Centroidal Voronoi Tessellations on Meshes

Leonardo K. Sacht & Thiago S. Pereira

Visgraf - IMPA

November 18, 2008

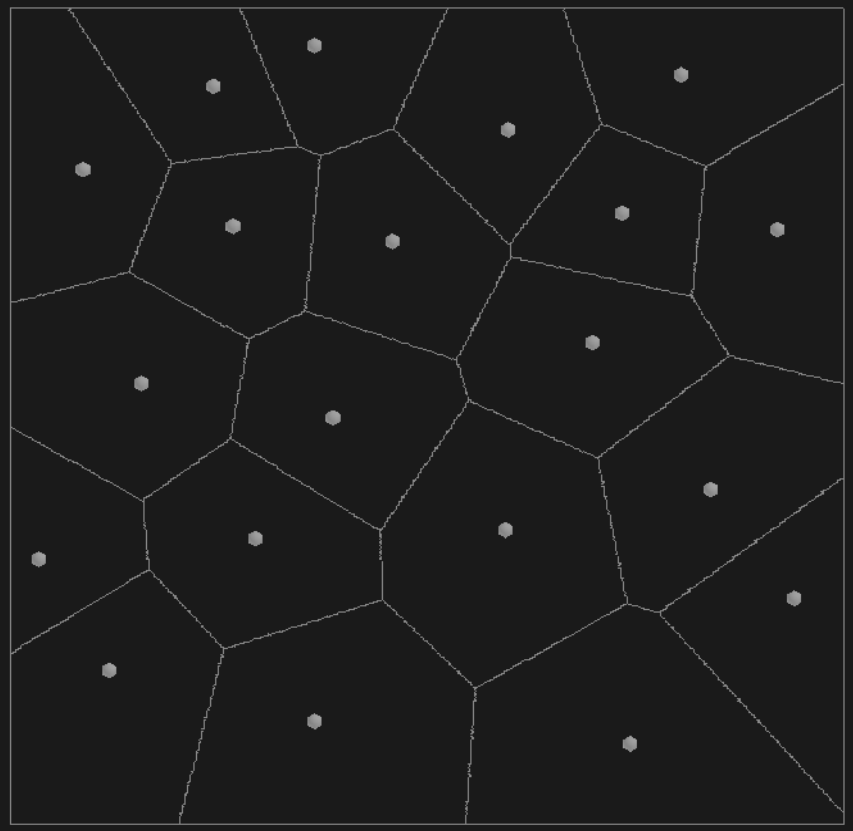
Centroidal Voronoi Tessellations

Definition: Given $\Omega \subseteq \mathbb{R}^n$, $\{z_i\}_{i=1}^k \subseteq \Omega$, the *Voronoi region* of each z_i is defined by

$$\hat{V}_i = \{x \in \Omega : \| x - z_i \| < \| x - z_j \| \text{ for } j = 1, \ldots, k, j \neq i\}.$$

Centroidal Voronoi Tessellations

A Voronoi Diagram of $[0,1] \times [0,1] \subseteq \mathbb{R}^2$



Centroidal Voronoi Tessellations

Definition: Let $V \subseteq \mathbb{R}^n$ and ρ a density function on V. The *mass centroid* of V is $z^* \in V$ such that

$$\int_V \rho(y) \parallel z^* - y \parallel^2 dy = \inf_{z \in V^*} \int_V \rho(y) \parallel z - y \parallel^2 dy,$$

where V^* is the closure of V.

The minimizer of this functional is

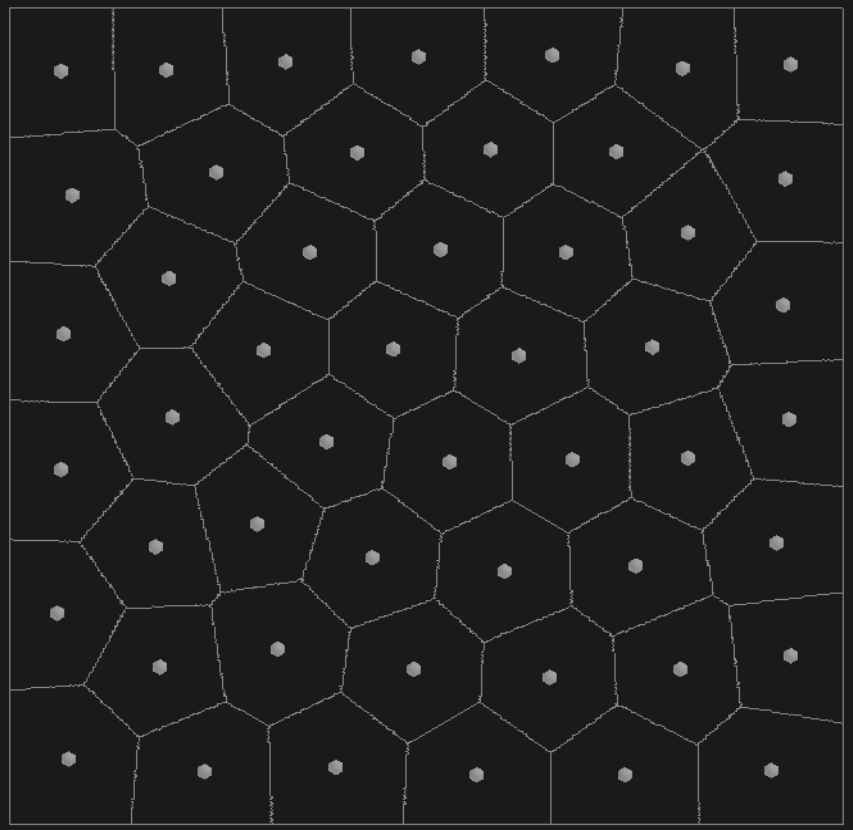
$$z^* = \frac{\int_V y\rho(y)dy}{\int_V \rho(y)dy}.$$

Centroidal Voronoi Tessellations

Definition: $\{\hat{V}_i\}_{i=1}^{n}$ is a *Centroidal Voronoi Tessellation* of $\Omega \subseteq \mathbb{R}^n$ if the generators of $\hat{V}_i$ are their mass centroids.

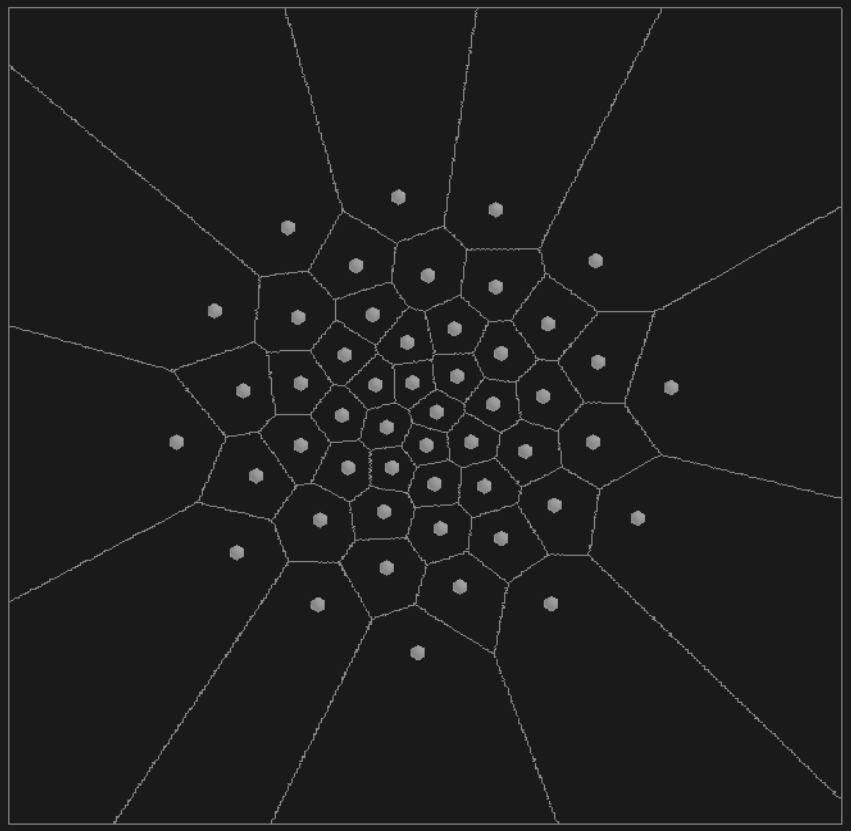
Centroidal Voronoi Tessellations

A Centroidal Voronoi Tessellation of with $\rho \equiv 1$.



