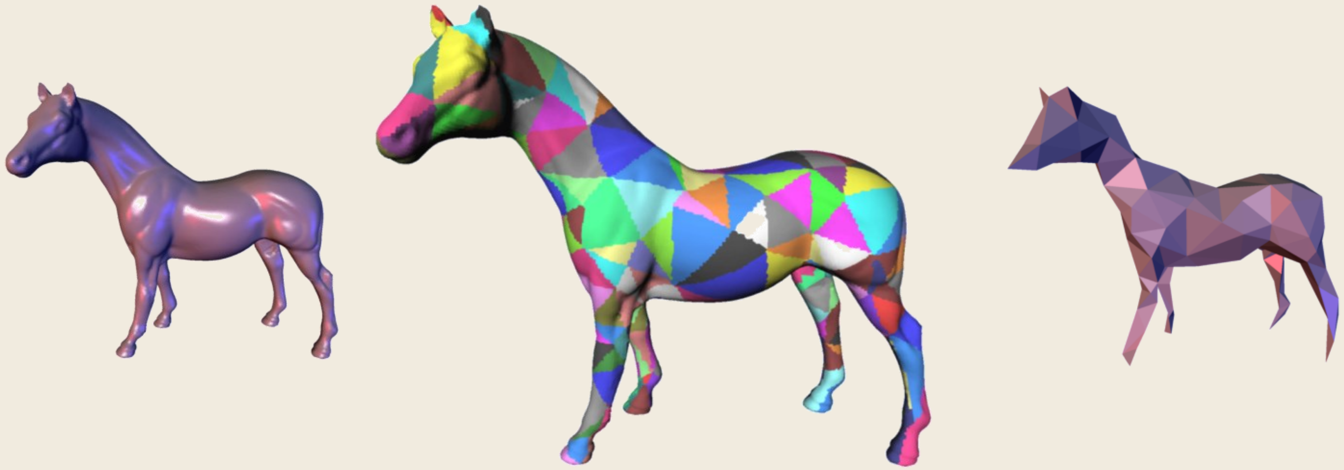
Multiresolution Adaptive Parameterization of Surfaces

Article by Aaron W. F. Lee, Princeton University - Wim Sweldens, Bell Laboratories - Peter Schröder, Caltech - Lawrence Cowsar, Bell Laboratories - David Dobkin, Princeton University

Clémentine Grethen – Alexis Gosselin – Léo Meissner – Héloïse Lafargue



I

Vertex Removal

Removing a maximally independent set of vertices with low outdegree

II

Priority queue

Vertices with small and flat 1-ring neighborhoods

III

Retriangulation

Flattening :
Conformal map &
Delaunay triangulation

IV

Decompression

Decompression method implemented

V

Results

Images, videos, quantification

VI

Issues

Encountered, solved and remaining problems

I - Vertex removal

Initialization

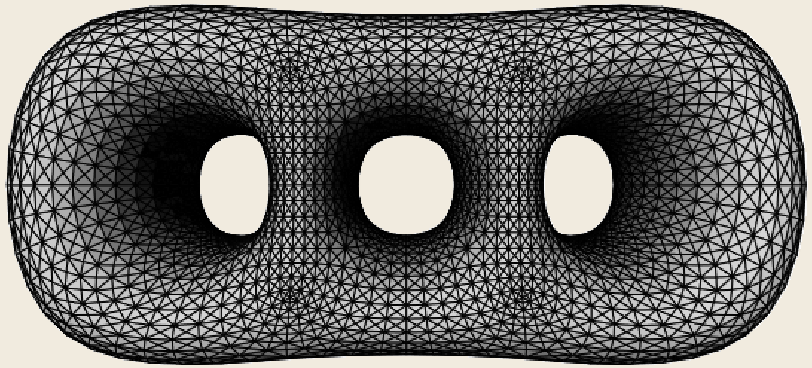
None of the vertices are marked and the set to be removed is empty
set_removed_vertices = []

