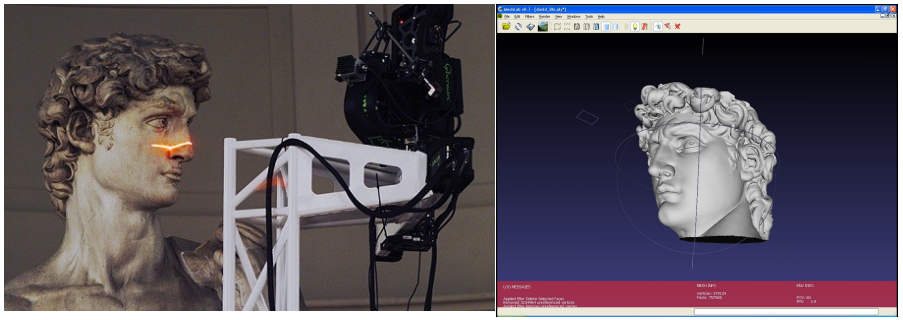
Structure aware mesh decimation

David Salinas, Florent Lafarge and Pierre Alliez

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 Paris
With same density, around 10^15 points to represent the whole land world area

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

A (very) brief history of geometrical simplification

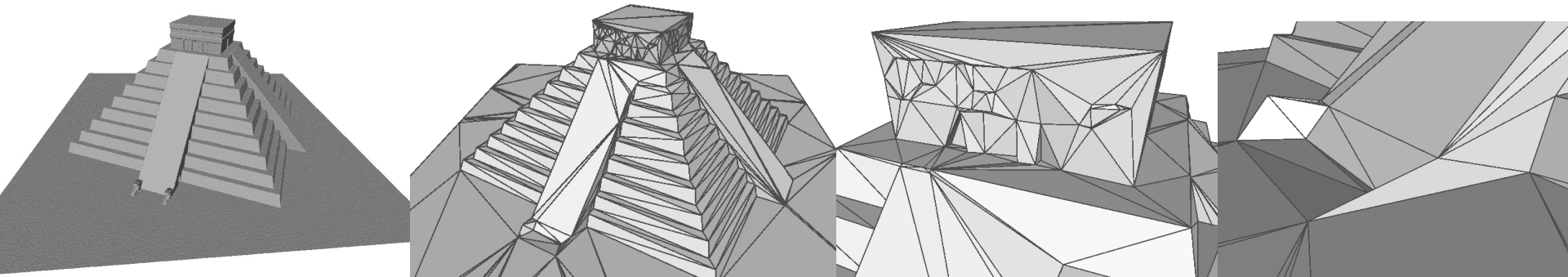
Geometrical simplification can be seen as lossy compression

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

Simplification with VSA from 400,000 to 400 vertices



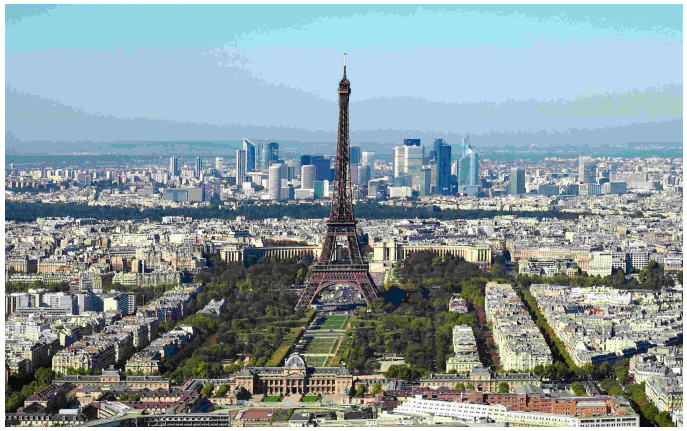
Structural disgression

But ***what*** is structure ?

«Shape structure is about the arrangement and relations between shape parts. Its goes beyond local geometry refinement and detail-preserving shape deformation.»
Survey [Structure-aware shape processing 13 Mitra et al.]

Our work focuses on ***man-made*** *shapes :*

Non man-made shape
Predominence of so-called smooth or free-form shapes



Man-made shape
Predominence of so-called geometric proxies
(planar, spheric, cylindric and so on)

