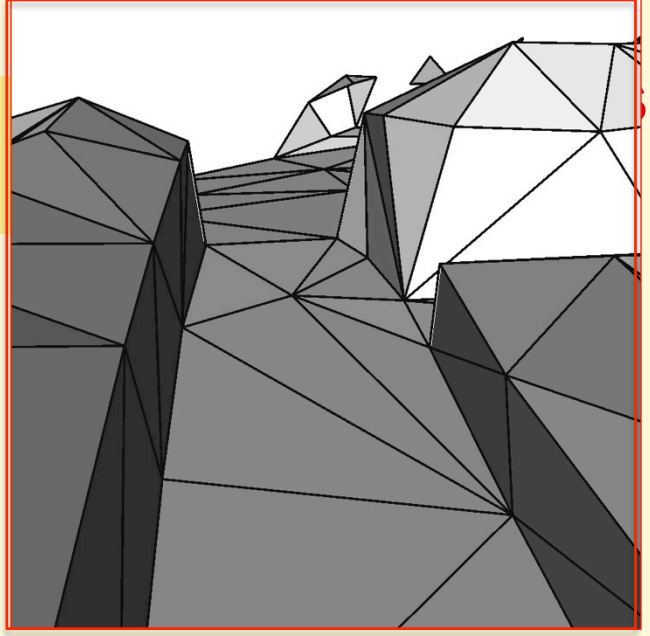


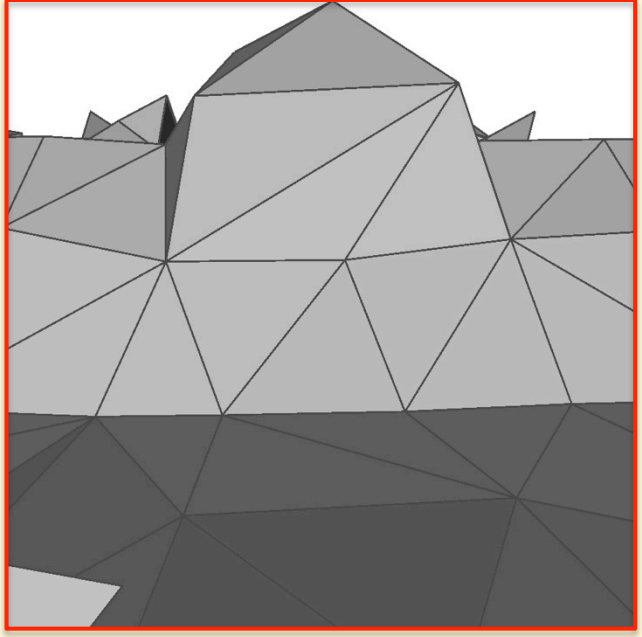


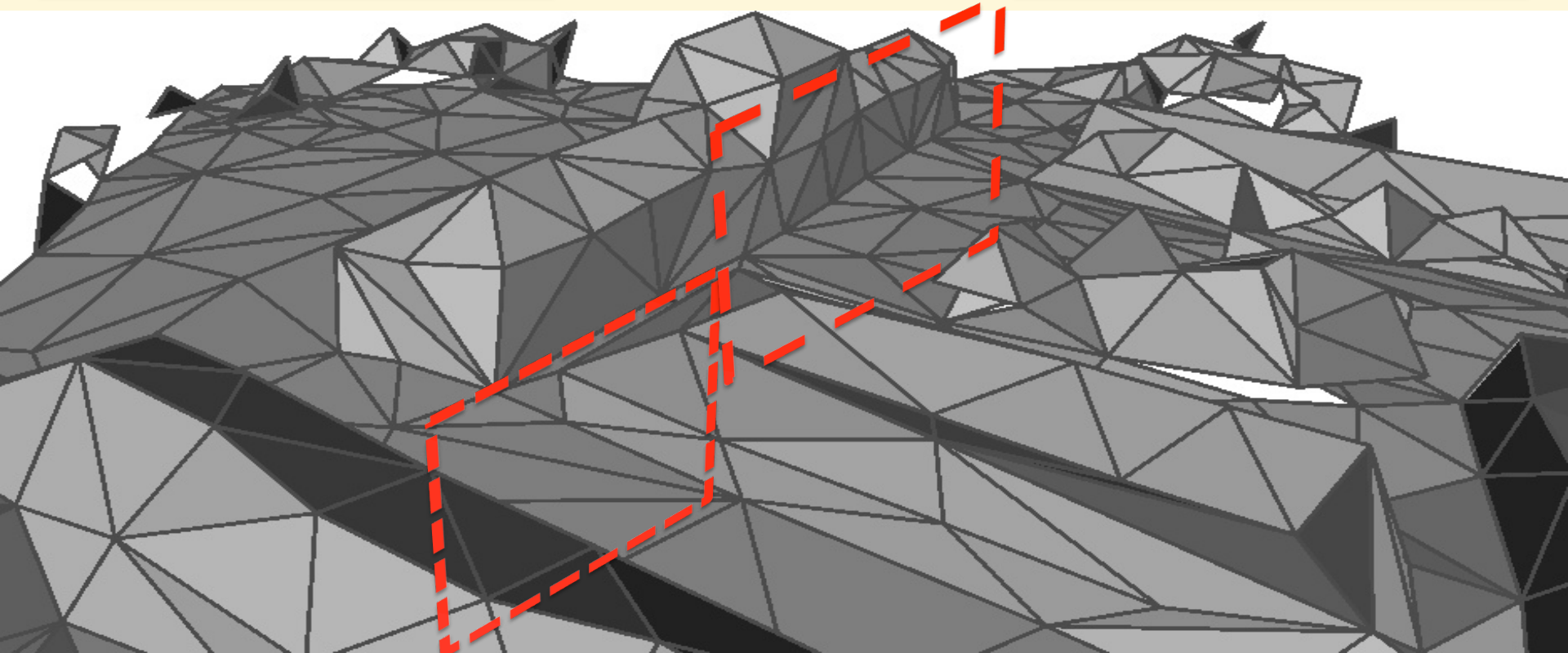


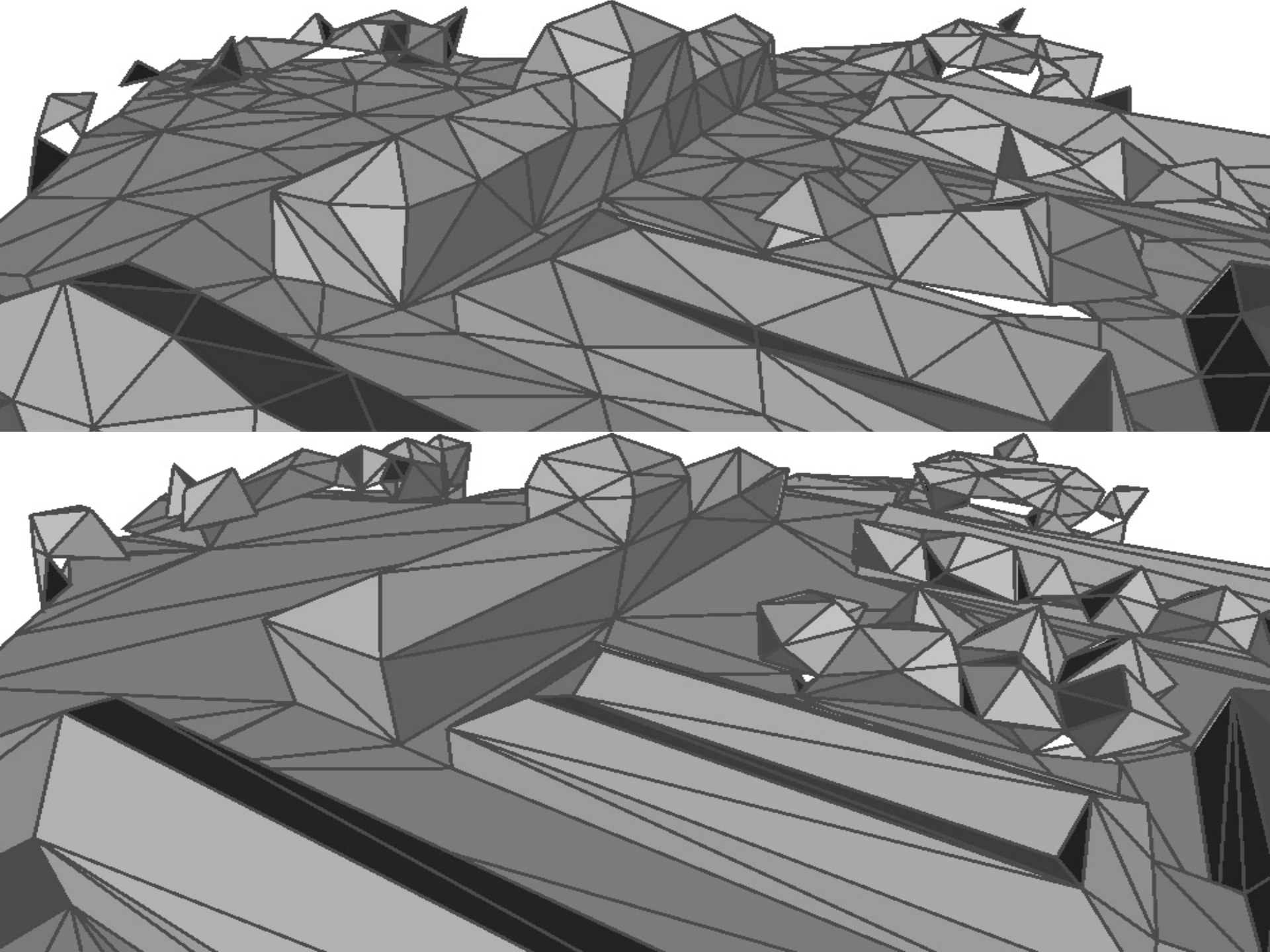












Generic data-structure

Skeleton-blocker data-structure

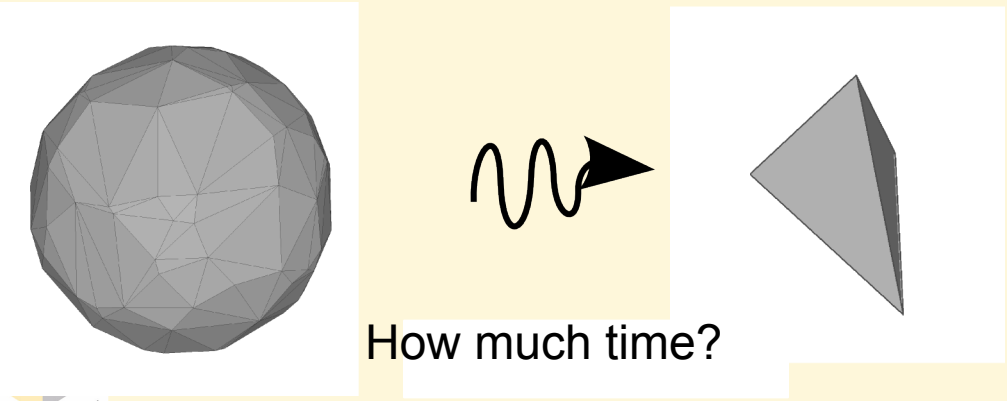
Based on 'Efficient data-structure for representing and simplifying simplicial complexes in high dimensions' [Attali Lieutier Salinas SoCG11, IJCGA12]

