Assignment Four: Geometric Algorithm for Spherical Optimal Transportation Map

David Gu

Yau Mathematics Science Center
Tsinghua University
Computer Science Department
Stony Brook University

gu@cs.stonybrook.edu

November 19, 2020

Convex Geometric View

Alexandrov Problem

Alexandrov Problem

Given discrete points $\{x_1, x_2, \cdots, x_k\}$ on the unit sphere $\mathbb{S}^2$, define a discrete spherical function $\rho : \mathbb{S}^2 \to \mathbb{R}^+$, $\rho(x) = \sum_{i=1}^k \rho_i \delta(x - x_i)$, the radio graph of ρ is the convex hull

$$S_\rho = \mathrm{conv}(\{\rho_1 x_1, \rho_2 x_2, \cdots, \rho_k x_k\}).$$

Given the discrete Gaussian curvature at the vertices of S_ρ, $\{\nu_1, \nu_2, \cdots, \nu_k\}$, satisfying Gauss-Bonnet theorem,

$$\sum_{i=1}^k \nu_i = 2\pi\chi(\mathbb{S}^2) = 4\pi, \quad \nu_i > 0,$$

then find ρ and S_ρ.

Input Mesh

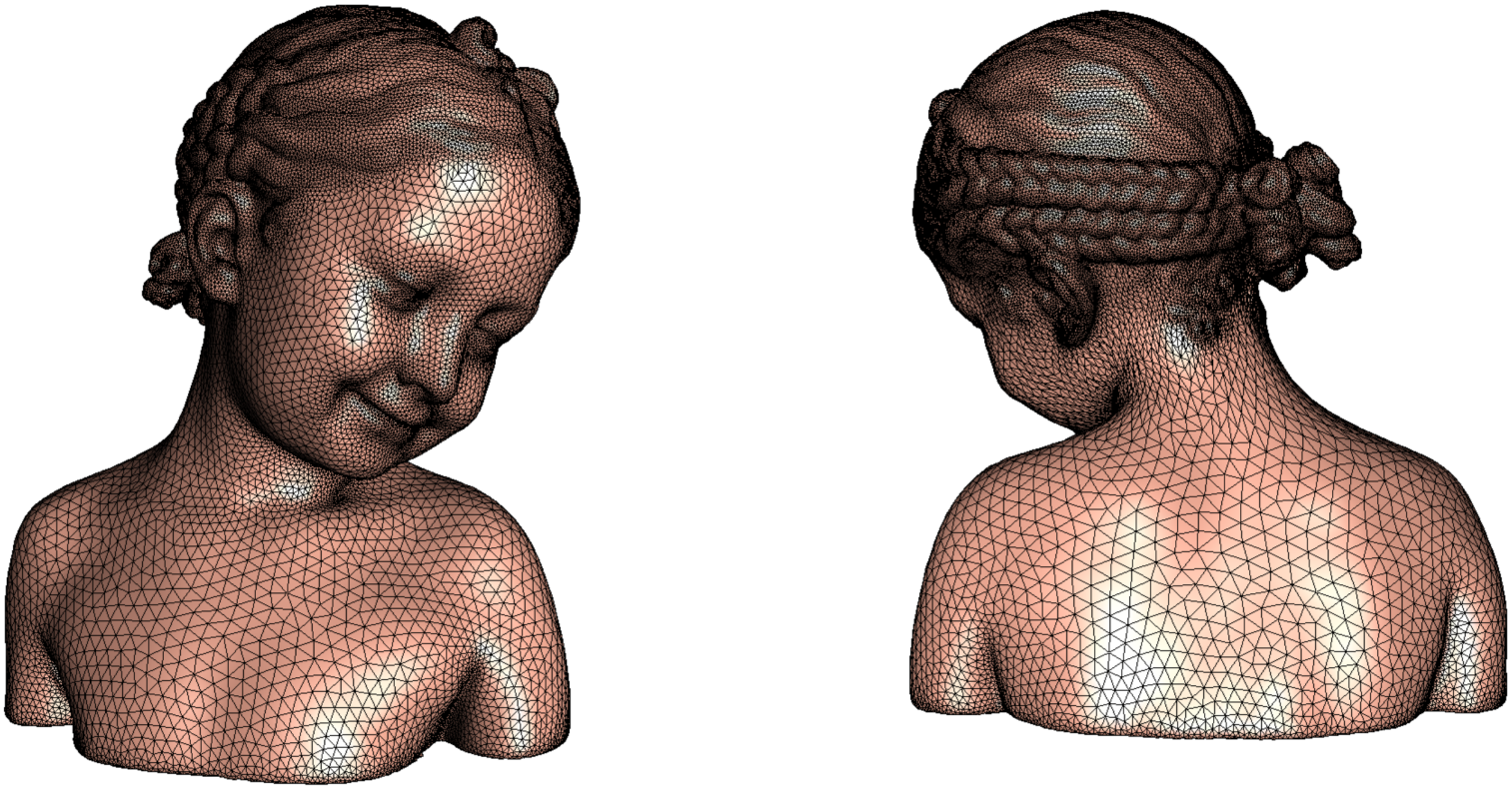


Figure: Input mesh M.