Centroidal Voronoi Tessellations

$$\rho(x, y) = e^{-40x^2-40y^2}$$



Applications

- Data Compression in Image Processing;

Applications

- Data Compression in Image Processing;
- Quadrature Rules;

Applications

- Data Compression in Image Processing;
- Quadrature Rules;
- Optimal Representation, Quantization, and Clustering;

Applications

- Data Compression in Image Processing;
- Quadrature Rules;
- Optimal Representation, Quantization, and Clustering;
- Finite Difference Schemes Having Optimal Truncation Errors;

Applications

- Data Compression in Image Processing;
- Quadrature Rules;
- Optimal Representation, Quantization, and Clustering;
- Finite Difference Schemes Having Optimal Truncation Errors;
- Optimal Placement of Resources;

Applications

- Data Compression in Image Processing;
- Quadrature Rules;
- Optimal Representation, Quantization, and Clustering;
- Finite Difference Schemes Having Optimal Truncation Errors;
- Optimal Placement of Resources;
- Cell Division;

Applications

- Data Compression in Image Processing;
- Quadrature Rules;
- Optimal Representation, Quantization, and Clustering;
- Finite Difference Schemes Having Optimal Truncation Errors;
- Optimal Placement of Resources;
- Cell Division;
- Territorial Behavior of Animals;

Applications

- Data Compression in Image Processing;
- Quadrature Rules;
- Optimal Representation, Quantization, and Clustering;
- Finite Difference Schemes Having Optimal Truncation Errors;
- Optimal Placement of Resources;
- Cell Division;
- Territorial Behavior of Animals;

- **Reference:** [2].

Applications

- Remeshing;

Applications

- Remeshing;
- Our applications...

Framework of the Problem

- Given
  - a region $\Omega \subseteq \mathbb{R}^n$,
  - a positive integer k,
  - a density function ρ, defined on Ω,

Framework of the Problem

- Given
  - a region $\Omega \subseteq \mathbb{R}^n$,
  - a positive integer k,
  - a density function ρ, defined on Ω,
- find
  - k points $z_i \in \Omega$,
  - k regions V_i that tesselate Ω,

Framework of the Problem

- Given
  - a region $\Omega \subseteq \mathbb{R}^n$,
  - a positive integer k,
  - a density function ρ, defined on Ω,
- find
  - k points $z_i \in \Omega$,
  - k regions V_i that tesselate Ω,
- such that *simultaneously* for each i
  - V_i is the Voronoi region for z_i,
  - z_i is the mass centroid of V_i.

LLoyd's Algorithm

Lloyd in Back to the Future III

Lloyd's Algorithm

It's the most standard and intuitive algorithm to compute CVD's. It's described by the following steps:

- (1) Compute randomly k points $z_i \in \Omega$;

Lloyd's Algorithm

It's the most standard and intuitive algorithm to compute CVD's. It's described by the following steps:

- (1) Compute randomly k points $z_i \in \Omega$;
- (2) Compute the Voronoi regions $\hat{V}_i$ determined by the points z_i;

Lloyd's Algorithm

It's the most standard and intuitive algorithm to compute CVD's. It's described by the following steps:

- (1) Compute randomly k points $z_i \in \Omega$;
- (2) Compute the Voronoi regions $\hat{V}_i$ determined by the points z_i;
- (3) Calculate the centroids z_i^* of $\hat{V}_i$ and set $z_i = z_i^*$;

Lloyd's Algorithm

It's the most standard and intuitive algorithm to compute CVD's. It's described by the following steps:

- (1) Compute randomly k points $z_i \in \Omega$;
- (2) Compute the Voronoi regions $\hat{V}_i$ determined by the points z_i;
- (3) Calculate the centroids z_i^* of $\hat{V}_i$ and set $z_i = z_i^*$;
- (4) If some stop criteria is met for this new set of points z_i, stop. Otherwise, return to step 2.

LLOYD'S ALGORITHM

Show Lloyd's Algorithm working with planar examples.

Energy Functional

Proposition: Let $\{z_i\}_{i=1}^k \subseteq S \subseteq \mathbb{R}^2$ and $\{V_i\}_{i=1}^k$ a tesselation of S. Define the **energy functional** or the **distortion value** for $\{(z_i, V_i)\}_{i=1}^k$ by

$$\mathcal{F}(\{(z_i, V_i)\}_{i=1}^k) = \sum_{i=1}^{k} \int_{x \in V_i} \rho(x) \parallel x - z_i \parallel^2 dx.$$

A necessary condition for $\mathcal{F}$ to be minimized is that $\{(z_i, V_i)\}_{i=1}^k$ is a centroidal Voronoi tesselation of S.

And for surfaces?

And for surfaces?

If $S \subseteq \mathbb{R}^3$, the euclidian centroid usually does not belong to S.

Definition: Let $V_i \subseteq S$ be a region of a surface. The **contrained mass centroid** of V_i on S, z_i^c, is a solution of the following problem:

$$\min_{z \in S} F_i(z), \qquad \text{where} \qquad F_i(z) = \int_{V_i} \rho(x) \parallel x - z_i \parallel^2 dx.$$

And for surfaces?

In [3], it's proved that the constrained centroid z_i^c is the projection of z_i^* (the euclidean centroid) onto S along the normal direction at z_i^c.

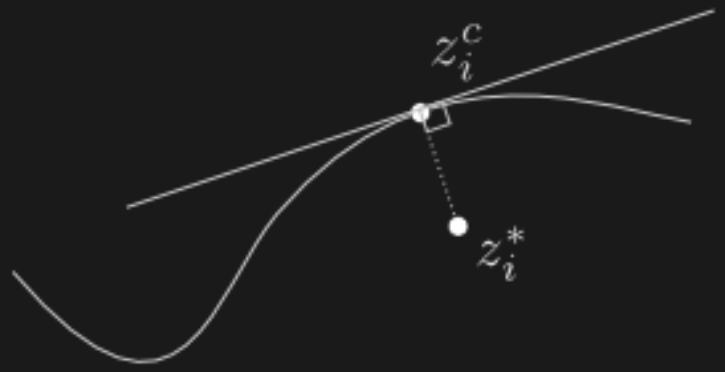


Figure: Projection onto a surface

And for surfaces?

Definition: $\{V_i\}_{i=1}^k$ is a **Constrained Centroidal Voronoi Tessellation** of S if $\{V_i\}_{i=1}^k$ are Voronoi regions on S defined by their constrained mass centroids.