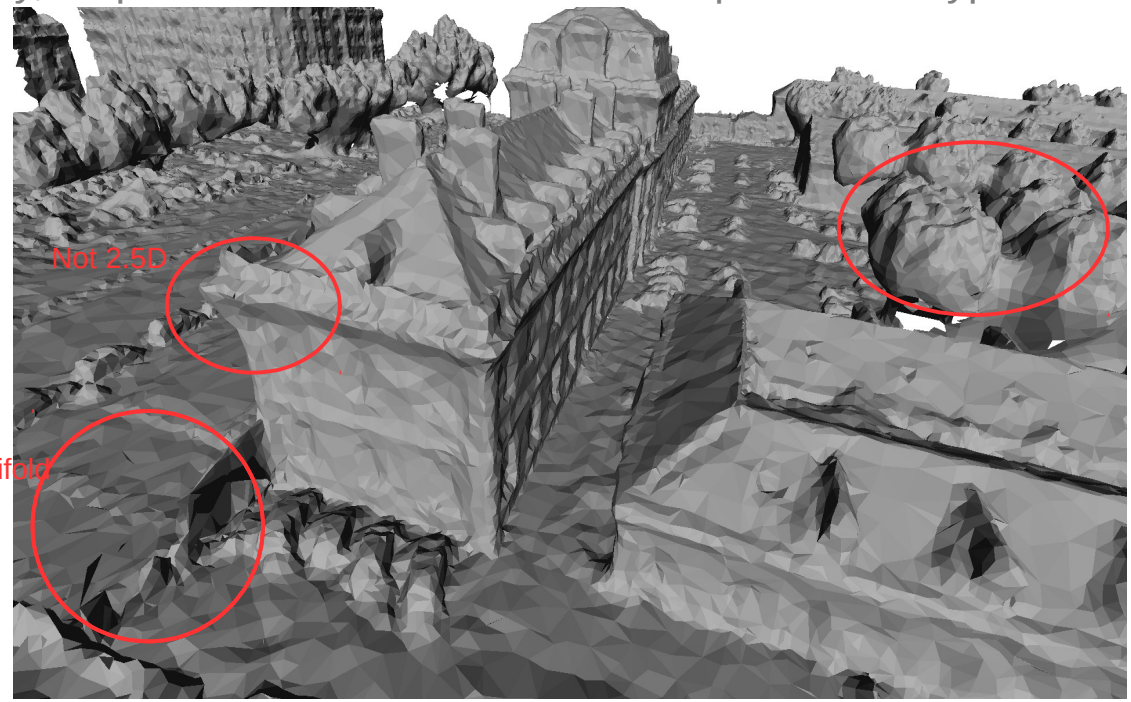
Structure aware methods

Avoid pured geometric based method because of structure alteration

Abstraction, filtering [Structure-aware shape processing 13 Mitra et al.]

Restrictive hypothesis : noise-free, 2-manifold, 2.5D, manhattan world, fully structured scene, not automated.

Unfortunately, acquired urban meshes **do not** respect these hypothesis.

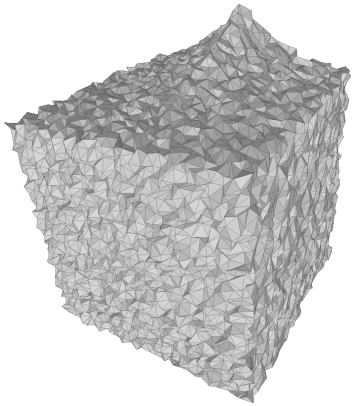


Need for automatic simplification algorithms to handle such data.

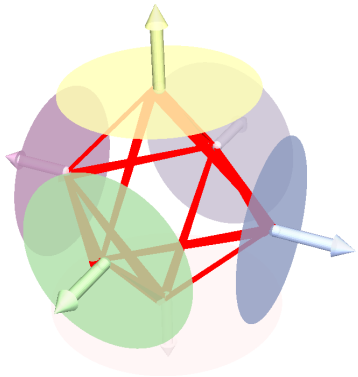
Our method

1) Detect structure

2) Simplify by iterative edge collapses while preserving detected structure

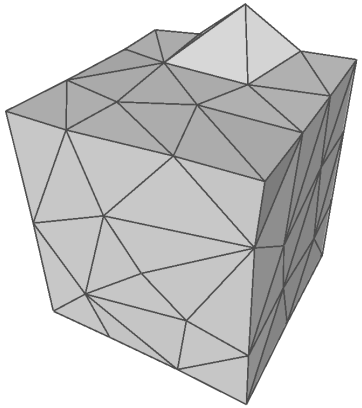


Input

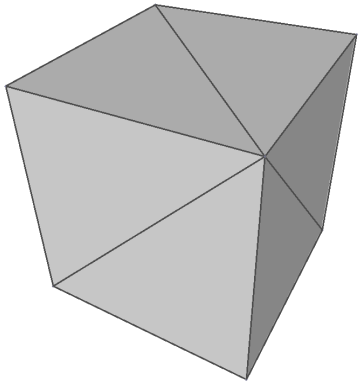


Structure detection :