Hierarchical loop

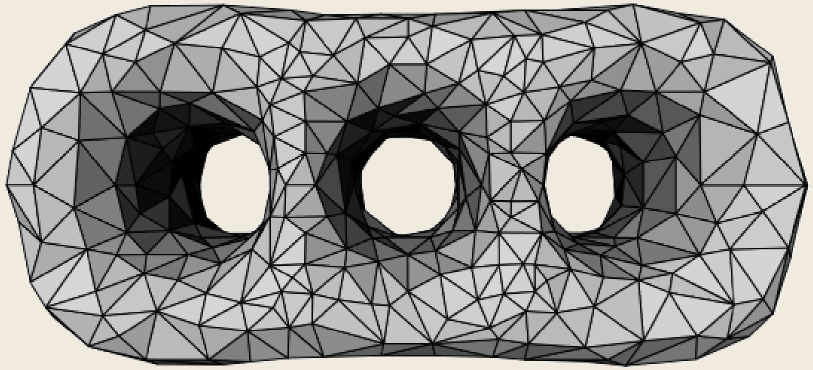
While there is a non-marked vertex of degree <12
    Selection of the 1st vertex in the priority queue
        Remove the vertex and its star from K^l
        Marks its neighbors as unremovable
        set_removed_vertices.append(vertex)

        Retriangularization
    update level

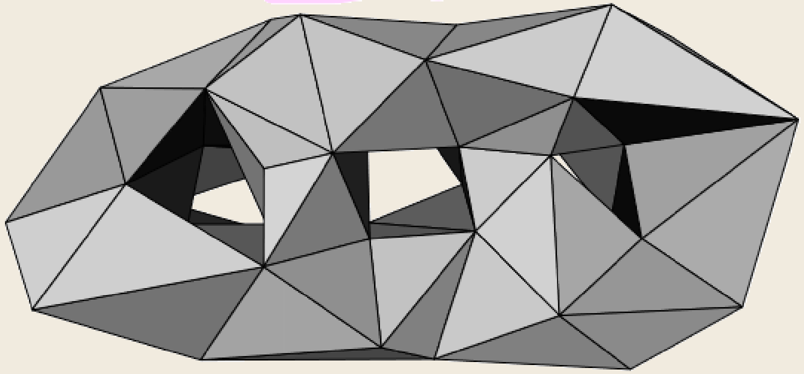
Original mesh (level 14)



Intermediate mesh (level 6)



Coarsest mesh (level 0)



II – Priority queue

The priority queue : vertices with small and flat 1-ring neighborhoods will be chosen

At level l, for a vertex $p_i \in P^l$, we consider its 1-ring neighborhood $\phi(|star(i)|)$ and compute its area a(i) and estimate its curvature κ(i). These quantities are computed relative to K^l, the current level. We assign a priority to {i} inversely proportional to a convex combination of relative area and curvature.

$$\boxed{w(\lambda, i) = \lambda \frac{a(i)}{\max_{p_i \in P^l} a(i)} + (1-\lambda) \frac{\kappa(i)}{\max_{p_i \in P^l} \kappa(i)}.}$$

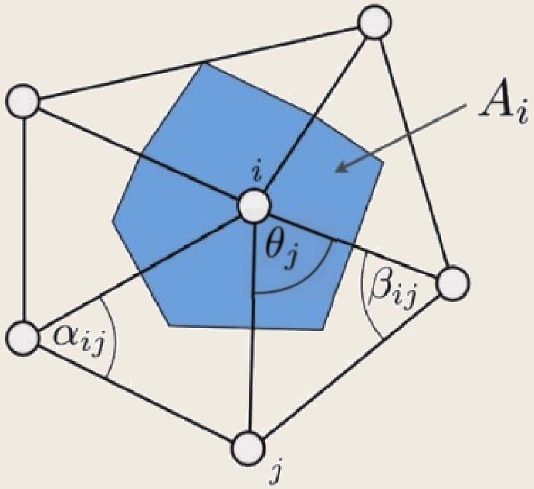
- area Ai
- connectivity
- angles θ(i)
- curvature κ(i)