Input Mesh

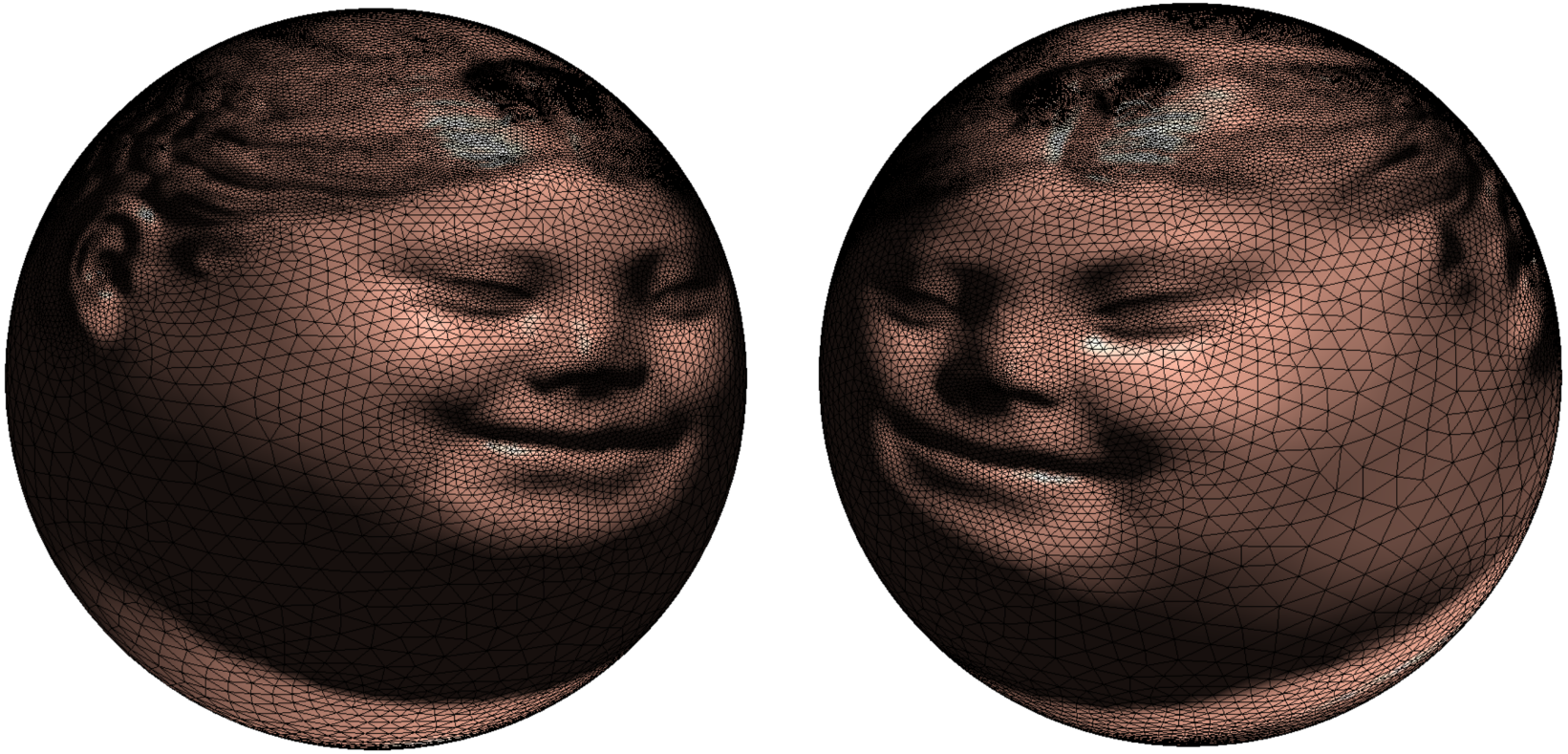