- Plane detection
- Proximity graph



End of simplification



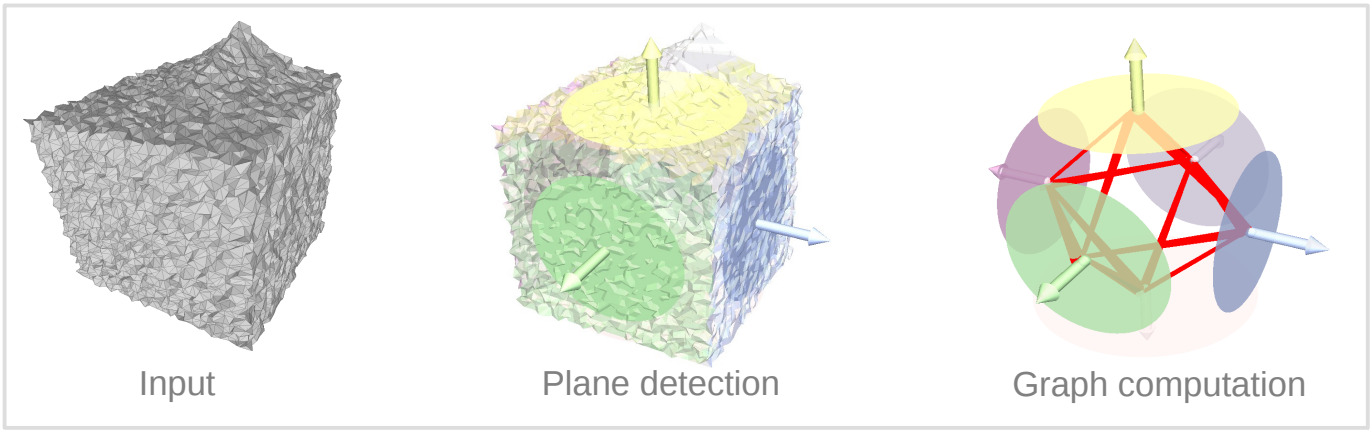
Structure-aware mesh decimation

Structure aware methods

In our case **structure** = set of planes and proximity relationship

Parts of the shape *:* Planes detected with region-growing method
(can be replaced by your favourite ransac algorithm)

Relation between the parts : Proximity relationship = a proximity graph whose vertices are planar proxies and edges are proxies that are within (euclidean set) distance lower than a threshold

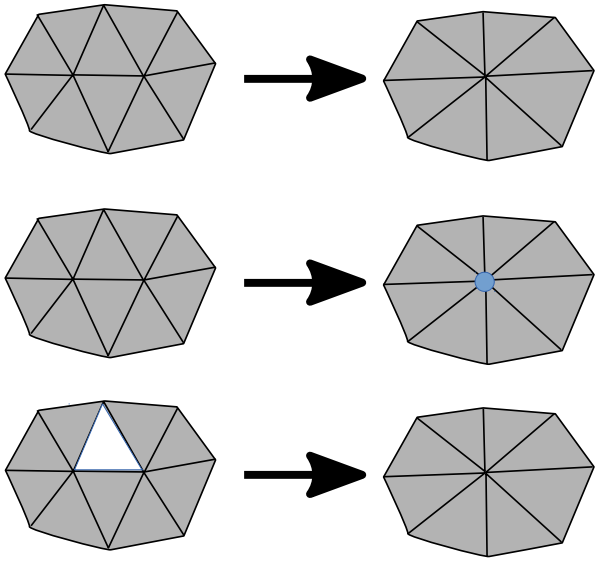


Two use of the structure :
- *geometrical* : incorporation into our error metric
- *topological* : preservation of graph properties

Decimation algorithm

Must specify :

- How much cost an edge collapse
- Where to place vertex after the edge collapse
- Say if the edge collapse is valid (may be rejected because of topological change, geometrical perturbation, ...)



How much does it cost?
e.g. estimate the deformation to your intial mesh

Where do I place the blue point?
usually at the point minimizing the cost function

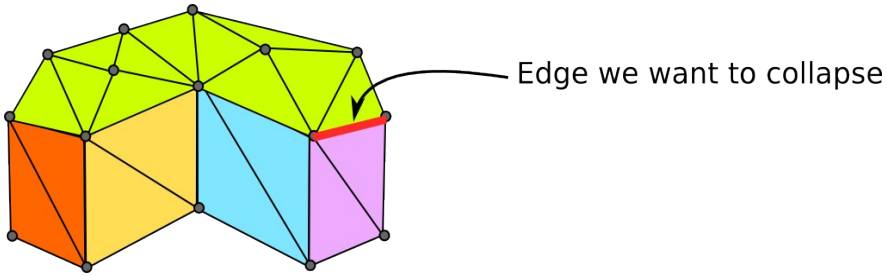
Is this a valid collapse?

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.