Energy Functional on Surfaces

Proposition: Let $\{z_i\}_{i=1}^k \subseteq S \subseteq \mathbb{R}^3$ and $\{V_i\}_{i=1}^k$ a tesselation of S. Define the **energy functional** or the **distortion value** for $\{(z_i, V_i)\}_{i=1}^k$ by

$$\mathcal{F}(\{(z_i, V_i)\}_{i=1}^k) = \sum_{i=1}^{k} \int_{x \in V_i} \rho(x) \parallel x - z_i \parallel^2 dx.$$

A necessary condition for $\mathcal{F}$ to be minimized is that $\{(z_i, V_i)\}_{i=1}^k$ is a **constrained** centroidal Voronoi tesselation of S.

Lloyd's Algorithm (again)

- (1) Compute randomly k points $z_i \in \Omega$;

Lloyd's Algorithm (again)

- (1) Compute randomly k points $z_i \in \Omega$;
- (2) Compute the Voronoi regions $\hat{V}_i$ determined by the points z_i;

Lloyd's Algorithm (again)

- (1) Compute randomly k points $z_i \in \Omega$;
- (2) Compute the Voronoi regions $\hat{V}_i$ determined by the points z_i;
- (3) Calculate the **constrained** centroids z_i^* of $\hat{V}_i$ and set $z_i = z_i^*$;

Lloyd's Algorithm (again)

- (1) Compute randomly k points $z_i \in \Omega$;
- (2) Compute the Voronoi regions $\hat{V}_i$ determined by the points z_i;
- (3) Calculate the **constrained** centroids z_i^* of $\hat{V}_i$ and set $z_i = z_i^*$;
- (4) If some stop criteria is met for this new set of points z_i, stop. Otherwise, return to step 2.

Data Structure

The surfaces will be discretized by half-edge meshes.

Step 1 - Initializing Centroids

To get a good first approximation for the centroids, we have to pick them according to the density function ρ:

- (1) Get randomly a face f, calculate its center f_c and a random number f_r;

Step 1 - Initializing Centroids

To get a good first approximation for the centroids, we have to pick them according to the density function ρ:

- (1) Get randomly a face f, calculate its center f_c and a random number f_r;
- (2) If $\rho(f_c) * area(f) \leqslant f_r$ then reject f_c. Otherwise, set f_c to be a centroid for the first iteration (the comparison is made after a normalization of the density);

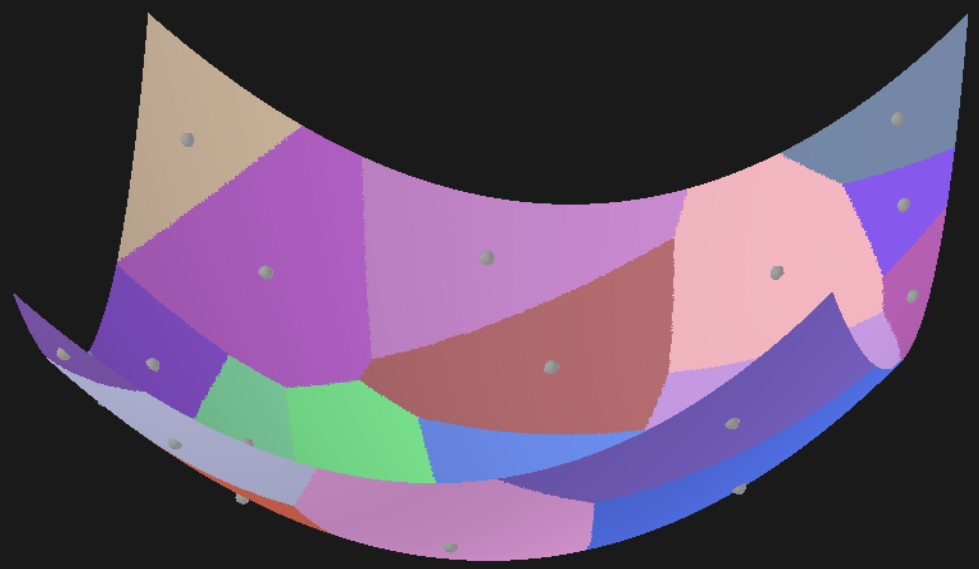
Step 1 - Initializing Centroids

To get a good first approximation for the centroids, we have to pick them according to the density function ρ:

- (1) Get randomly a face f, calculate its center f_c and a random number f_r;
- (2) If $\rho(f_c) * area(f) \leqslant f_r$ then reject f_c. Otherwise, set f_c to be a centroid for the first iteration (the comparison is made after a normalization of the density);
- **Time:** Random variable.

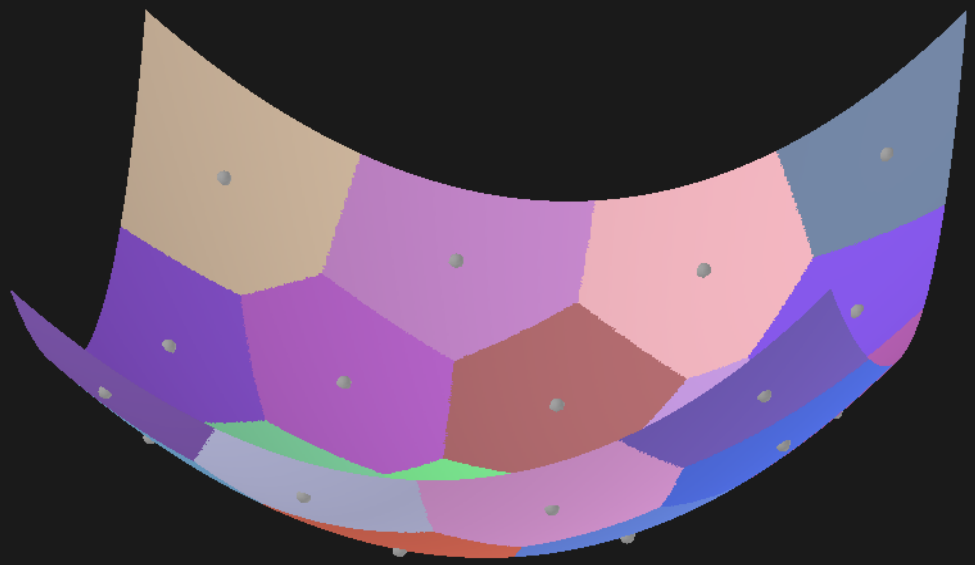
Step 1 - Initializing Centroids

Initialization of the centroids on a paraboloid, with $\rho \equiv 1$



Step 1 - Initializing Centroids

Final result



Step 2 - Computing Voronoi Regions

- (1) For each face f, calculate the distance of its center to all the z_i's. Put f on the *patch* of the z_i with minimum distance;

Step 2 - Computing Voronoi Regions

- (1) For each face f, calculate the distance of its center to all the z_i's. Put f on the *patch* of the z_i with minimum distance;
- This step will tesselate the set of faces in k regions corresponding to the k centroids;

Step 2 - Computing Voronoi Regions

- (1) For each face f, calculate the distance of its center to all the z_i's. Put f on the *patch* of the z_i with minimum distance;
- This step will tesselate the set of faces in k regions corresponding to the k centroids;
- **Time:** $O(kn)$.

