3D Real Time Visualisation in a Rich Internet Application

Atos Wordline : Aurélien Barbier-Accary
Geomod – Liris : Raphaëlle Chaine
Geomod – Liris : Pierre-Marie Gandoin

Fabien CELLIER

Décembre 2010

An Atos Origin™ Company

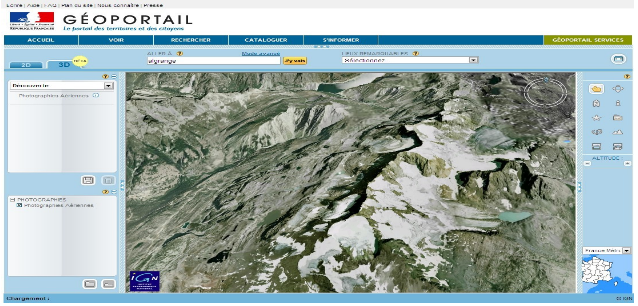
Table of Contents

Context
Atos targets

- Geographic Information System Software based on web browser

- Multi platform
  - OS
  - Architecture (PC, netbooks, smartphones…)

- For the next Geoportail
  - 30 000 users simultaneously
  - Interoperability and compatibility OGC's standards
(Open Geospatial Consortium)



Context
Research Goals

- Limited bandwidth
- Heavy load on servers
- Heterogeneous clients (software)

- *Data compression*
- *Solution scalability*
- *Algorithm efficiency for visualization*

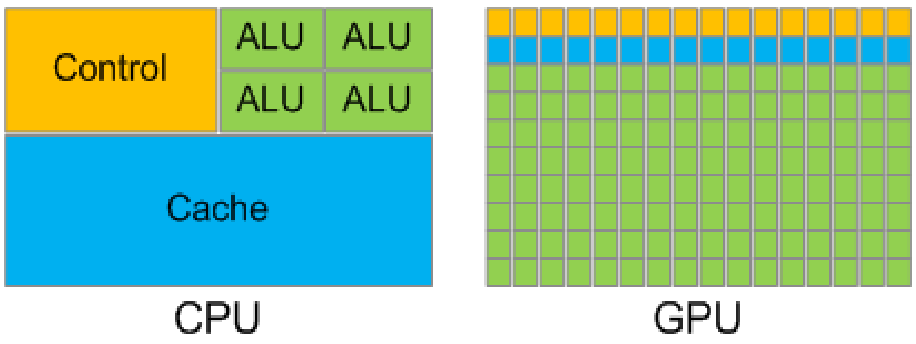
Context
Technologies

- HTML5 and WebGL:
  - **Small javascript OS embedded** in browser
    - Thread → worker
    - Filesystem → localStorage
    - Network → websocket (ajax evolution)
    - Display → canvas (2d/3d)
  - API for using GPU directly
  - Compatibility with large number of devices (nexus one, N900, iPhone...)
  - **Poor performance with the CPU** (javascript : 1700 times slower than GPU for convolution on pictures)

Context

Technologies

Parallelization with GPU :



4/8/16 cores	*512 cores*
1 global cache	1 cache per group of core

Context
Technologies

- **CPU**: **parallelization on models** : one thread for simplification, one thread for display...
- **GPU** : **parallelization on data**
- Example (OpenCL) : Adding two arrays in a third

```
__kernel void part1(__global float* a, __global float* b, __global float* c)
{
    unsigned int i = get_global_id(0); // returns thread id
    c[i] = a[i] + b[i];
}
```

- __kernel Part1 is launched with one « thread » per array element

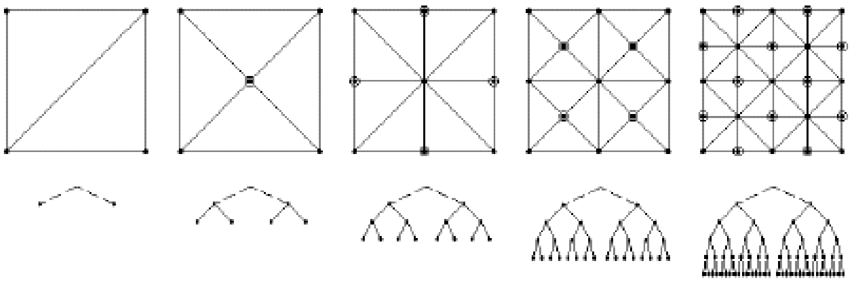
  (ratio element/thread is configured by user)

Table of Contents

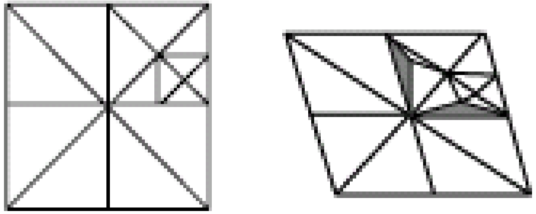
1. Context
    1. Atos targets
    2. Research goals
    3. Technologies
2. State of the art
    1. Bdam
    2. C-Bdam
    3. Hu/Hoppe
3. Observations and solutions
    1. Quadric error metric
    2. Octree and QEM
    3. Semi parallelized framework

State of the art
Bdam (P. Cignoni et al.)

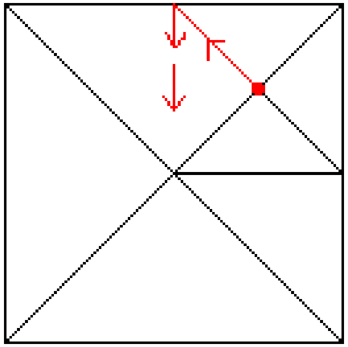
- Binary tree segmentation



- Localization based on triangular patch
- One patch has many points
  - Irregular triangulation in patches
  - Regular triangulation on borders (points which are shared by patches)