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



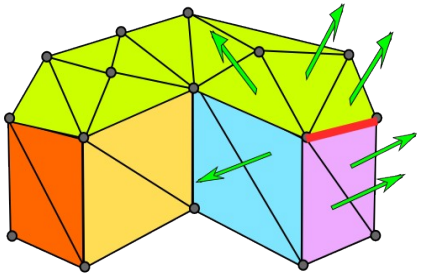
Decimation algorithm

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

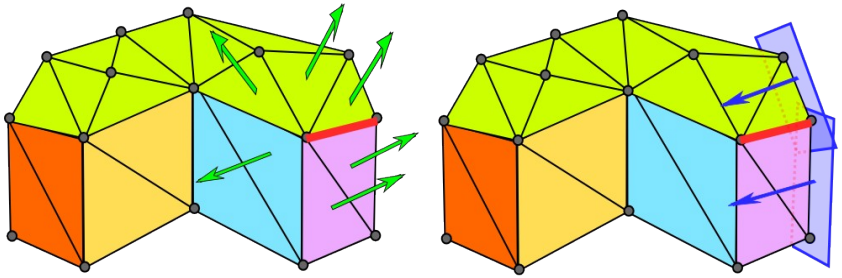
Decimation algorithm

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



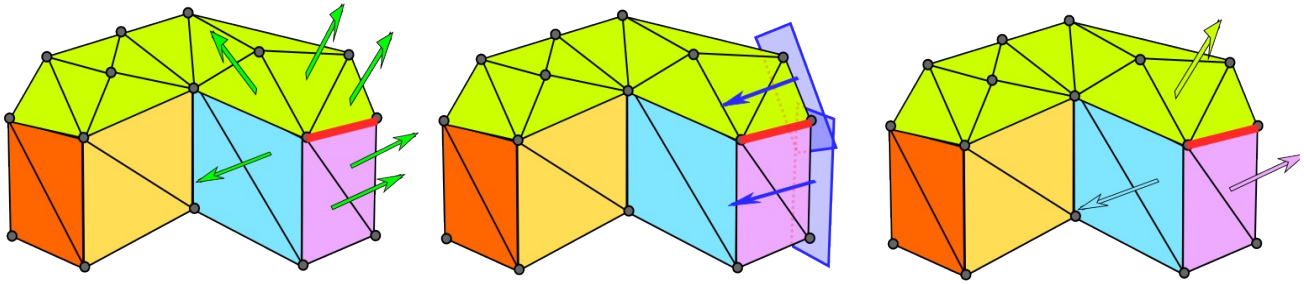
Decimation algorithm

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



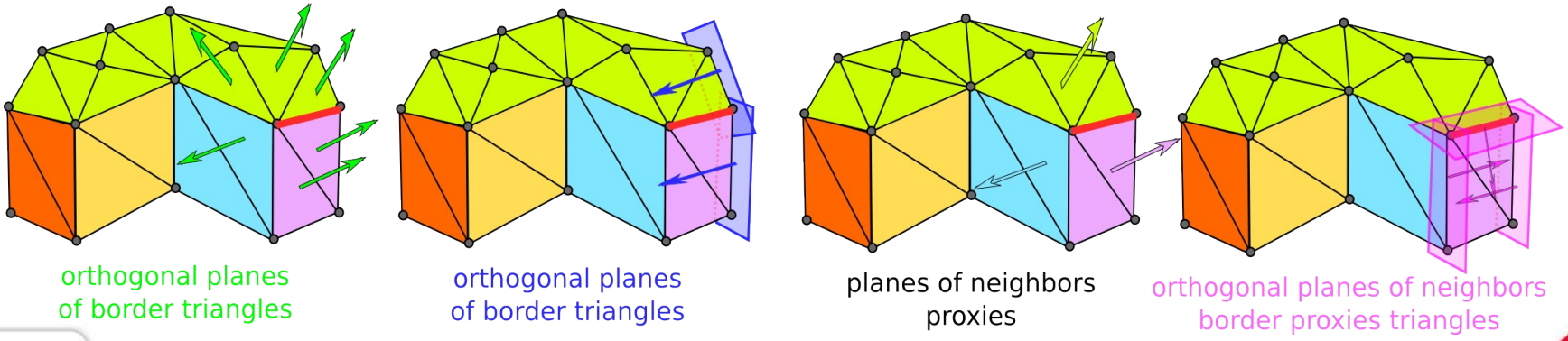
Decimation algorithm

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

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)

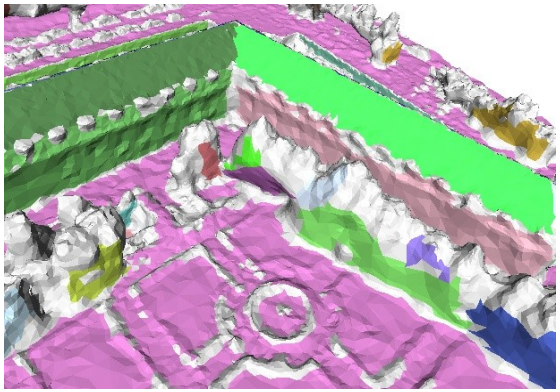
Two parameters in (0,1) :

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

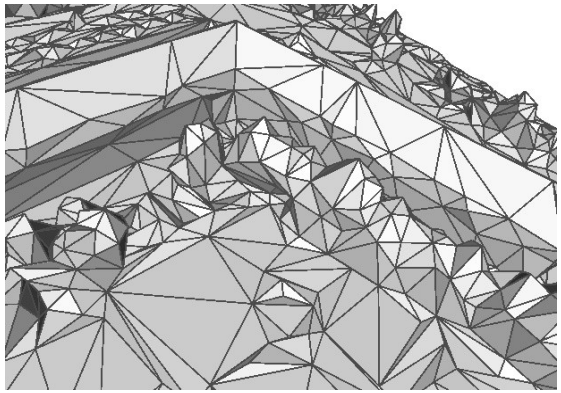
Decimation algorithm

Two parameters in (0,1) :

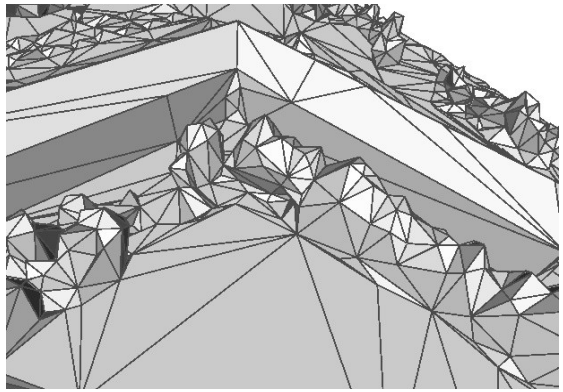
- ***Border*** parameter : trade border preservation for inner surface preservation
- ***Abstraction*** parameter : trade planar proxy proximity to initial surface proximity