→ **Sorted weitghs list w**

II – Priority queue

Curvature method 1

Curvature of a triangle mesh, definition and computation - Rodolphe Vaillant



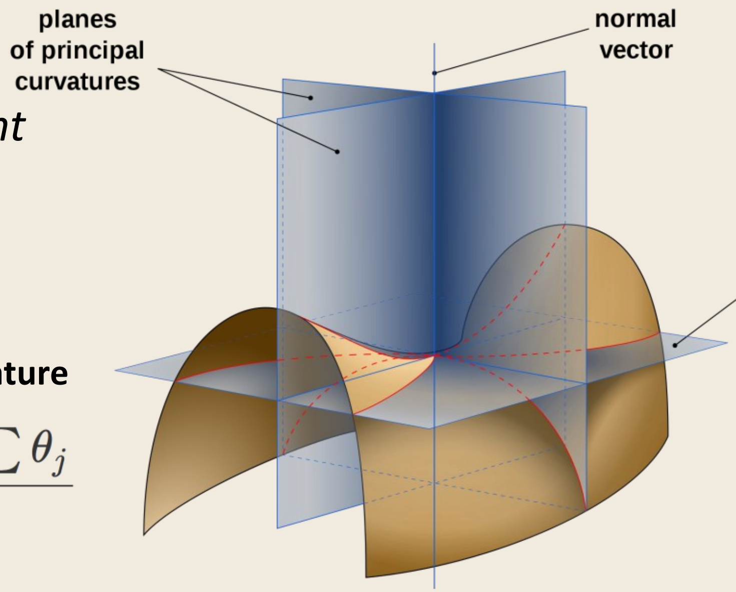
$$\kappa(i) = |\kappa_1| + |\kappa_2|$$

- **Min and max**

$$k_1 = H + \sqrt{H^2 - k_g}$$
$$k_2 = H - \sqrt{H^2 - k_g}$$

- **Gaussian curvature**

$$k_g = \frac{2\pi - \sum \theta_j}{A_i}$$



- **Voronoï area**

$$A_i = \frac{1}{3} \sum_{T_j \in \mathcal{N}(i)} area(T_j)$$

- **Mean curvature**

$$|H| = \frac{\|\Delta p_i\|}{2}$$

sign of the mean-curvature H

$$dot(\vec{n}_i, -\Delta p_i)$$

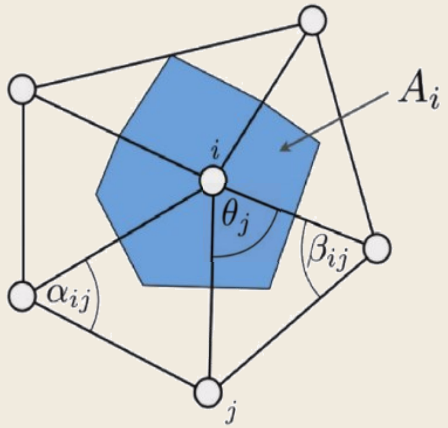
- **Discrete Laplace operator**

$$\Delta \vec{p}_i = \frac{1}{2A_i} \sum_{j \in \mathcal{N}(i)} (\cot \alpha_{ij} + \cot \beta_{ij})(p_j - p_i)$$

$$\Delta \vec{p}_i = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} 1. (p_j - p_i)$$

II – Priority queue

Curvature method 2
PC-MSDM: A quality metric for 3D point clouds - Gabriel Meynet, Julie Digne, Guillaume Lavoué



- Local approximation of surface near 3D points
- Computation of mean curvature
- Steps:
  - Principal Component Analysis for approximate an orthonormal frame (done with SVD)
  - Local least squares fitting of a quadric surface $Q(x,y) = ax^2 + by^2 + cxy + dx + ey + f$
  - That minimizes: $\sum_i \|z_i - Q(x_i, y_i)\|^2$
  - Derivatives of the quadric surface's coefficients used to estimate the curvature

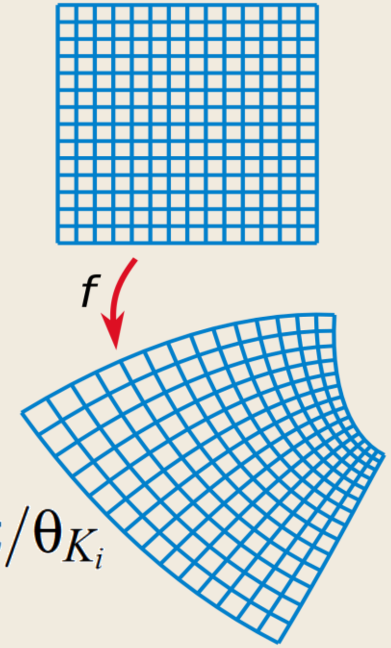
$$Curv(p) = \frac{(1+d^2)a + (1+e^2)b - 4abc}{(1+e^2+d^2)^{\frac{3}{2}}}$$

III - Retriangulation

➔ We need to retriangulate the holes left by removing the independent set.

➔ Use the **conformal map** z^a which minimizes metric distortion to **map the neighborhood of a removed vertex into the plane**.