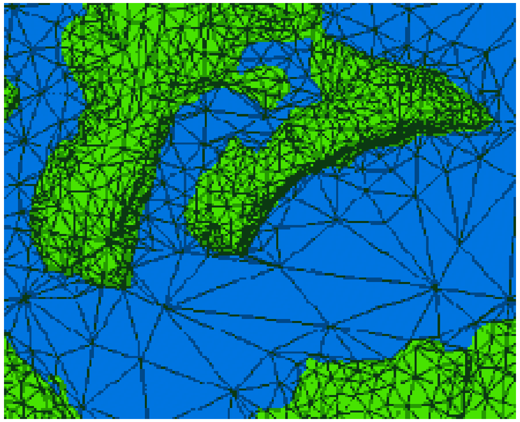


- Constraints in the tree to avoid « cracks »



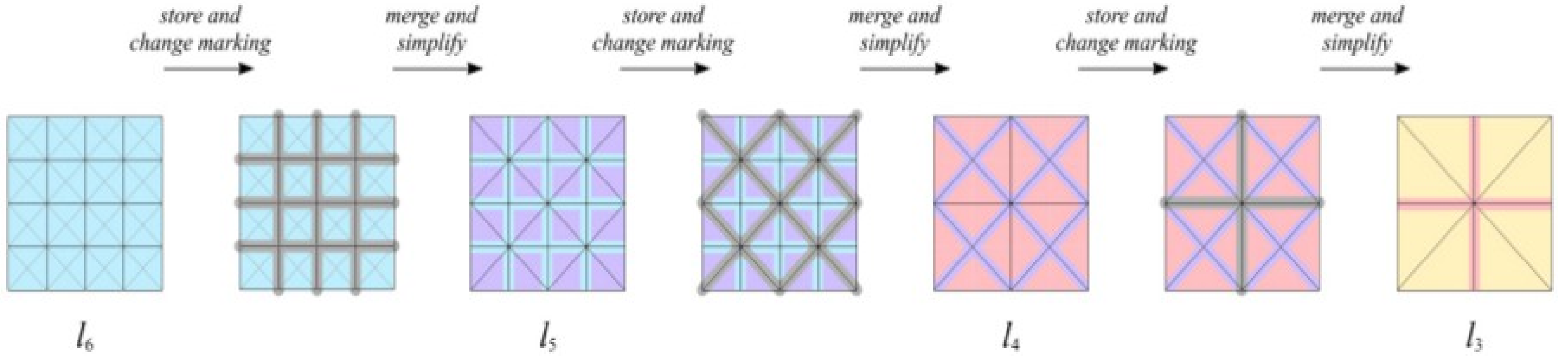
State of the art
Bdam (P. Cignoni et al.)

- 2 steps in patches :
  - Iterative edge collapse (Quadric Error Metric from Garland and Heckbert)
  - Point moves to improve triangle quality
- Iterative algorithm (can't be done in GPU)
- No streaming
  No optimization between a patch and its children (all data are resent)



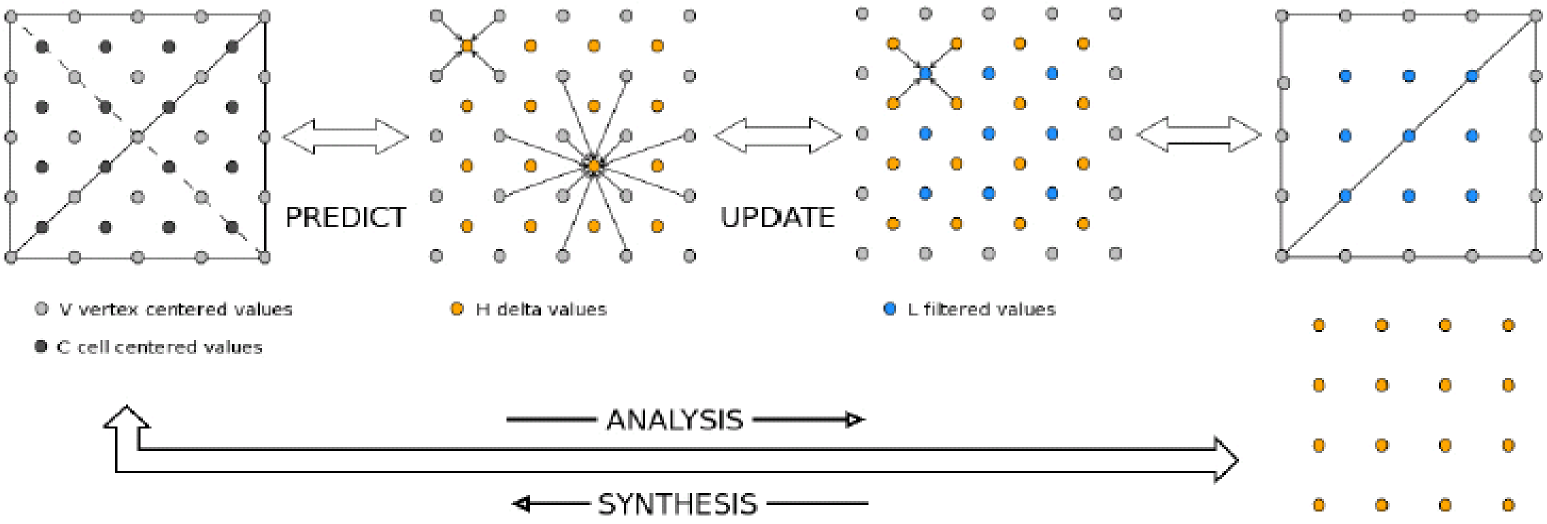
State of the art
Bdam (P. Cignoni et al.)

- Could handle 3d models (not in the paper)
- Each couple of patches is independent (and can be computed independently)



State of the art
C-Bdam (E. Gobbetti et al.)

- Same patch principle with the same constraints for the binary tree
- But regular grid in each patch
- Using wavelet to compress data



State of the art
C-Bdam (E. Gobbetti et al.)