Initial mesh with planar proxies



Simplified with abstr. parameter **λ=0**

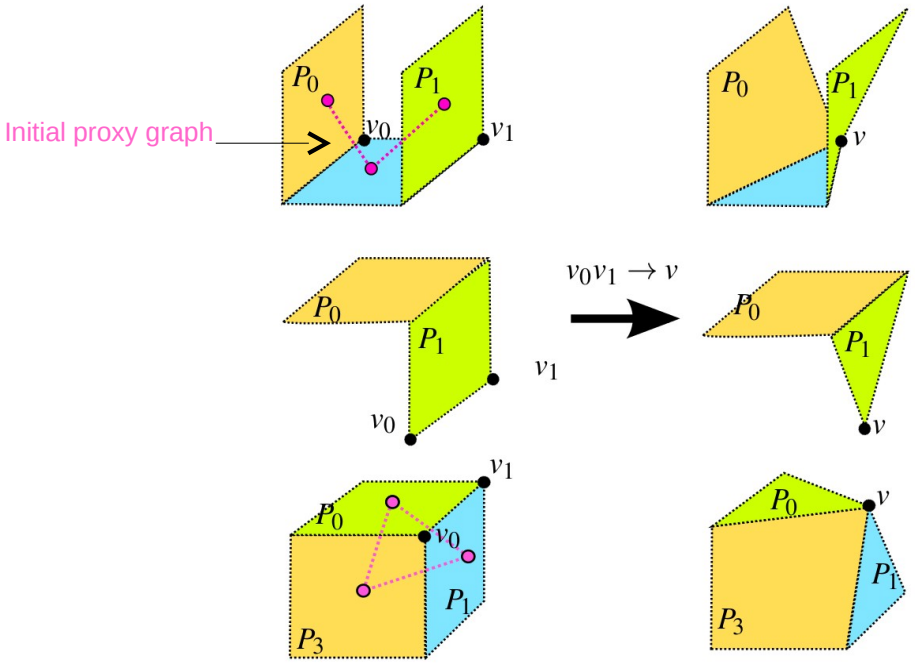


Simplified with abstr. parameter **λ=0.99**

Decimation algorithm

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 :

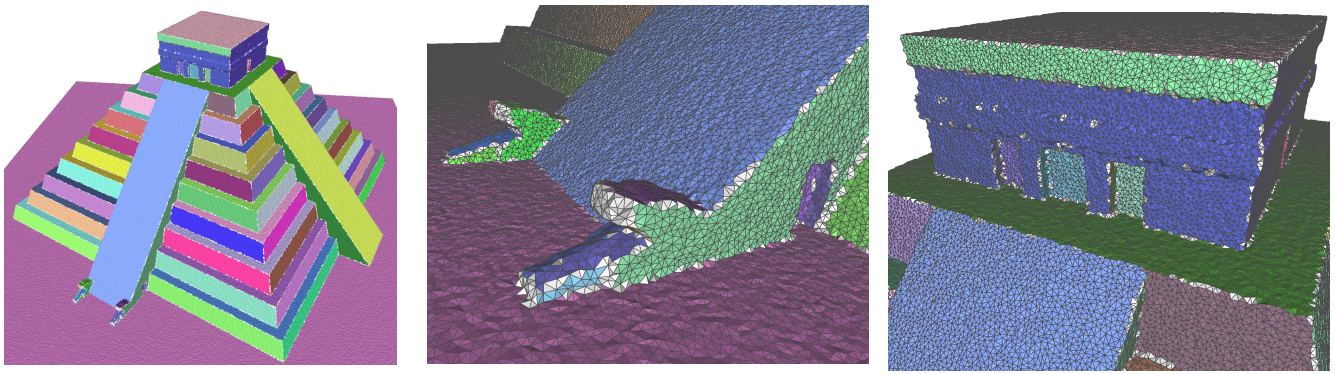
A proxy had less than 4 vertices after an edge collapse.

Corner alteration :

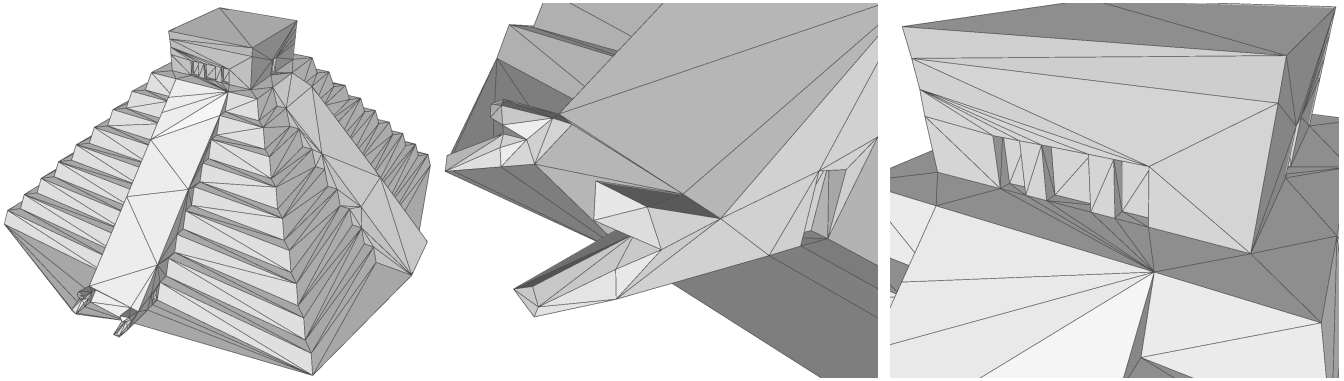
A corner is destroyed after an edge collapse.