Let {i} be a vertex to be removed. Enumerate cyclically the K_i vertices in the 1-ring $N(i) = \{ j_k \mid 1 \leq k \leq K_i \}$ such that $\{ j_{k-1}, i, j_k \} \in K^l$ with $j_0 = j_{K_i}$. A piecewise linear approximation of z^a, which we denote by μ_i, is defined by its values for the center point and 1-ring neighbors; namely, $\mu_i(p_i) = 0$ and $\mu_i(p_{jk}) = r_k^a \exp(i\theta_k a)$, where :



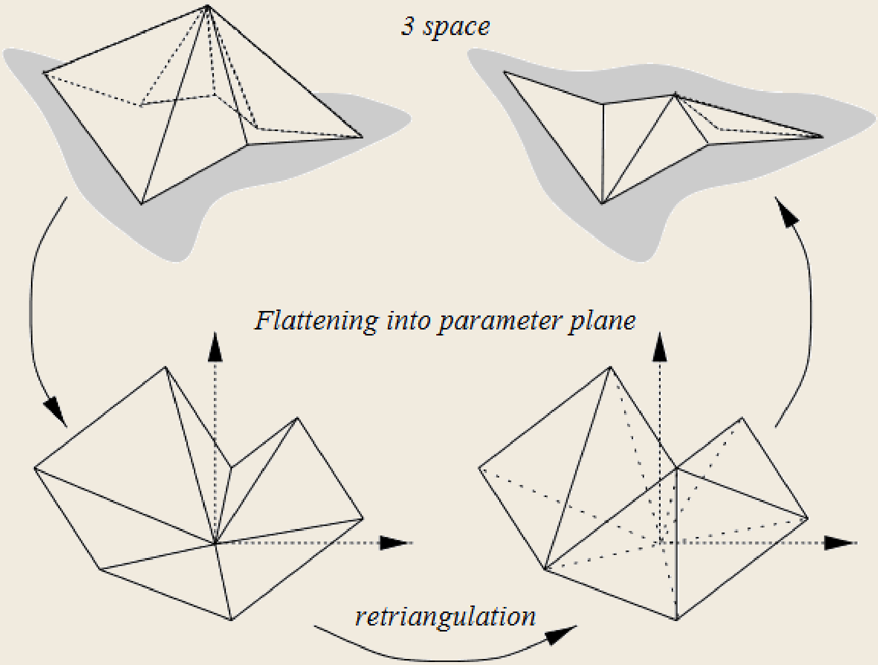
$$r_k = \|p_i - p_{j_k}\| \qquad \theta_k = \sum_{l=1}^{k} \angle(p_{j_{l-1}}, p_i, p_{j_l}) \qquad a = 2\pi/\theta_{K_i}$$

➔ When the vertex to be removed is a **boundary vertex**, we map to a half disk by setting $a = \pi/\theta_{K_i}$

III - Retriangulation

- ➔ **Advantages** of the conformal map:
  always exists + easy to compute + minimizes metric distortion + bijection

- ➔ Once the 1-ring is flattened, we can retriangulate the hole using, with a **constrained Delaunay triangulation**. This tells us how to build K^{l-1} .



In order to remove a vertex p_i , its star (i) is mapped from 3-space to a plane using the map z^a .
In the plane the central vertex is removed and the resulting hole retriangulated (bottom right).

III - Decompression

While the number of vertices > desired number of vertices:

- Calculate 'w' for each vertex
- While 'w' is not empty:
    - Retrieve the vertex with the highest weight
    - Lock neighboring vertices
    - Retriangulate the faces associated with the vertex
    - Populate the operations list
- Retrieve faces that have not been retriangulated in this iteration to integrate into the model
- Reindex the vertices and faces of the new model

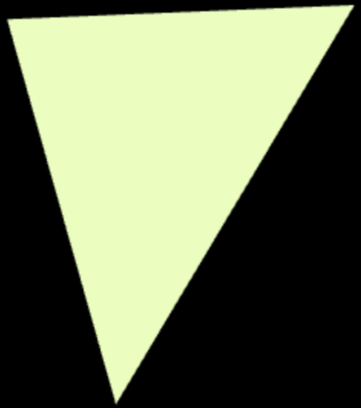
Add the last vertices and faces to the operations list