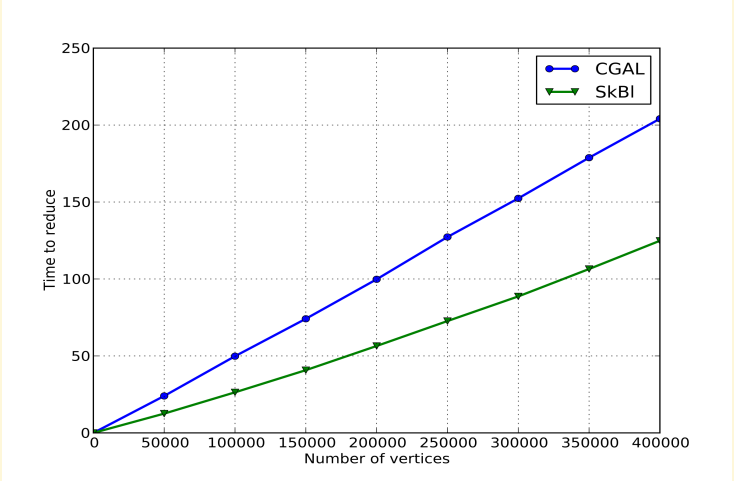
Able to represent simplicial complexes (meshes) of ***any dimension***, not necessarily ***manifold*** (contrarilty to CGAL Polyhedron_3)

Around ***65 % times faster*** than CGAL to contract a random sphere in 3D with n points to a tetrahedron

Public released by the end of 2014 in Gudhi's open source library

https://project.inria.fr/gudhi/software/





Conclusion and future works

Structure-aware mesh decimation :

- Decimation : automated, efficient, robust (geometrical and topological noise)
- Structure : robustness to imperfect structure detection, hybrid

Possible extensions :

- Non planar primitives
- Simplification of primitives set and graph in tandem
- Semantic rules


Thank you!