Some results

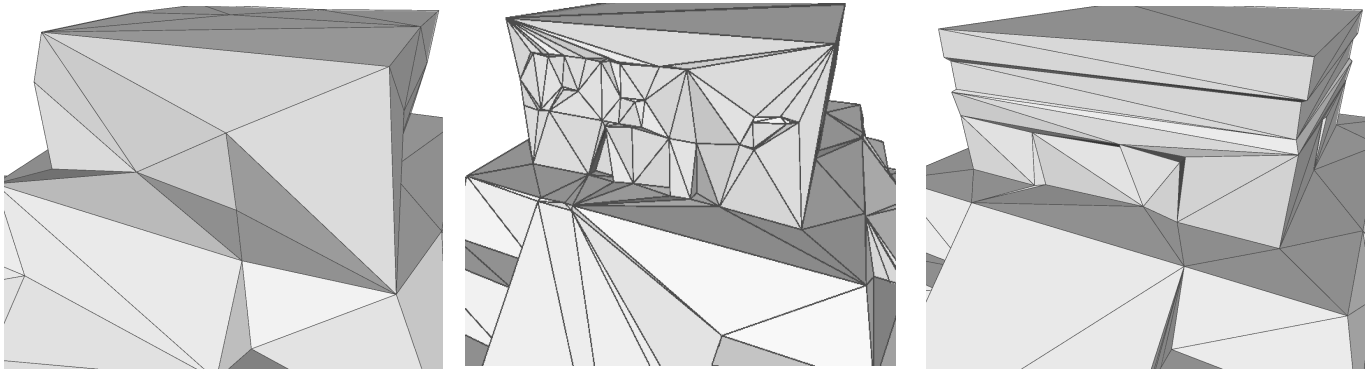
Input : 400,000 vertices



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

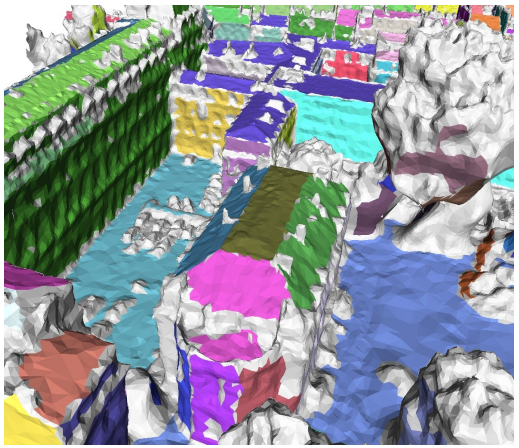


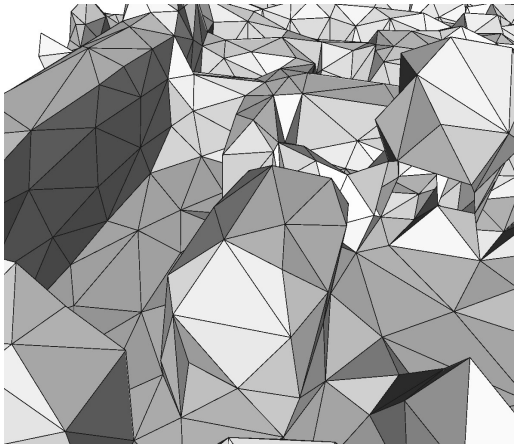
Structure unaware methods (350 or more vertices)