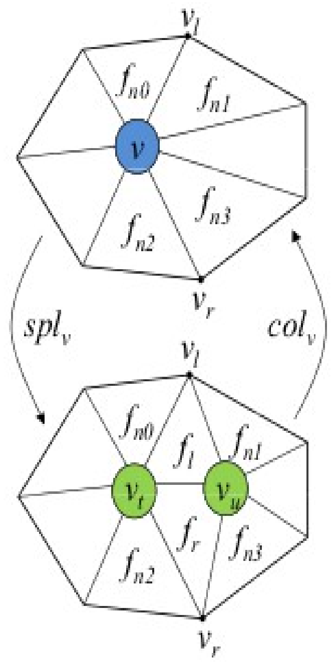
- Straightforward and efficient :
  - Regular grids
  - Integer wavelet (Neuville filter)
- Can be done in GPU
- No metric for global error (in the paper, L_∞ or L_2)

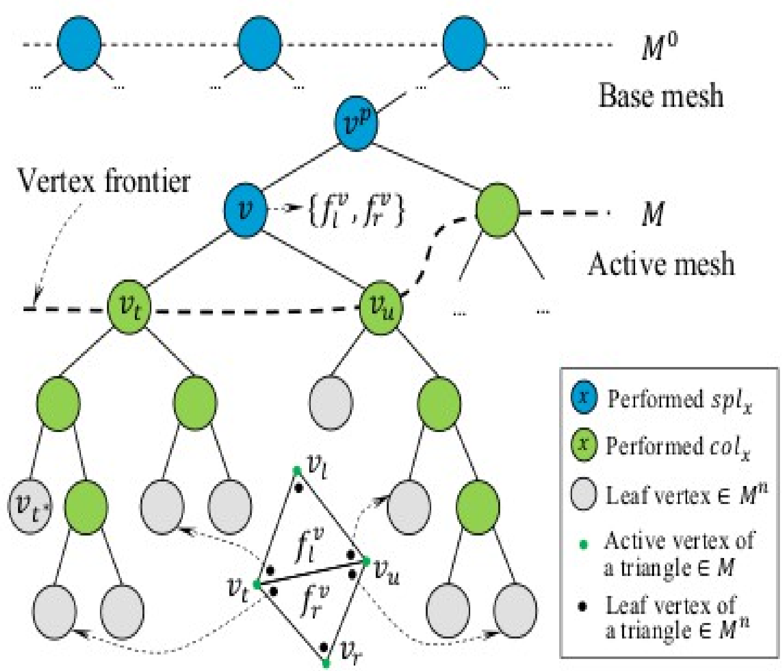
  → Need to use a zeroTree
- Can only be used on landscape

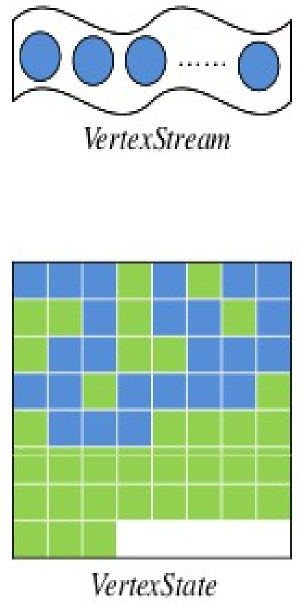
  → Neuville filter is not efficient for buildings or other 3d objects

State of the art

View Dependent Level Of Detail
L. Hu and H. Hoppe

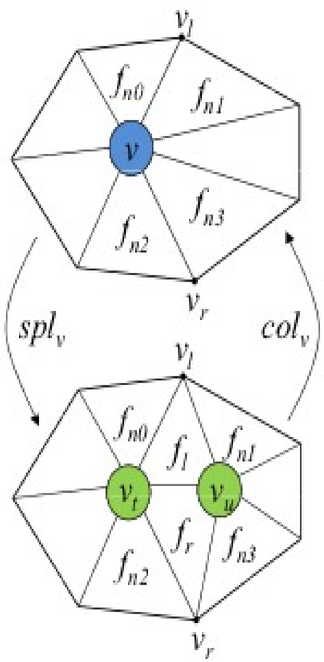






State of the art

View Dependent Level Of Detail
L. Hu and H. Hoppe

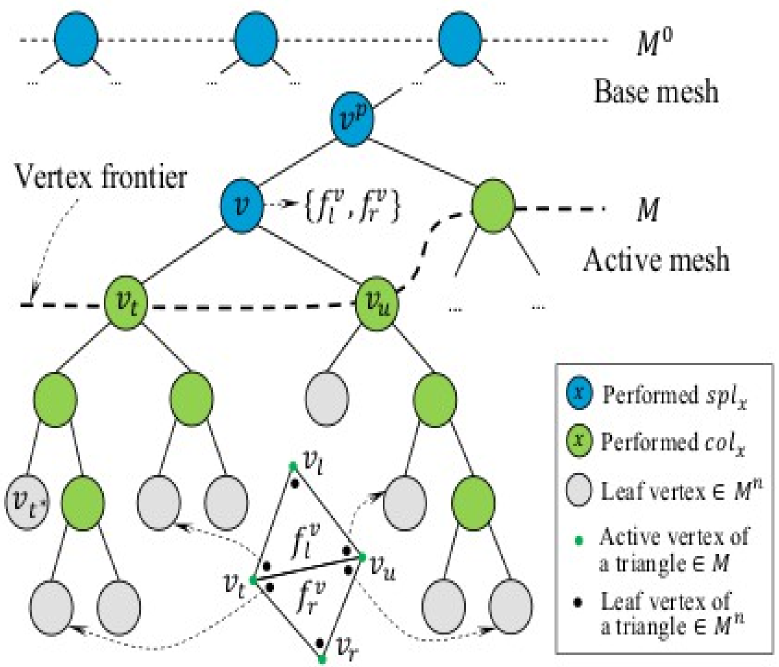


- Vertex fusion/split
- Hierarchy (Vt and Vu exist only if Vl and Vr exist)
- Use energy criterion for minimization :
  - Erep = energy associated to points (Erep decreases with the number of vertices)
  - Ep = vertex error (distance to the original vertices which generated it by fusion, with L_2 norm)
  - Edist = sum (Ep) : model error
  - Espring = sum ($||v_i-v_j||^2$) (adjacent vertices error to avoid thin triangles)