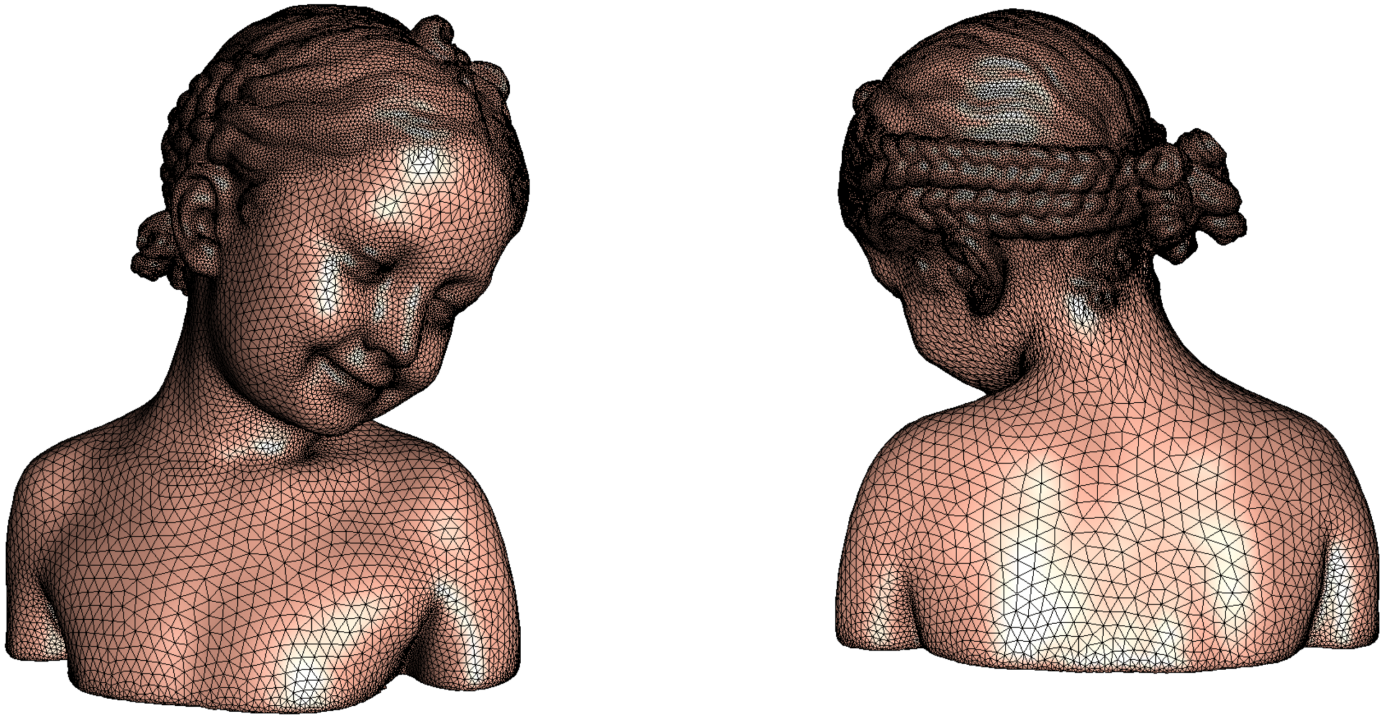