Step 3 - Calculating Centroids

Now, for all patches V_i (set of faces) we wish to calculate their centroids:

- (1) For each face $f \in V_i$ calculate

$$f_c * area(f) * \rho(f) \approx \int_f y\rho(y)dy;$$

Step 3 - Calculating Centroids

Now, for all patches V_i (set of faces) we wish to calculate their centroids:

- (1) For each face $f \in V_i$ calculate

$$f_c * area(f) * \rho(f) \approx \int_f y\rho(y)dy;$$

- (2) Sum over all faces:

$$\sum_{f \in V_i} f_c * area(f) * \rho(f) \approx \int_{V_i} y\rho(y)dy;$$

Step 3 - Calculating Centroids

Now, for all patches V_i (set of faces) we wish to calculate their centroids:

- (1) For each face $f \in V_i$ calculate

$$f_c * area(f) * \rho(f) \approx \int_f y\rho(y)dy;$$

- (2) Sum over all faces:

$$\sum_{f \in V_i} f_c * area(f) * \rho(f) \approx \int_{V_i} y\rho(y)dy;$$

- (3) Do the same to calculate

$$\sum_{f \in V_i} area(f) * \rho(f) \approx \int_{V_i} \rho(y)dy;$$

Step 3 - Calculating Centroids

- (4) Compute

$$\frac{\sum_{f \in V_i} f_c * area(f) * \rho(f)}{\sum_{f \in V_i} area(f) * \rho(f_c)}$$

(approximate centroid of V_i).

Step 3 - Calculating Centroids

- (4) Compute

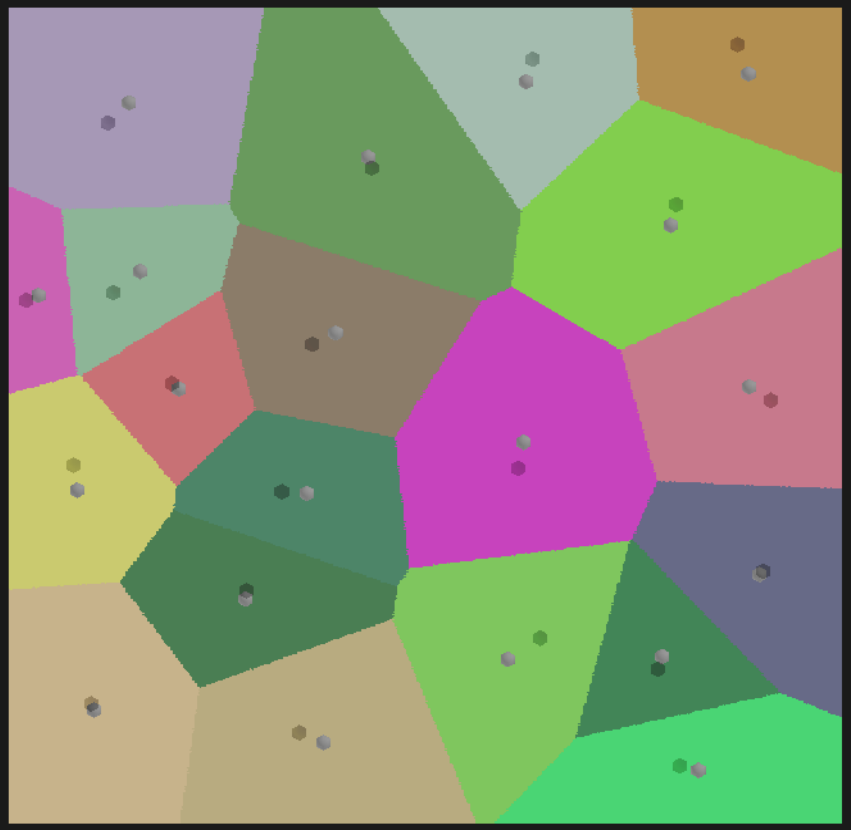
$$\frac{\sum_{f \in V_i} f_c * area(f) * \rho(f)}{\sum_{f \in V_i} area(f) * \rho(f_c)}$$

(approximate centroid of V_i).

- **Cost:** $O(n)$.

Step 3 - Calculating Centroids

Update of the centroids



Step 3.5 - Projecting Centroids

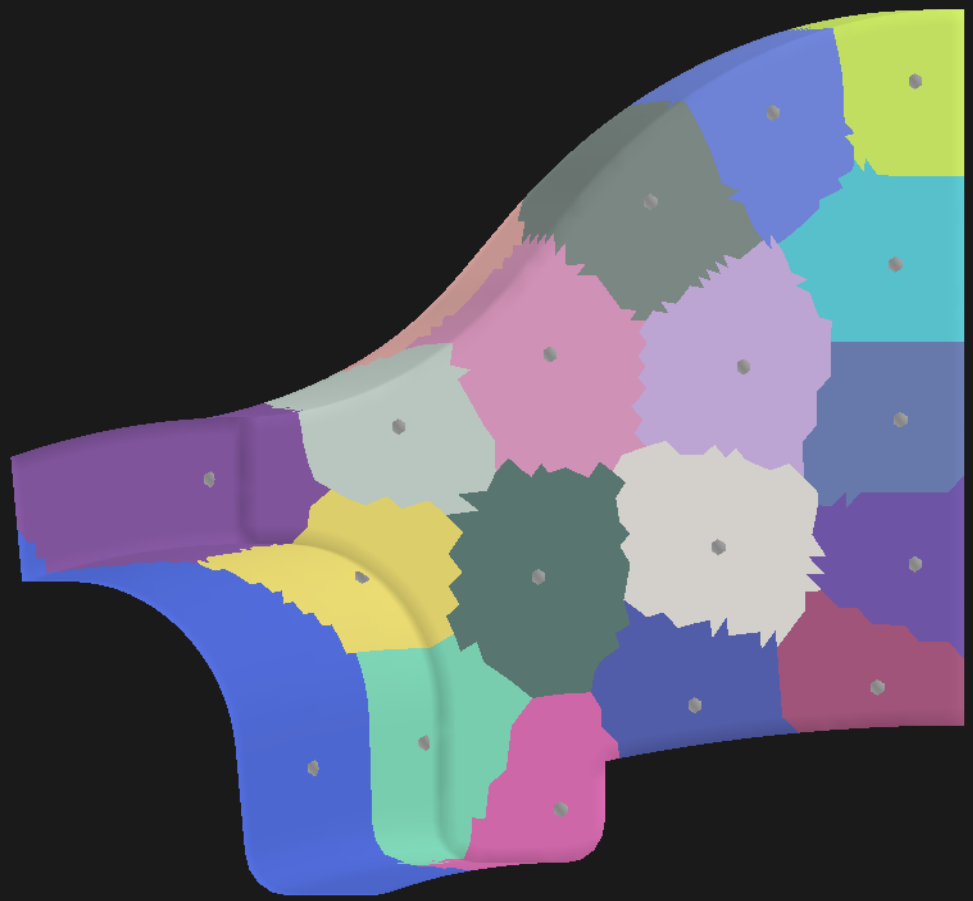
- (1) For each centroid calculated on step 3, look for the face that is closest to it and set its center to be the projected centroid.

Step 3.5 - Projecting Centroids

- (1) For each centroid calculated on step 3, look for the face that is closest to it and set its center to be the projected centroid.
- **Cost:** $O(n)$.