  - E=Espring+Edist+Erep

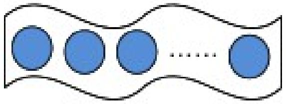
State of the art

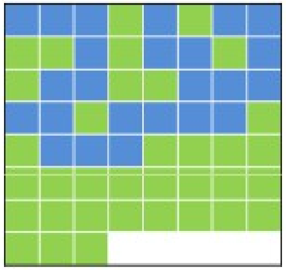
View Dependent Level Of Detail
L. Hu and H. Hoppe



- Simplification creates a binary tree (the collapse of 2 vertices creates a parent node)
- Each node has an energy
- The choice of visible nodes depends on their energy and their distance from the point of view
- Not efficient for one pass decompression

State of the art
View Dependent Level Of Detail
L. Hu and H. Hoppe


VertexStream


VertexState

- Vertices are in a common « simplification – display» memory
- 2 indices
  - One for computation
  - One for visualization
- After n steps in simplification/decompression, index swap
- The time before the next swap is predicted from the last times required to compute the n operations (to keep a good framerate)

State of the art

View Dependent Level Of Detail
L. Hu and H. Hoppe

- Can be used in all 3d models
- Not out-of-core
- Doesn't need this level of refinement for simplification (can't be used on smartphone due to hardware limitation)
- No streaming:

  Using binary tree directly? But...
  - Network latency?
  - How can we cut the model to have only some parts?
  - Storage? Ressources on servers

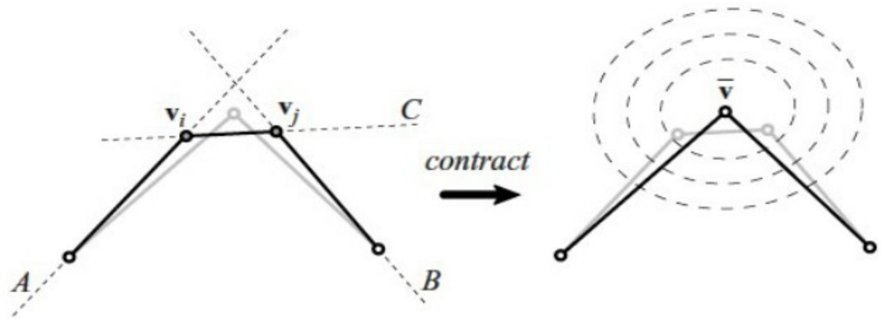
- Patches are necessary (lighweight clients) -> implies a constant number of vertices

  Remark : hierarchy is implicit in patches so we don't need it anymore

Table of Contents

Observations and solutions
QEM (Garland/Heckbert)

- Minimize square distance to plans



We define Q for each vertex (Q: 4x4 matrix)

Q contains all information of all plans attached to the vertex (quadrics)

The error for a vertex is given by: $\Delta(v) = v^T Q v$.

Where v is the coordinate of a point (x,y,z,1)

(at the beginning, the error is null)

Observations and solutions

QEM (Garland/Heckbert)

- For collapse $(v_1, v_2) \rightarrow v'$, one matrix Q' must be defined to associate a new error:

  $Q' = Q_1 + Q_2$

- We choose v' such that $\Delta(v') = v'^{T} Q' v'$ is minimal

  $\rightarrow v'^{T} Q' v' = v'^{T} (Q_1+Q_2) v' = v'^{T} Q_1 v' + v'^{T} Q_2 v'$

  so errors from Q_1 and Q_2 are saved

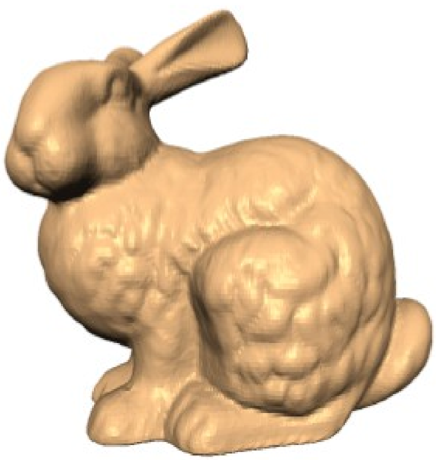
Observations and solutions
QEM and octree

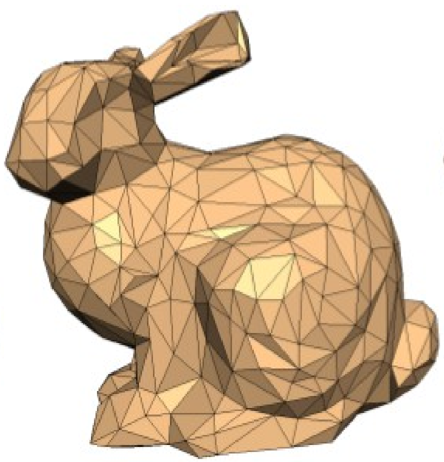
➢

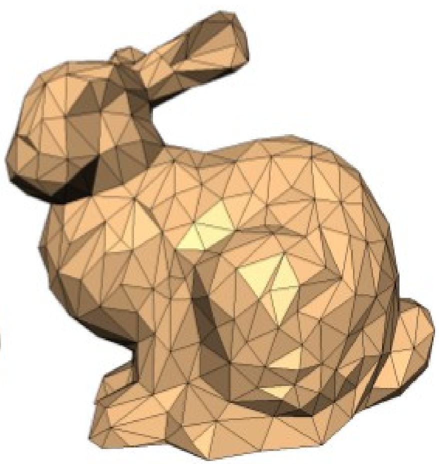
Original

Quadric error metric

Octree: fusion of vertices (with their associated Q) which are in the same cell; then simplification with Garland/Heckbert







QEM and octree

- What is important here:
  - Always keeps original geometry in memory
  - Bad choice in one fusion is not important if vertices are merged a second time → only the last choice is very influent for a given view.