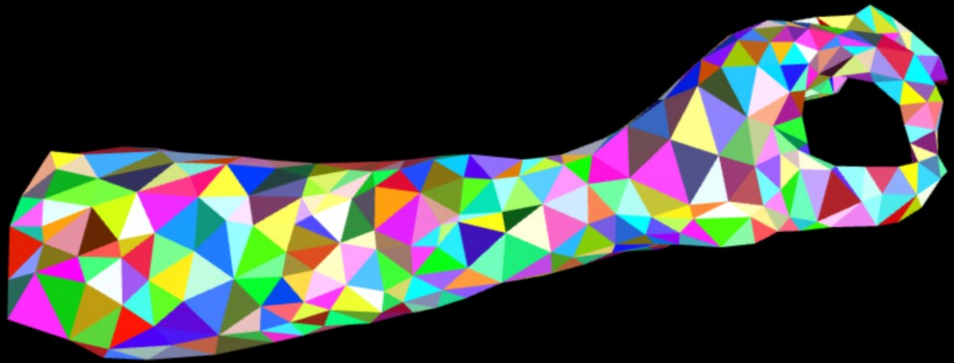
Reverse operations

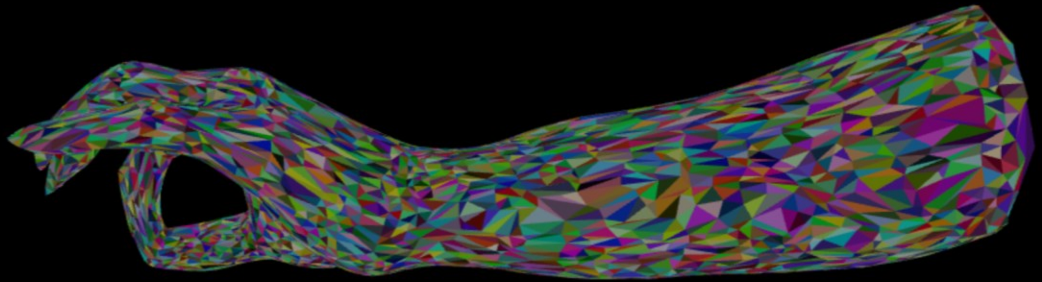
Create the model

```python
class MappingFaceIndex:
    def __init__(self, faces):
        self.faces = faces
        self.face_indices = {}

    def assign_indices(self):
        return {self.face_to_tuple(face): index for index, face in enumerate(self.faces)}

    def init_indices(self):
        self.face_indices = self.assign_indices()

    def face_to_tuple(self, face):
        return (face.a, face.b, face.c)

    def modify_face(self, old_face, new_face):
        # Trouver l'indice de l'ancienne face
        old_face_tuple = self.face_to_tuple(old_face)
        if old_face_tuple in self.face_indices:
            index = self.face_indices[old_face_tuple]

            # Remplacer l'ancienne face par la nouvelle dans la liste
            self.faces[index] = new_face

            # Mettre à jour le dictionnaire de mapping
            del self.face_indices[old_face_tuple]
            new_face_tuple = self.face_to_tuple(new_face)
            self.face_indices[new_face_tuple] = index
        else:
            raise ValueError("L'ancienne face n'existe pas dans le modèle.")

    def add_face(self, face, index_face):
        # Ajouter la face à la liste
        #self.faces.append(face)
        compteur = 0
        if index_face < len(self.faces):
            while compteur < len(self.faces):
                if self.faces[compteur] == face:
                    break
                compteur += 1
            if compteur == len(self.faces):
                self.faces.append(face)
        else:
            self.faces[index_face] = face
```