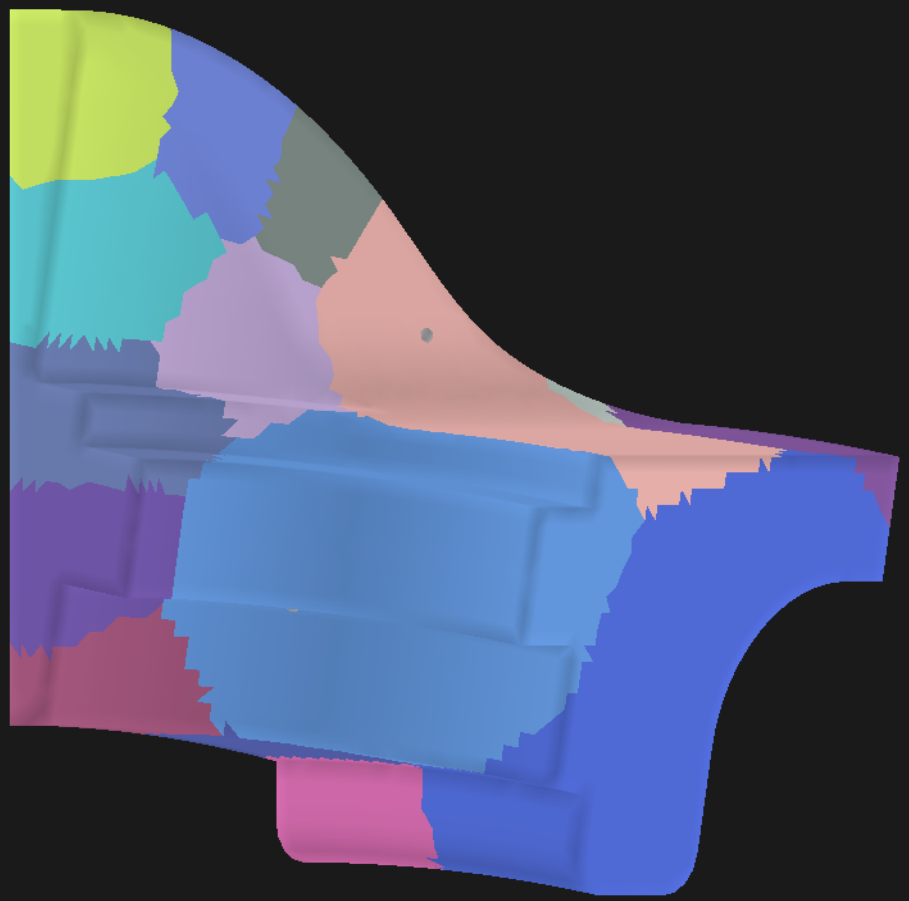
Results

Final result, with $\rho(x, y, z) = e^{-40x}$



Results

Final result, with $\rho(x, y, z) = e^{-40x}$ (other side).



Results

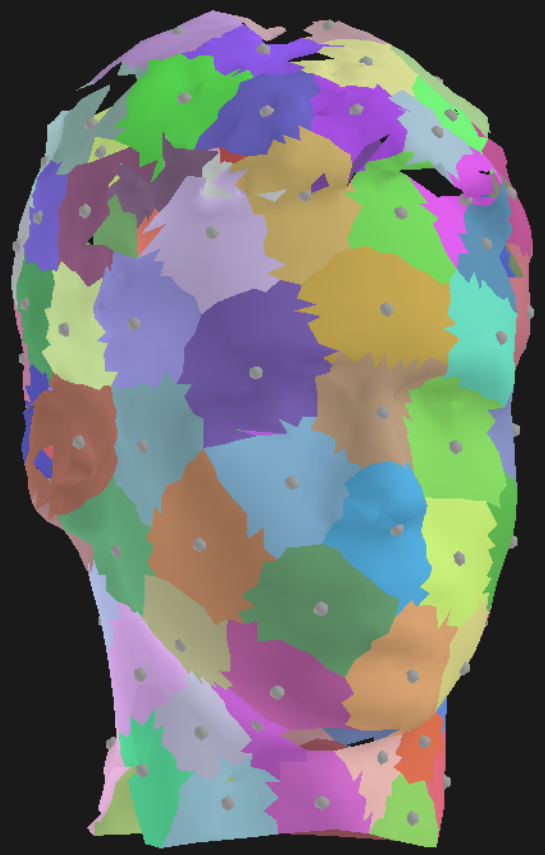
$\rho(x) =$ discrete mean curvature [1]

Results

$\rho(x) =$ discrete mean curvature [1]

Results

Thiago's Mesh with $\rho \equiv 1$



Results

Leo's Mesh with $\rho \equiv 1$

Normal Clustering

In [1], the authors proposed a way to generalize the concept of centroid, taking into account the normals of the surface at the points. They created the *proxies*:

- Given a partition $R = \{R_i\}_{i=1}^{n}$ of a triangulated surface S, the *proxy* of each region is a pair $P_i = (X_i, N_i)$, where $X_i \in R_i$ and N_i is a normal vector that *best* represents the normals on the region.

Normal Clustering

In [1], the authors proposed a way to generalize the concept of centroid, taking into account the normals of the surface at the points. They created the *proxies*:

- Given a partition $R = \{R_i\}_{i=1}^{n}$ of a triangulated surface S, the *proxy* of each region is a pair $P_i = (X_i, N_i)$, where $X_i \in R_i$ and N_i is a normal vector that *best* represents the normals on the region.
- Given a face f, we represent it by its center (f_c) and the normal (f_N) of the piecewise linear suface on f.

VSA Energy

- Given a region $V_i \subseteq S$ and it's corresponding proxy P_i, the $\mathcal{L}^{2,1}$ energy functional of (V_i, P_i) is defined by

$$\mathcal{L}^{2,1}(V_i, P_i) = \int_{x \in V_i} \rho(x) \parallel n(x) - n_i \parallel^2 dx,$$

VSA Energy

- Given a region $V_i \subseteq S$ and it's corresponding proxy P_i, the $\mathcal{L}^{2,1}$ energy functional of (V_i, P_i) is defined by

$$\mathcal{L}^{2,1}(V_i, P_i) = \int_{x \in V_i} \rho(x) \parallel n(x) - n_i \parallel^2 dx,$$

- **Definition:** Given a number k of proxies and an input surface S, we call **optimal shape proxies** a set P of proxies P_i associated to regions V_i of a partition R of S that minimizes the total distortion:

$$E(R, P) = \sum_{i=1}^{k} \mathcal{L}^{2,1}(V_i, P_i)$$

"Lloyd's" Algorithm for Normal Clustering

- (1) Compute randomly k faces $f_i \in S$ and set their centers and their normals to be the proxies $P_i = (X_i, N_i)$;

"Lloyd's" Algorithm for Normal Clustering

- (1) Compute randomly k faces $f_i \in S$ and set their centers and their normals to be the proxies $P_i = (X_i, N_i)$;
- (2) Compute the "Voronoi regions" V_i determined by the proxies P_i in the following way: get the normal of each face f and set f to be in the region determined by the proxy that has the closest normal.

"Lloyd's" Algorithm for Normal Clustering

- (3) For each new V_i, set the constrained centroid of V_i to be X_i and the average normal

$$N_i = \frac{\sum_{f \in V_i} \rho(f) * f_N * area(f)}{\sum_{f \in V_i} \rho(f) * area(f)} \approx \frac{\int_{V_i} \rho(x) n(x) dx}{\int_{V_i} \rho(x) dx}.$$

Note: This new proxies minimize, for each V_i, the $\mathcal{L}^{2,1}(V_i, P_i)$ energy ([1]).