  - We get no intermediate results for multiresolution visualization

Observations and solutions

Parallelization

- Our method: merging of Bdam (Cignoni) and VDLOD (Hu/Hoppe)
  - we keep the patches for distribution
  - differential coding from a patch to its children
  - one error per patch (QEM max from all vertices in the patch)
  - use QEM (quadric error metric) instead of Hoppe's metric
  - simplification with 3 goals :
    - Constant number of vertices in a patch
    - Minimize error for each vertex
    - The depth of the binary tree must be limited (GPU optimization)

Observations and solutions
semi-parallelized framework

- Having a total parallelism for the vertex fusion is not a good idea, because :
  - We need to find THE solution which minimize the global error (hard)
  - Some vertices must no be merged in order to keep error low
- Total parallelism keeps the same vertex density everywhere

Observations and solutions
semi-parallelized framework

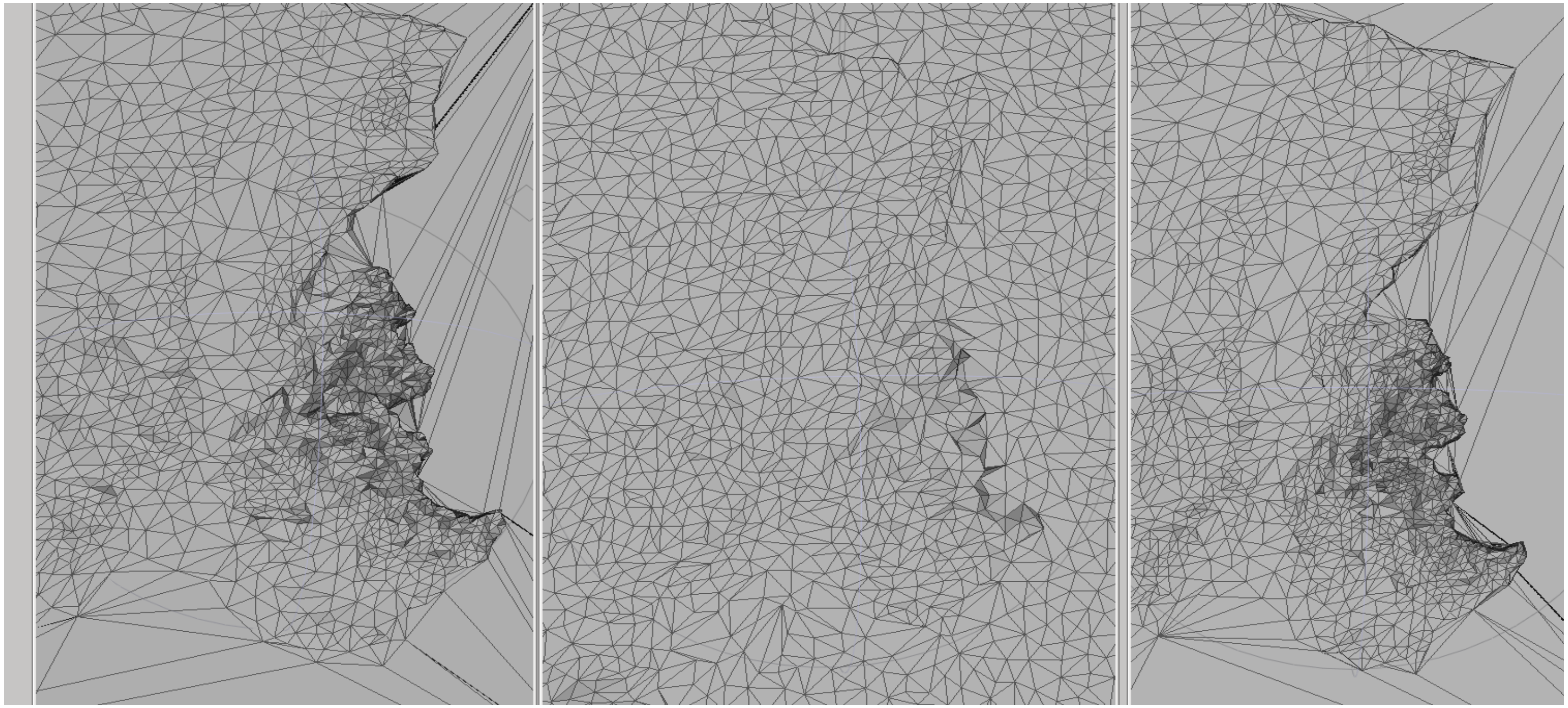
Vocabulary :

- Fusion level n : minimal level between two vertices (in the binary tree) which will merge (cf VDLOD)

- % of parallelism = % of vertices merged in each step
  - 100 % → 1 step (100 pts → 50pts)
  - 50 % → 2 steps (100 pts → 71 pts → 50 pts)

Observations and solutions
semi-parallelized framework

Garland | Fully parallel | Our method



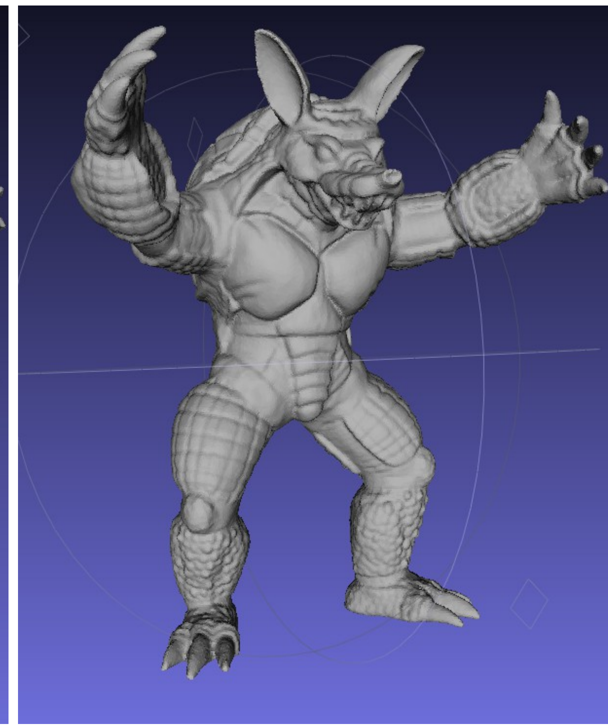
- Our method is 33% parallel in this case

Observations and solutions
semi-parallelized framework

Garland (16k faces)

Our method (16k faces)