IV - Results







IV - Results

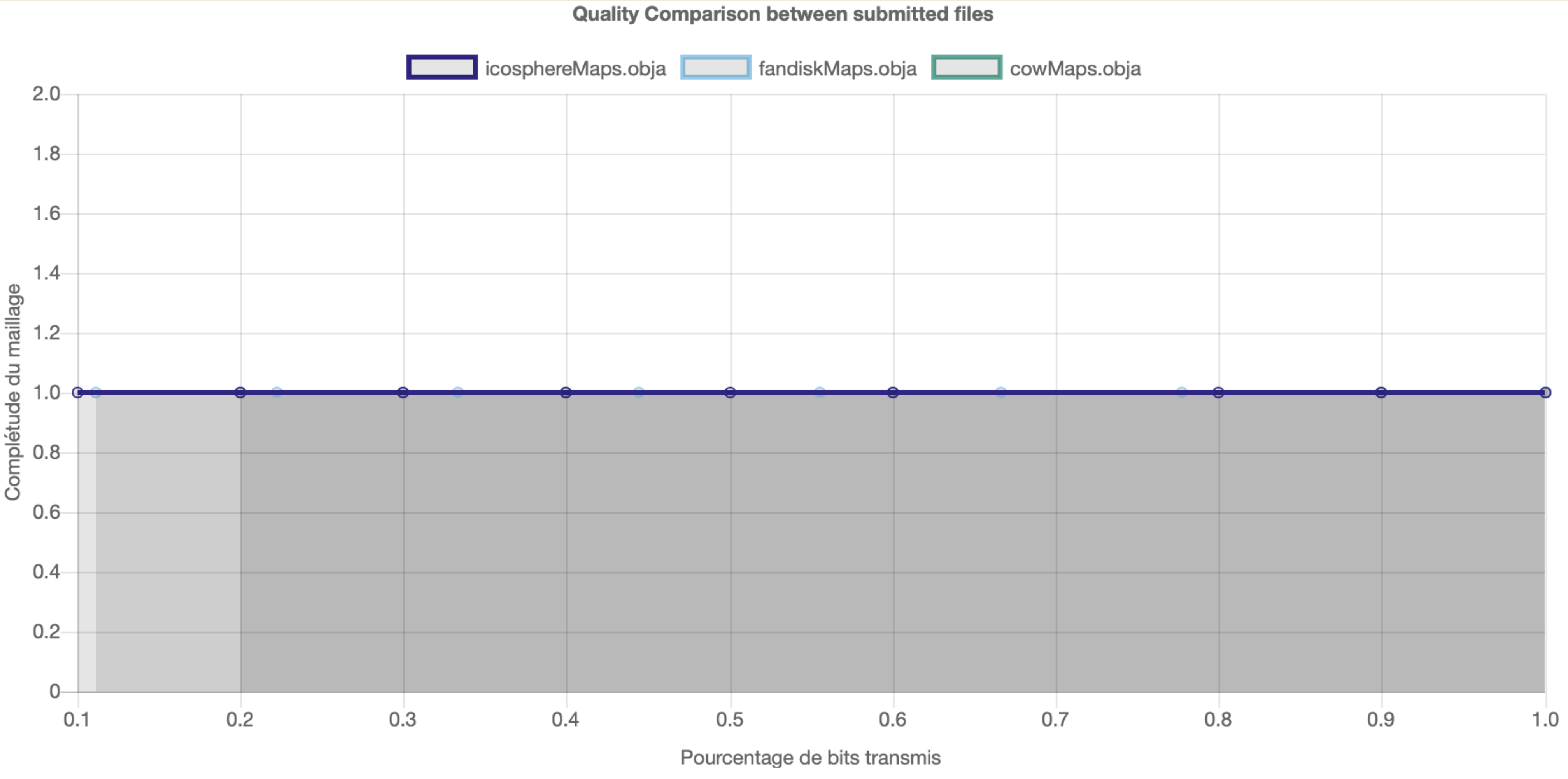
Table showing the effect of compression on the number of vertices

Dataset	Input size (vertex)	Output size (vertex)
Bunny	2503	84
Cow	3 906	1 126
Hand	10 000	473
FanDisc	9 864	4
Icosphere	642	3
Hippo	32 144	6 393