Figure: Spherical harmonic map to $\mathbb{S}^2$, (CCG homework 4).

The spherical harmonic map $\varphi : M \to \mathbb{S}^2$, the image of all the vertices, $\{\varphi(v_1), \varphi(v_2), \cdots, \varphi(v_k)\}$ are treated as $\{x_i\}_{i=1}^k$.

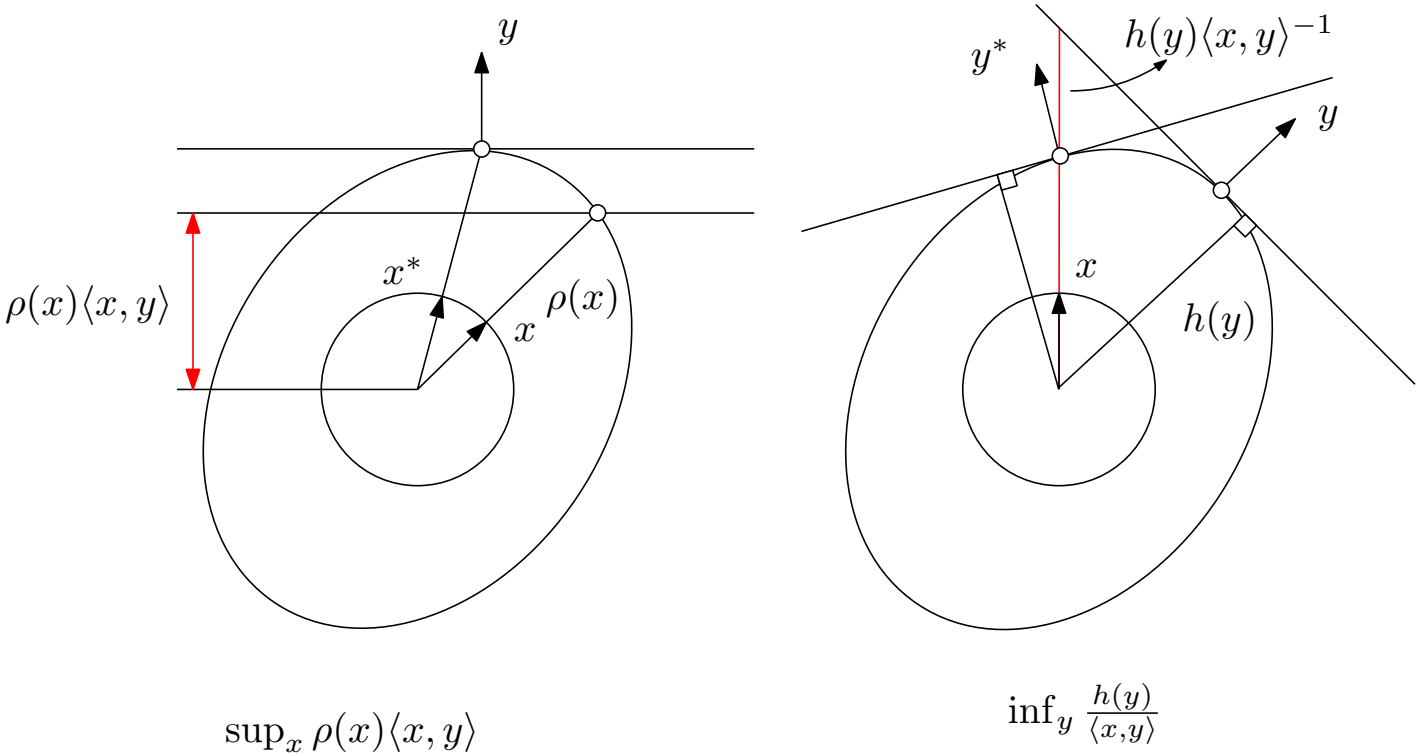
Input Mesh



The total area of M is normalized to 4π. Each vertex $v_i \in M$ is adjacent to faces f_i^{jk} with vertices $[v_i, v_j, v_k]$, then the Diract measure ν_i is defined as the one third of the total areas of f_i^{jk}'s,

$$\nu_i = \frac{1}{3} \sum_{j,k} \text{Area}(f_i^{jk}).$$

Supporting Planes



Given a supporting plane with normal $y \in \mathbb{S}^2$, height is denoted as $\rho^*(y)$, then

$$\rho^*(y) = \sup_{x \in \mathbb{S}^2} \rho(x)\langle y, x\rangle.$$

Supporting Planes

By

$$\rho^*(y) = \sup_{x\in\mathbb{S}^2} \rho(x)\langle y, x\rangle \iff \frac{1}{\rho^*(y)} = \inf_{x\in\mathbb{S}^2} \frac{1}{\rho(x)} \frac{1}{\langle x, y\rangle}$$

Denote $\eta(y) := \frac{1}{\rho^*(y)}$, then we obtain

$$\eta(y) = \inf_{x\in\mathbb{S}^2} \frac{1}{\rho(x)} \frac{1}{\langle x, y\rangle}, \quad \rho(x)\eta(y) \leq \frac{1}{\langle x, y\rangle}$$

Therefore $\varphi(x) = \log \rho(x)$, $\psi(y) = \log \eta(y)$,