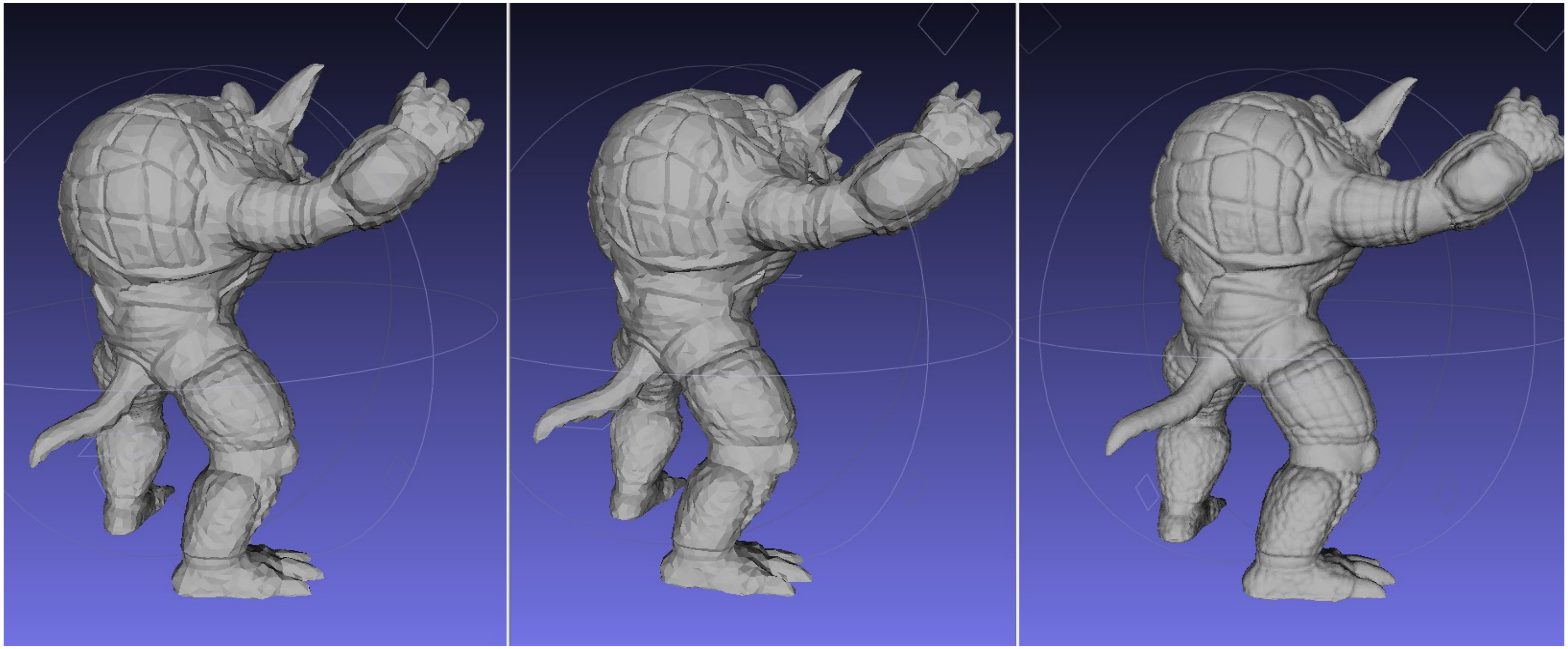
Original (1M faces)



- The point of view has been chosen to show differencies

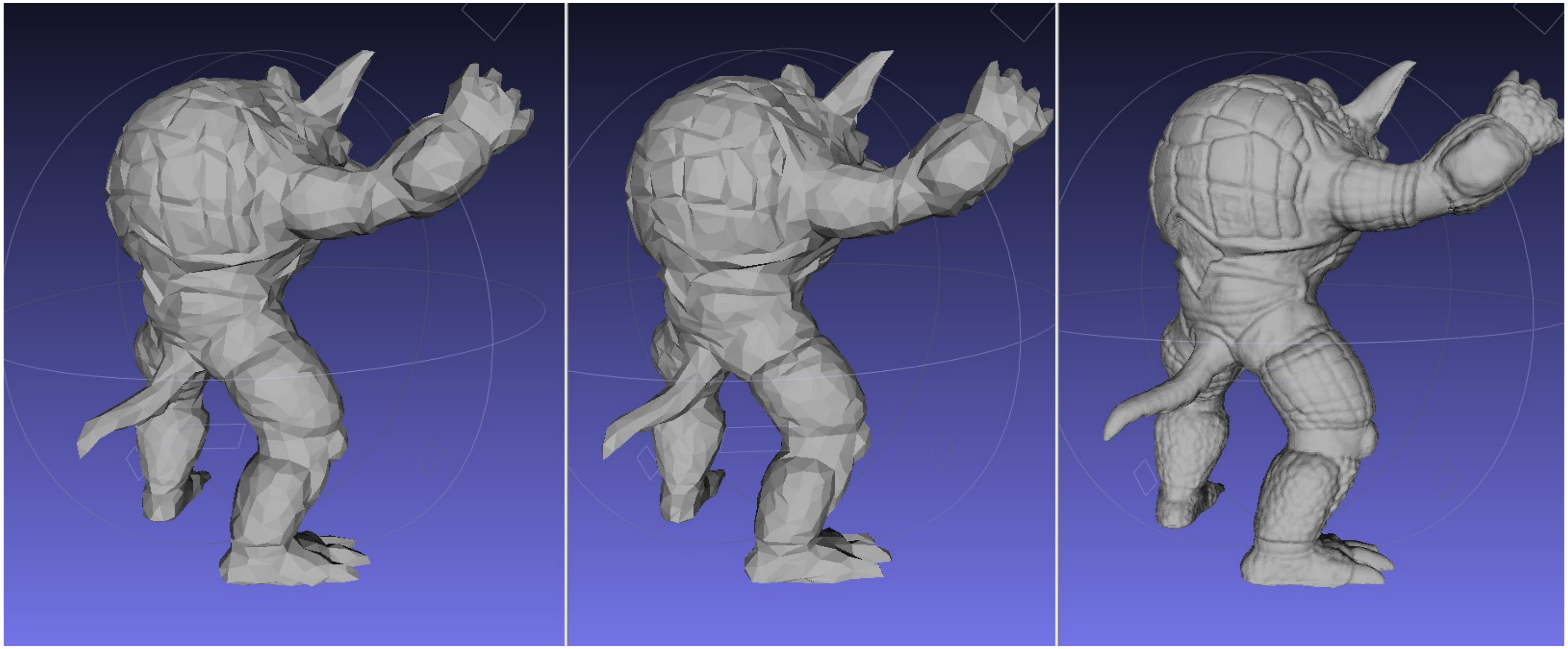
Observations and solutions
semi-parallelized framework