"Lloyd's" Algorithm for Normal Clustering

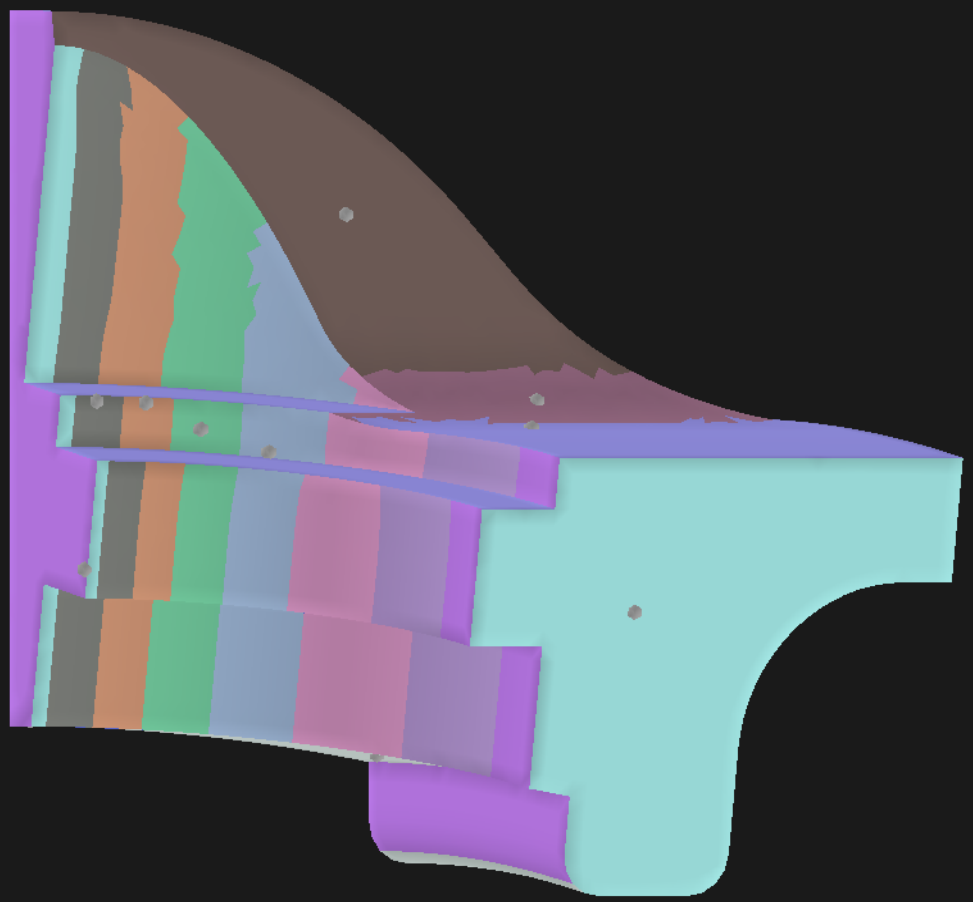
- (3) For each new V_i, set the constrained centroid of V_i to be X_i and the average normal

$$N_i = \frac{\sum_{f \in V_i} \rho(f) * f_N * area(f)}{\sum_{f \in V_i} \rho(f) * area(f)} \approx \frac{\int_{V_i} \rho(x) n(x) dx}{\int_{V_i} \rho(x) dx}.$$

  Note: This new proxies minimize, for each V_i, the $\mathcal{L}^{2,1}(V_i, P_i)$ energy ([1]).
- (4) If some stop criteria is met for this new set of points z_i, stop. Otherwise, return to step 2.

Results

Normal clustering with $\rho \equiv 1$



Mixing both approaches

We propose a new energy functional that takes into acount both informations of positions and normals given by the proxies.

$$\mathcal{E}(\{(P_i, V_i)\}_{i=1}^k) = \sum_{i=1}^{k} \alpha \mathcal{F}((X_i, V_i) + (1-\alpha)\mathcal{L}^{2,1}(V_i, P_i) =$$

$$\sum_{i=1}^{k} \alpha \int_{x \in V_i} \rho(x) \parallel x - z_i \parallel^2 dx + (1-\alpha) \int_{x \in V_i} \rho(x) \parallel n(x) - n_i \parallel^2 dx.$$

- Here we have to normalize the first energy to have the same magnitude of the second.

Mixing both approaches

We propose a new energy functional that takes into acount both informations of positions and normals given by the proxies.

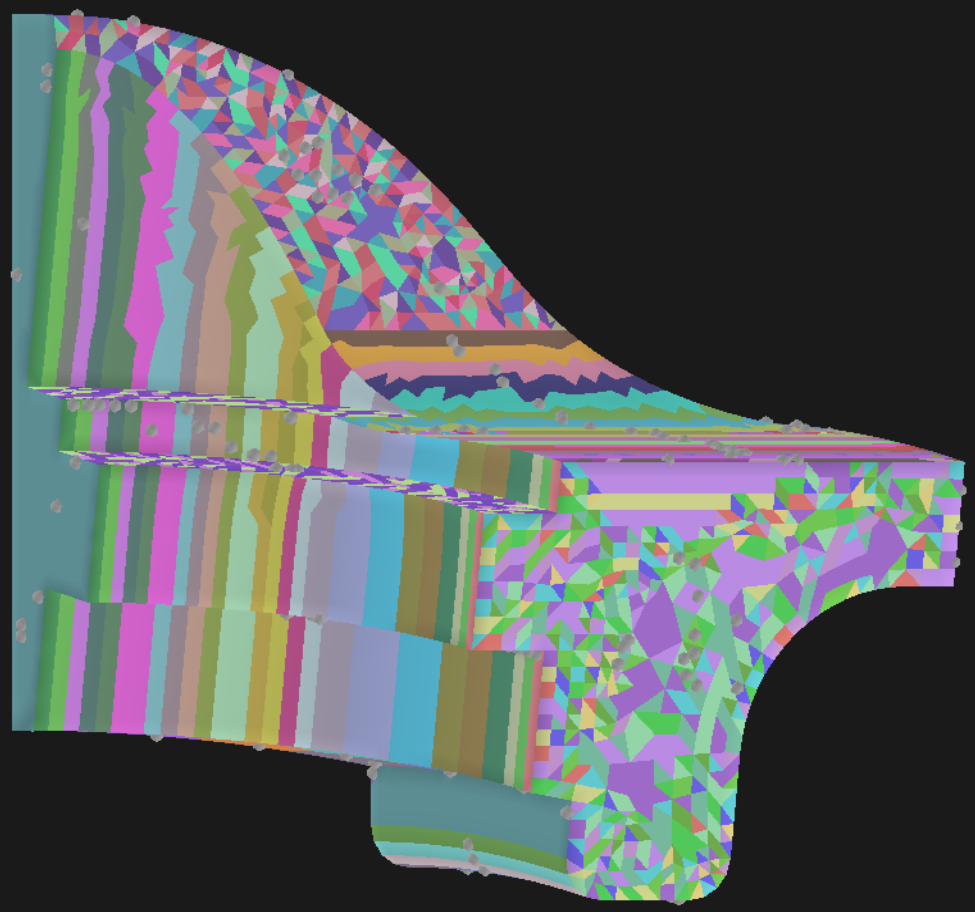
$$\mathcal{E}(\{(P_i, V_i)\}_{i=1}^k) = \sum_{i=1}^{k} \alpha \mathcal{F}((X_i, V_i) + (1-\alpha)\mathcal{L}^{2,1}(V_i, P_i) =$$

$$\sum_{i=1}^{k} \alpha \int_{x \in V_i} \rho(x) \parallel x - z_i \parallel^2 dx + (1-\alpha) \int_{x \in V_i} \rho(x) \parallel n(x) - n_i \parallel^2 dx.$$

- Here we have to normalize the first energy to have the same magnitude of the second.
- The updating of the regions and the proxies is done in the same way already described.