Videos compression + decompression

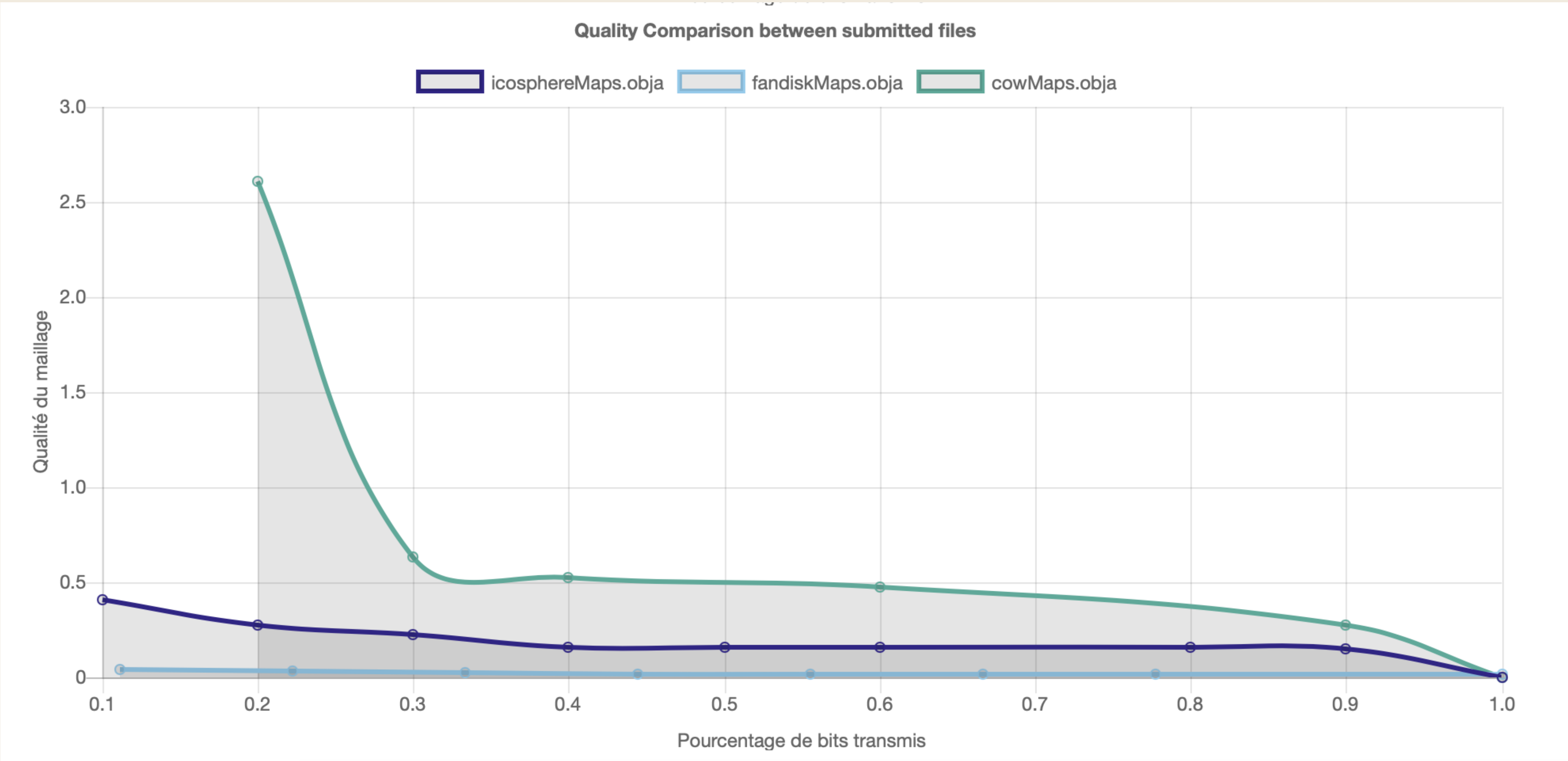
IV - Results

Quantification - Completude



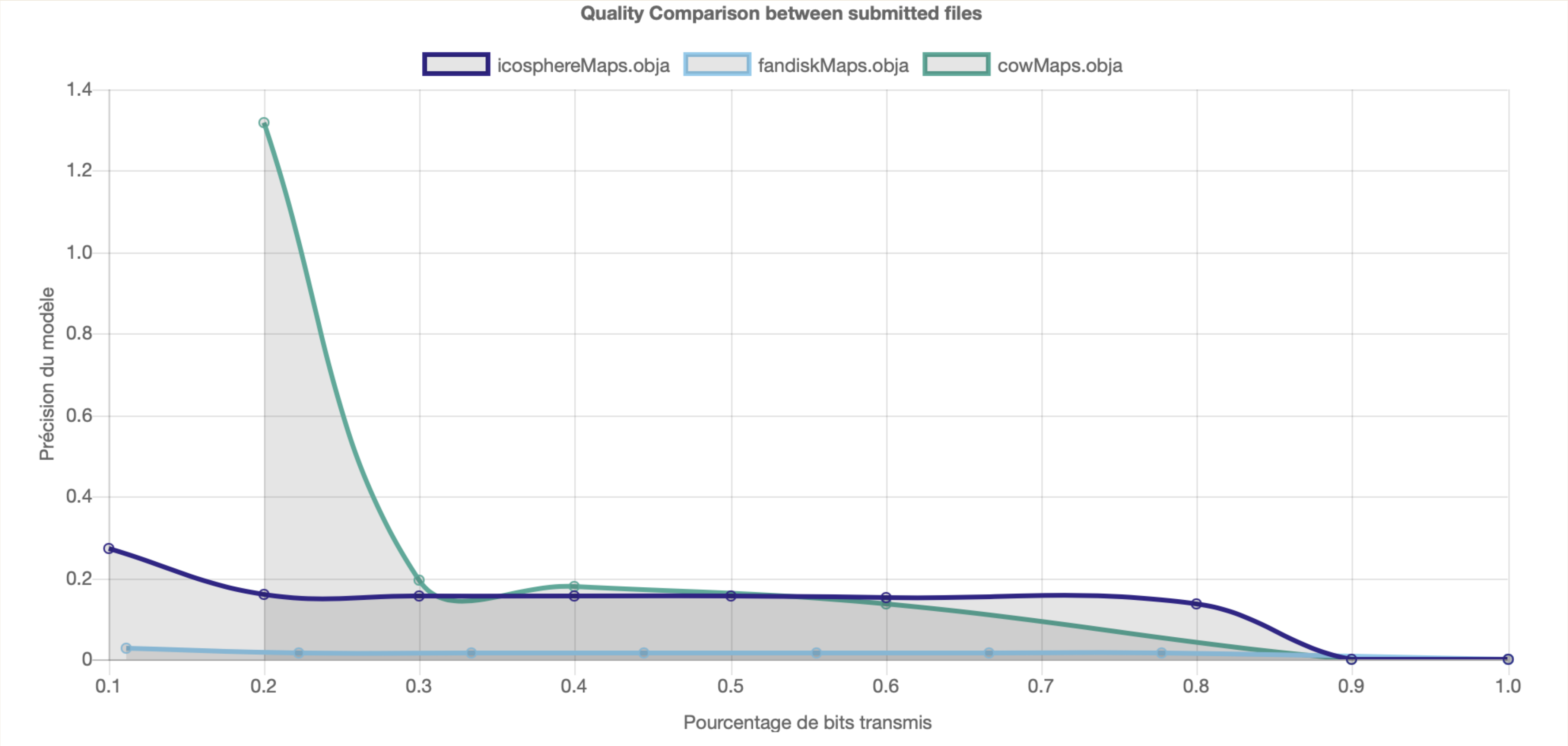
IV - Results

Quantification – Hausdorff distance



IV - Results

Quantification – Middlebury



V - Issues

The issues we encountered and solved:

- Delaunay to impose constraints (segment indexing issue)
- Indexing problem (replace 'for' loops with mappings)
- Curvature calculation (switching methods based on another scientific article)

The remaining issues to address:

- Retriangulation issue for non-compact problems
- Edge handling (manual retriangulation)

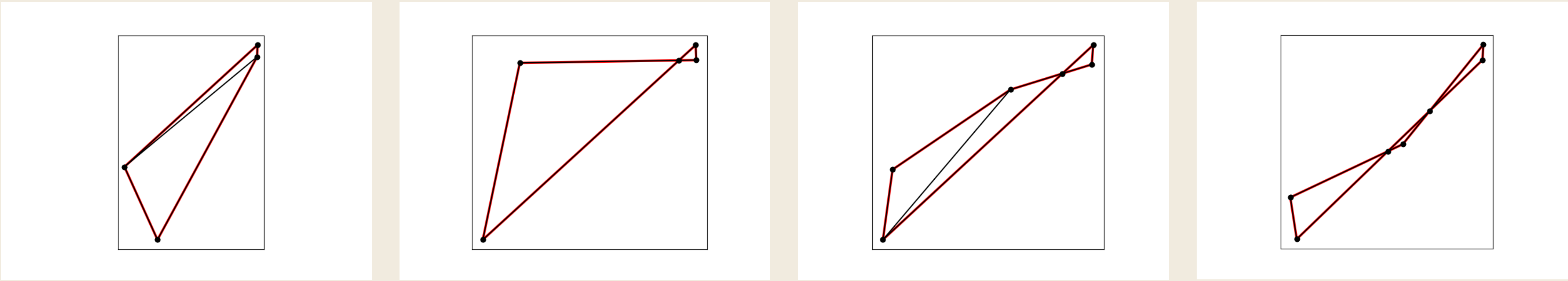
V - Issues

Edge handling

- To find points on the borders:
    - an edge of the border is an edge that belongs to only one triangle.
    - a vertex on the border is a vertex that belongs to an edge on the border.

- Conformal map: for the borders, instead of mapping the neighbors of vertices to be removed by a disc, they are mapped by a half-disc.

- Issue during Delaunay retriangulation on the borders: handled manually.



Thank you for your attention