Garland (16k faces) | Our method (16k faces) | Original (1M faces)



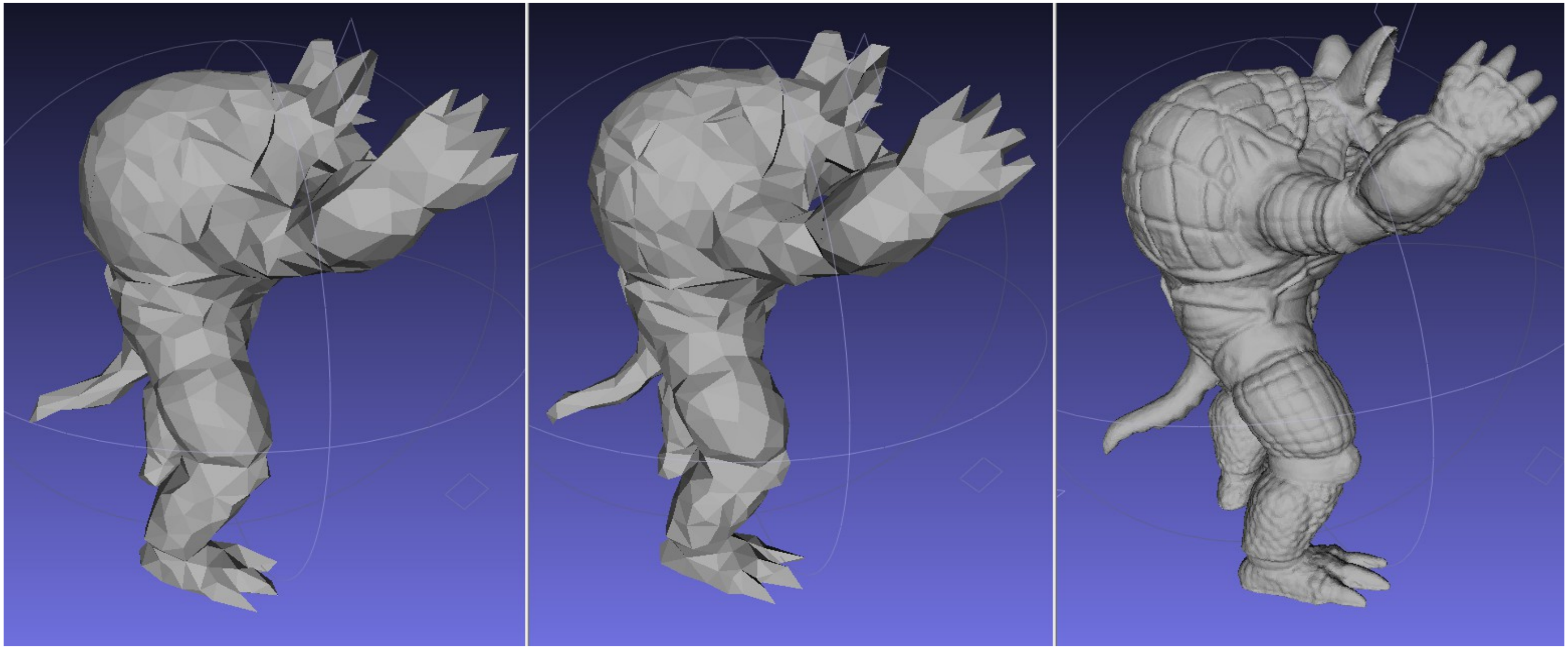
Observations and solutions
semi-parallelized framework

Garland (5k faces) | Our method (5k faces) | Original (1M faces)



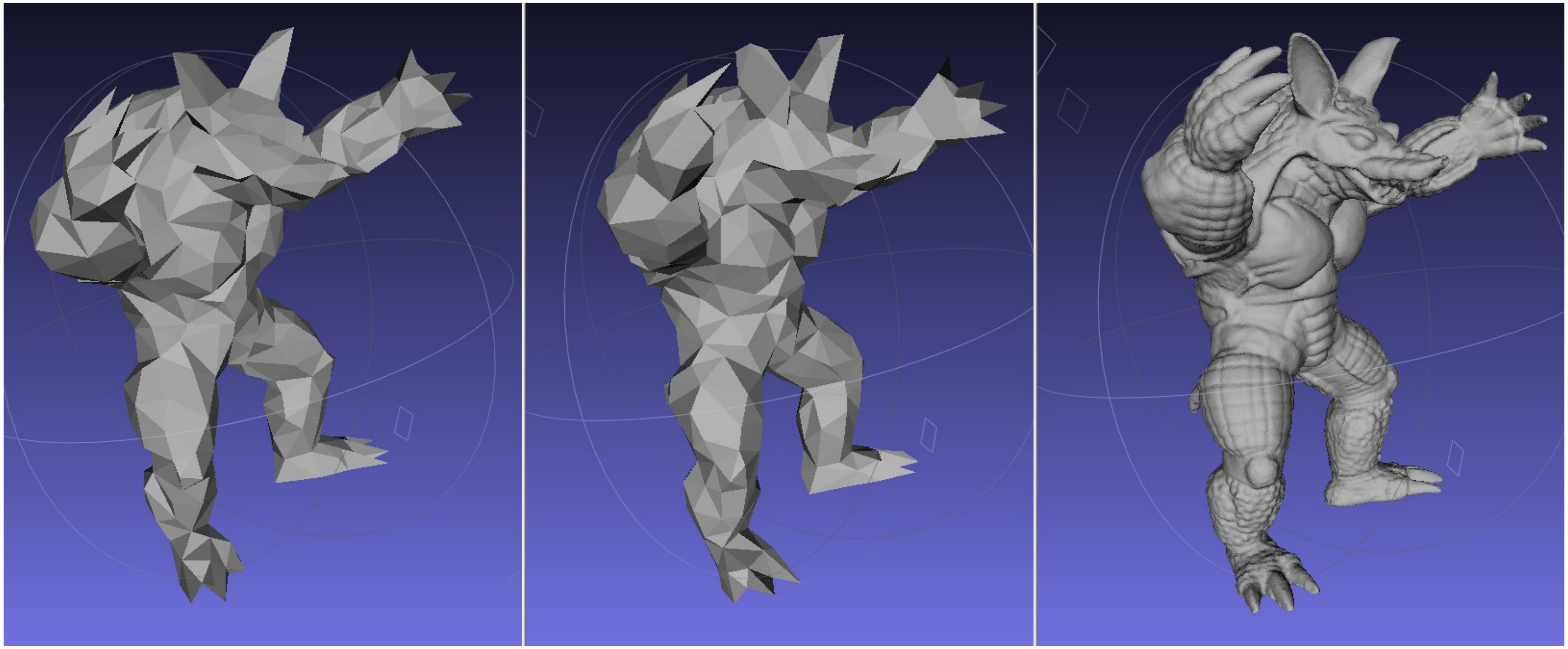
Observations and solutions
semi-parallelized framework