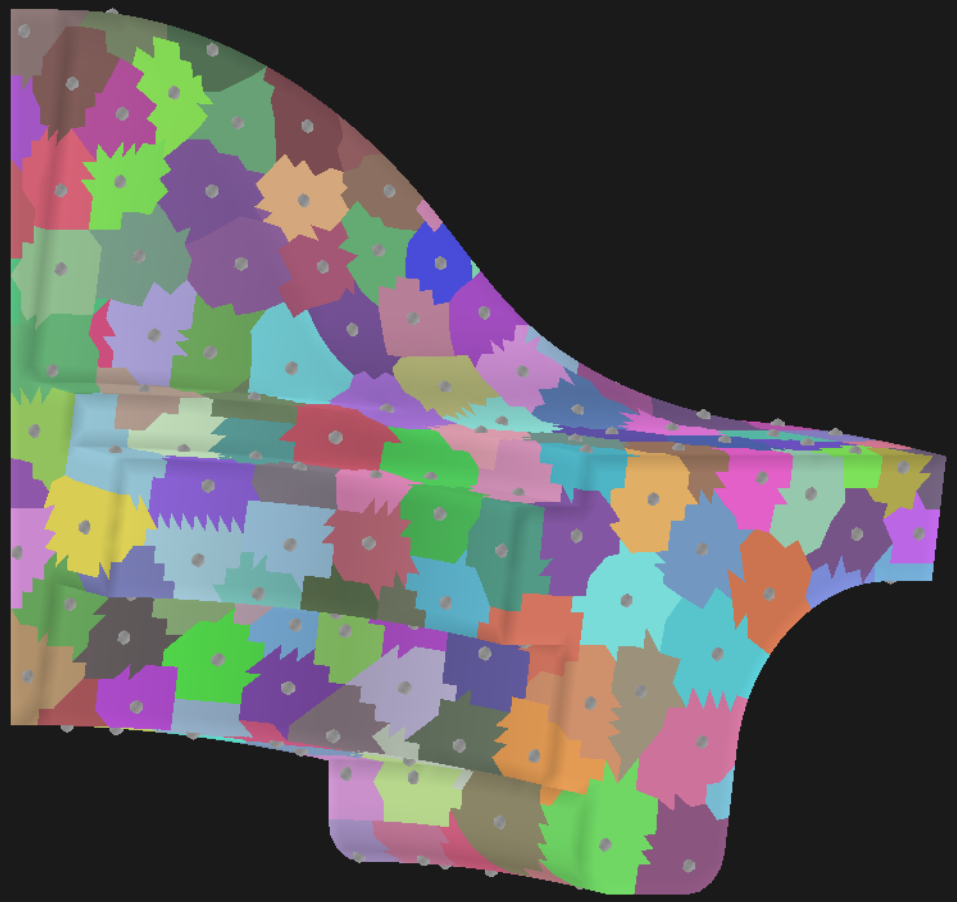
Results

Mixed Approach with $\alpha = 0$ and $\rho \equiv 1$



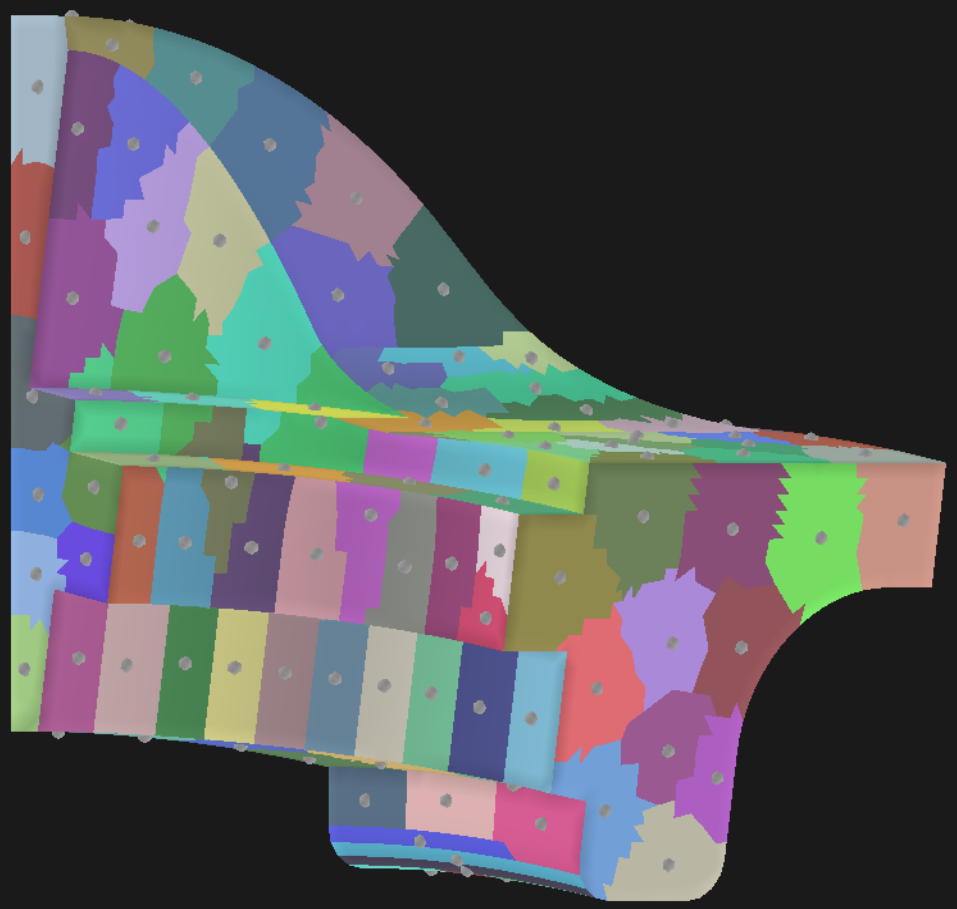
Results

Mixed Approach with $\alpha = 1$ and $\rho \equiv 1$

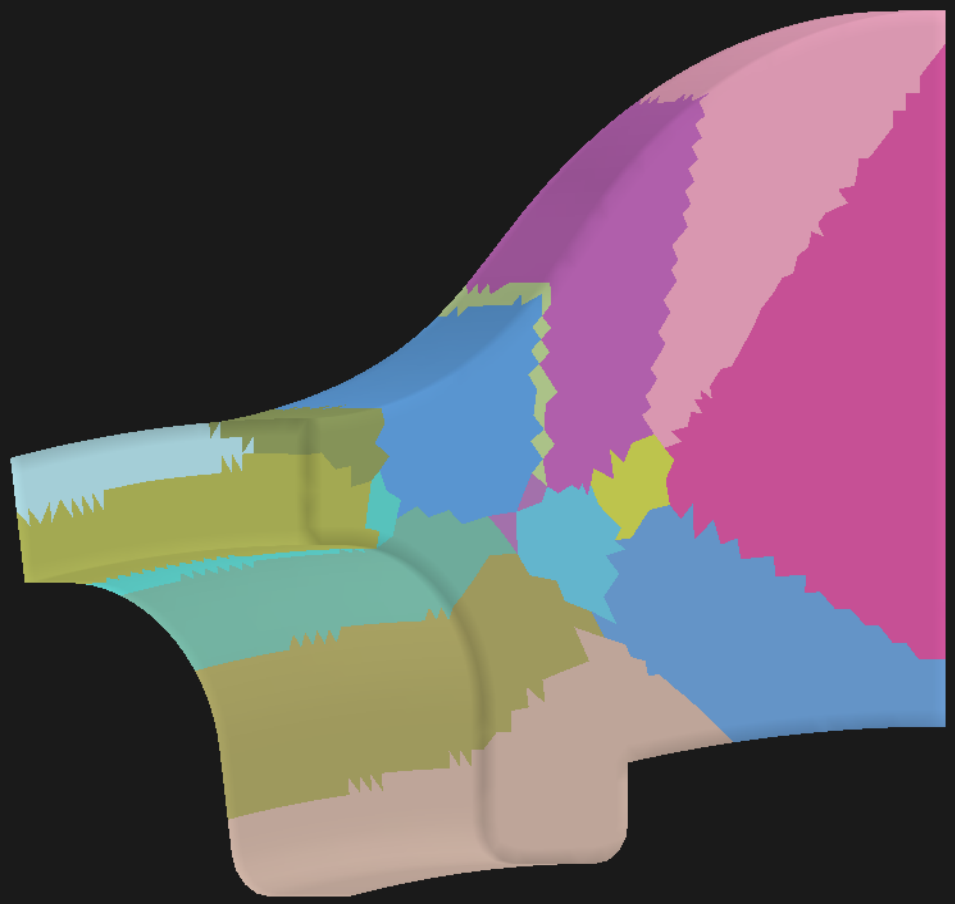


Results

Mixed Approach with $\alpha = 0.5$ and $\rho \equiv 1$



Results



Flooding Approach

- Due to the problems observed in the last examples, we desire a method that keep the Voronoi regions connected and takes into account the geodesic distance.

Flooding Approach

- Due to the problems observed in the last examples, we desire a method that keep the Voronoi regions connected and takes into account the geodesic distance.
- Given the faces that discretize the surface, we'll work with the *dual graph*, i.e., the graph with nodes being the center of the faces and edges describing the neighborhood of each face.

Flooding - Update Regions

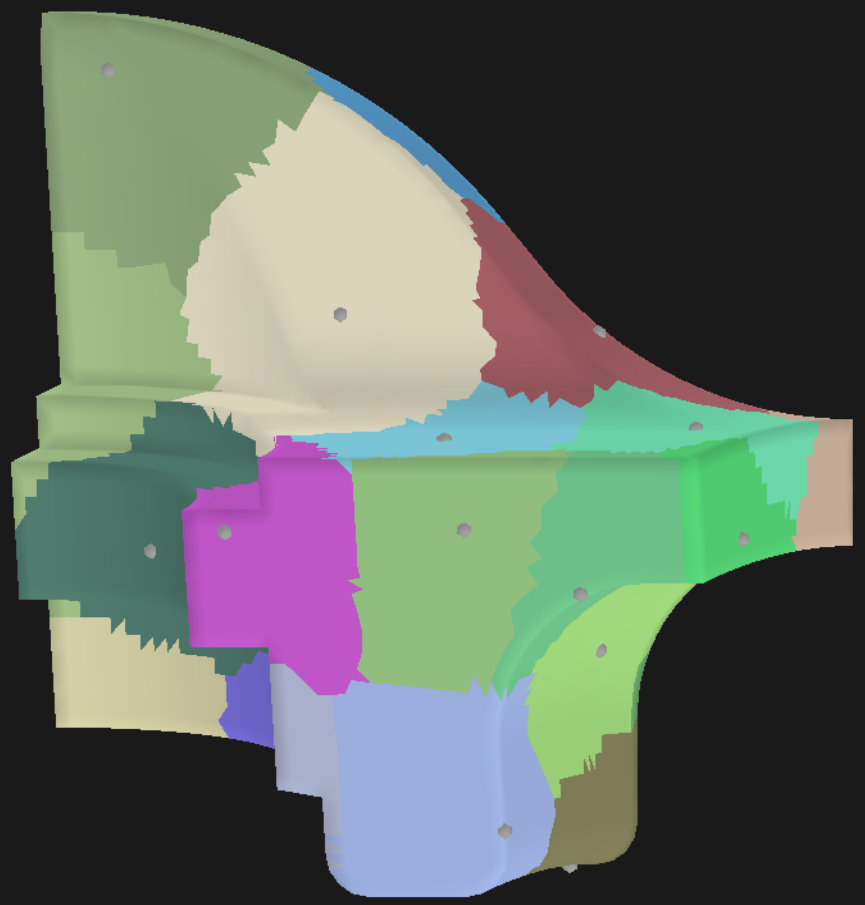
- For each centroid, grow a flooding regions with Dijkstra's Algorithm. All faces will be associated to the closest centroid, respecting the "discretized surface metric". We'll call the set of faces associated to a centroid a *patch*.

Flooding - Update Regions

- For each centroid, grow a flooding regions with Dijkstra's Algorithm. All faces will be associated to the closest centroid, respecting the "discretized surface metric". We'll call the set of faces associated to a centroid a *patch*.
- **Cost:** $O(V \log V) = O(n \log n)$.

Results

Calculating regions with geodesic metric, $\rho \equiv 1$.



Flooding - Update Patch Centroids

- For each face f in patch P, calculate

$$r(f) = \max_{\tilde{f} \in P} d(f, \tilde{f}).$$

Flooding - Update Patch Centroids

- For each face f in patch P, calculate

$$r(f) = \max_{\tilde{f} \in P} d(f, \tilde{f}).$$

- Set the center c of the patch P to be

$$c = \arg\left(\min_{f \in P} r(f)\right).$$

Flooding - Update Patch Centroids

- For each face f in patch P, calculate

$$r(f) = \max_{\tilde{f} \in P} d(f, \tilde{f}).$$

- Set the center c of the patch P to be

$$c = \arg\left(\min_{f \in P} r(f)\right).$$