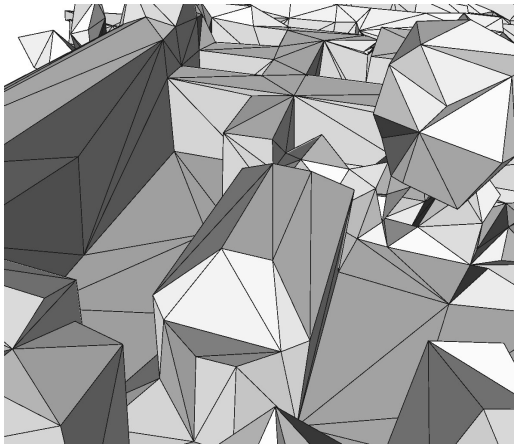


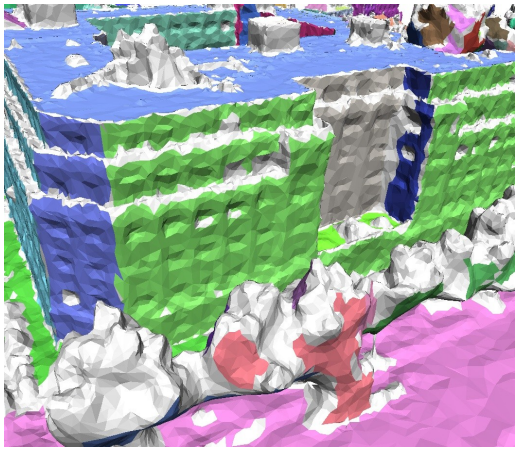
Garland Heckbert VSA Lindstrom-Turk

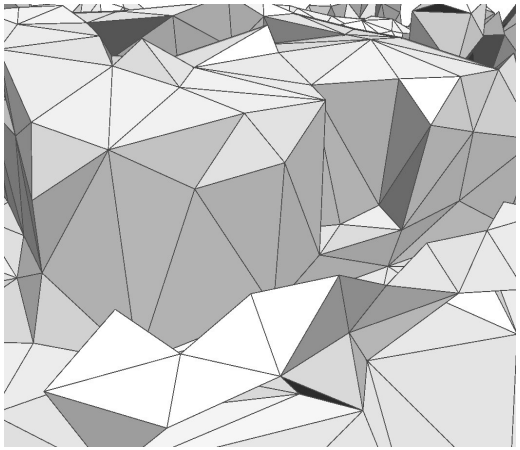
Some results

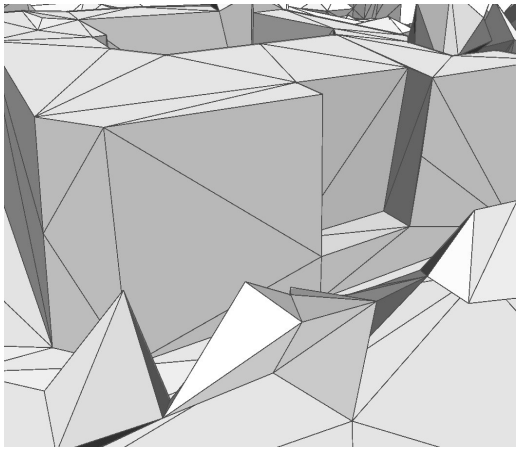










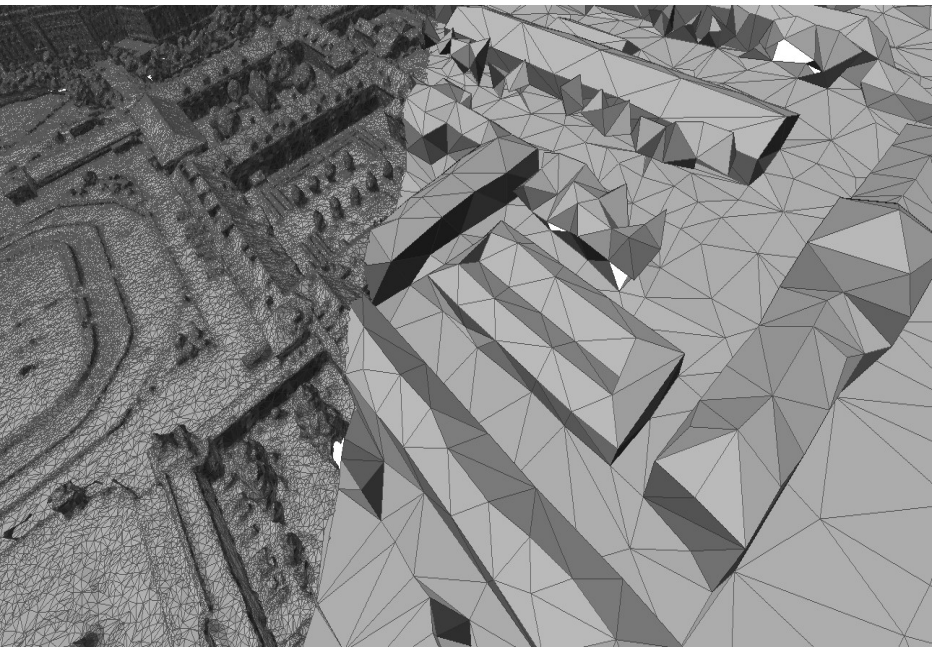


Input mesh and planar proxies

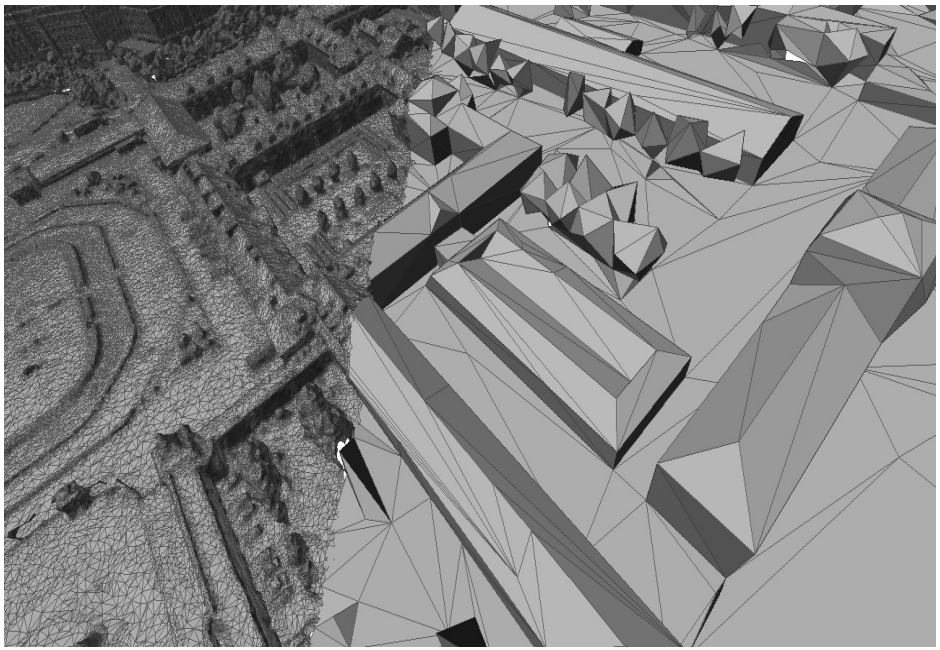
Mesh decimated with Garland-Heckbert

Mesh decimated with our method

Some results



Garland Heckbert



Ours

Generic data-structure

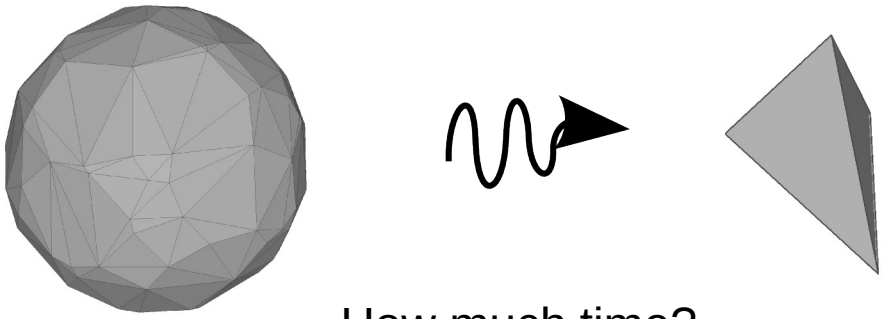
Skeleton-blocker data-structure