Garland (2k faces) | Our method (2k faces) | Original (1M faces)



Observations and solutions
semi-parallelized framework

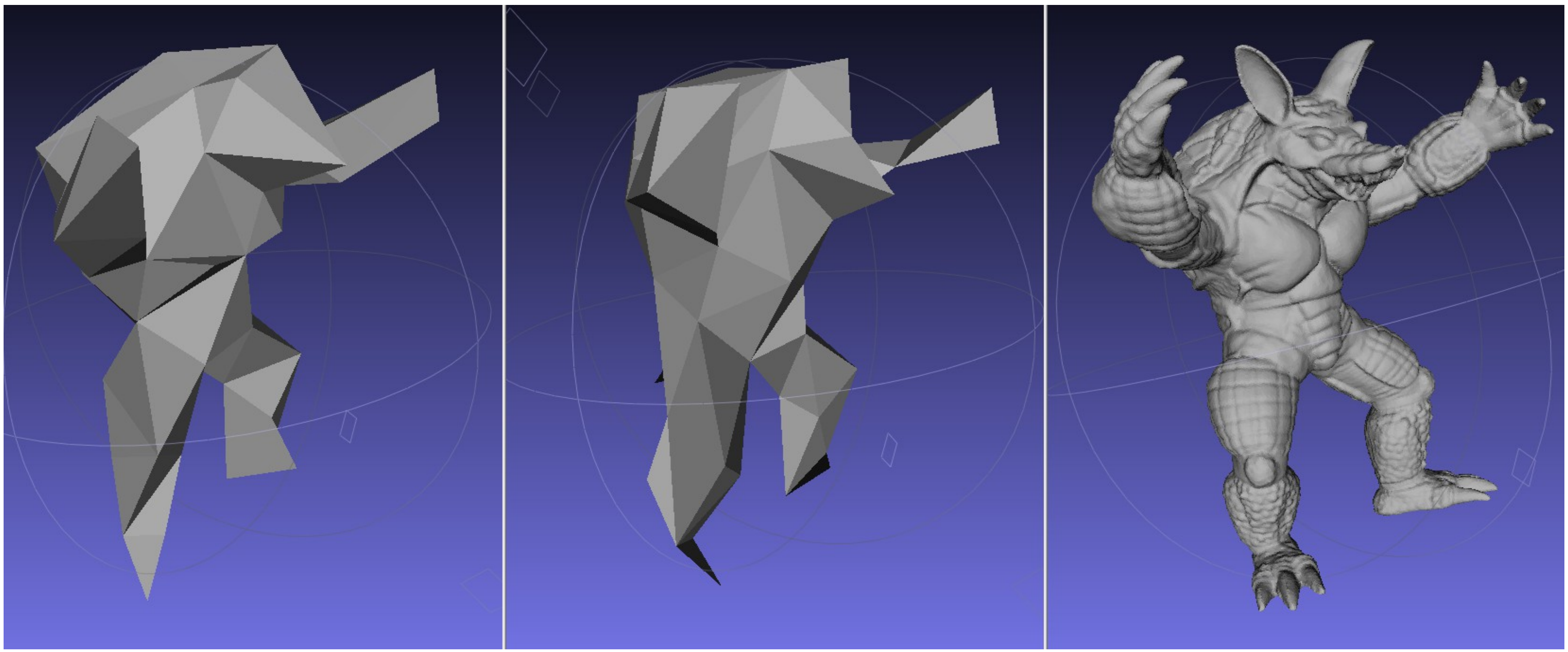
Garland (1000 faces) | Our method (1000 faces) | Original (1M faces)



- Differencies are not visible
  - In practive we try to have at most one triangle per pixel

Observations and solutions
semi-parallelized framework

Garland (80 faces) | Our method (80 faces) | Original (1M faces)



- At this stage, the number of possible choices for the fusion is too low

Observations and solutions
semi-parallelized framework

- Main differences with Hu/Hoppe :
  - out-of-core becomes possible (thanks to the patches)
  - constant number of vertices in patches → efficiency for visualization
  - patches are important for network
- Quick reconstruction (each triangle and vertex can be computed separately for visualization → GPU et HTML5 compliant)
- Each step is useful for visualization (multiresolution)
- Each patch can be computed independently

Conclusion

- An unified method for all kind of visualizations (landscape, buildings, monuments, *etc.*)

- Perspectives :
  - Find an efficient way to add texture to 3d models (easy on landscape and buildings, not for arbitrary 3d models)
  - Find a method to predict the optimal parallelism ratio according to the model

Questions

?
?
?
?
?
?
?
?
?
?
?
?