- **Cost:** $O(n^2 \log n)$.

Results

Final result with geodesic metric, $\rho \equiv 1$.

Future Work

- Use better methods to calculate geodesics (Fast Marching, [2]).

Future Work

- Use better methods to calculate geodesics (Fast Marching, [2]).
- Use a Gradient Descent approach to perform geodesic calculations ([2]).

Future Work

- Use better methods to calculate geodesics (Fast Marching, [2]).
- Use a Gradient Descent approach to perform geodesic calculations ([2]).
- Explore more the geodesic approach, including densities.

Future Work

- Alternatives to discretization:
  1) continous centroids;
  2) continous regions.

Future Work

- Alternatives to discretization:
  1) continous centroids;
  2) continous regions.
- Post process the results to avoid zig zag border.

Future Work

- Alternatives to discretization:
  1) continous centroids;
  2) continous regions.
- Post process the results to avoid zig zag border.
- Use the results of this work to current research projects (texture atlas construction and segmentation of the viewing sphere to work with panoramas). The flexibility of metrics and densities will be useful.

Future Work

- Refine region borders, simplify interiors.

Future Work

- Refine region borders, simplify interiors.
- Avoid reprocessing converged regions.

References

- D. Cohen-Steiner, P. Alliez, M. Desbrun: *Variational Shape Approximation*. ACM Siggraph Papers, pp. 905-914, 2004.
- Q. Du, V. Faber, M. Gunzburger: *Centroidal Voronoi Tessellations: Applications and Algorithms*. SIAM Review, Vol. 41, No. 4, pp. 637-676, 1999.
- Q. Du, M. Gunzburger, L. Ju: *Constrained Centroidal Voronoi Tessellations on General Surfaces*. Report No. AM253, Math. Dept., Penn State, 2002.

References

- M. Meyer, M. Desbrun, P. Schroder, A.H. Barr: *Discrete Differential-Geometry Operators for Triangulated 2-Manifolds*. Caltech, UCS, 2002.
- G. Peyre, L. Cohen: *Surface Segmentation Using Geodesic Centroidal Tessellation*. Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium, IEEE Computer Society 2004.

Thanks a lot!

Home page: *http* : //www.impa.br/ leo-ks/comp_geometry