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

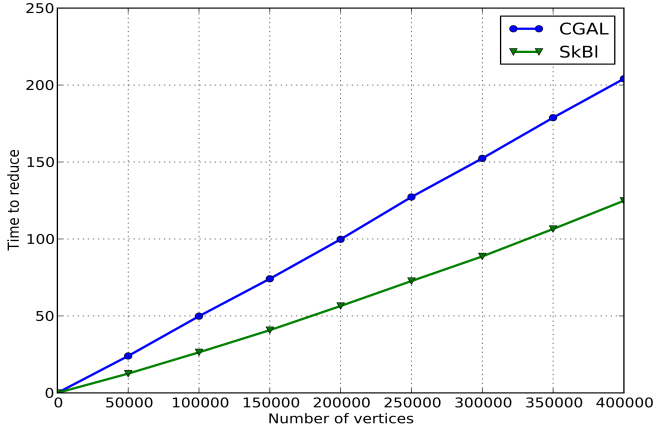
Able to represent simplicial complexes (meshes) of ***any dimension***, not necessarily ***manifold*** (contrarilty to CGAL Polyhedron_3)

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/



How much time?



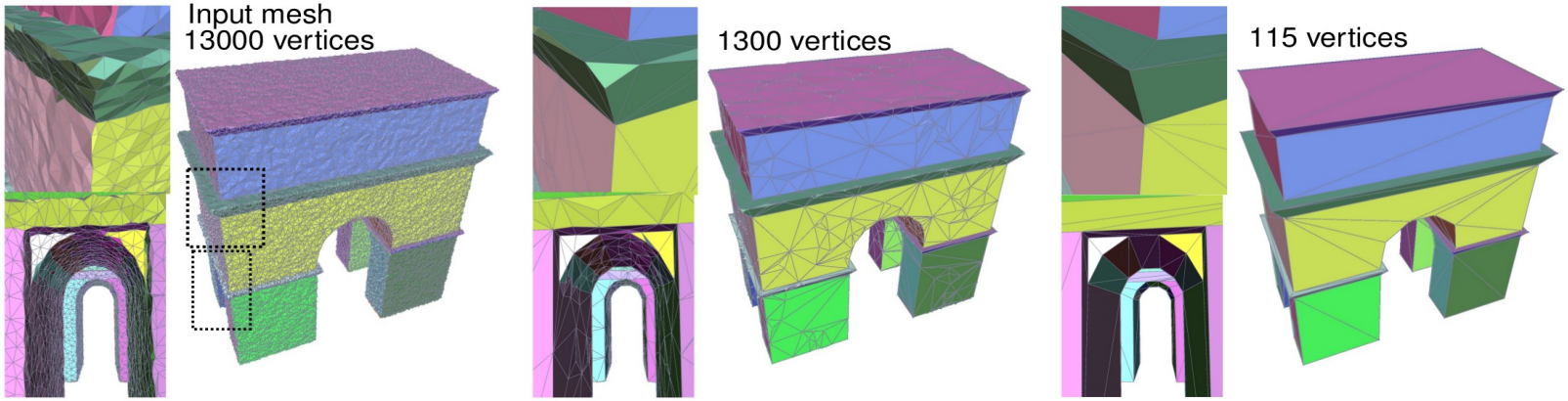
Conclusion and future works

Structure-aware mesh decimation :

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

Future works :

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



Thank you