$$\varphi(x) + \psi(y) \leq -\log\langle x, y\rangle,$$

let $c(x, y)$ be $-\log\langle x, y\rangle$, we obtain c-transform,

$$\varphi^c(y) := \inf_{x\in\mathbb{S}^2} c(x, y) - \varphi(x).$$

Spherical Optimal Transportation

This gives the Kantorovich formulation of spherical optimal transportation!

$$\sup\left\{\int_{\mathbb{S}^2}\varphi(x)f(x)dx+\int_{\mathbb{S}^2}\psi(y)g(y)dy,\varphi(x)+\psi(y)\leq c(x,y)\right\}$$

By c-transform,

$$\varphi^c(y):=\inf_{x\in\mathbb{S}^2}c(x,y)-\varphi(x).$$

we optimize the functional by finding $\{(\varphi_k,\psi_k)\}$, where

$$\psi_k=\phi_k^c,\quad \phi_{k+1}=\psi_k^c,$$

(φ_k,ψ_k)'s are bounded, whose Lipschitz constant equals to the $\sup\nabla c(x,y)$ on $\mathbb{S}^2$, the energy is monotonously increasing. This shows the existence of the solution.

Legendre Dual

$\rho : \mathbb{S}^2 \to \mathbb{R}^+$ has a Legendre dual $\rho^* : \mathbb{R}^{\mathbb{R}+}$,

$$\rho^*(y) = \max_{i=1}^{k} \rho_i \langle x_i, y \rangle, \iff \frac{1}{\rho^*(y)} = \min_{i=1}^{k} \frac{1}{\rho_i} \frac{1}{\langle x_i, y \rangle},$$

The radial graph of $1/\rho*(y)$ is the envelope of the planes

$$\pi_\rho^i(y) = \frac{1}{\rho_i} \frac{1}{\langle x_i, y \rangle}.$$

S_{ρ^*} is given by

$$S_{\rho^*} = \mathrm{Env}\{\pi_\rho^1, \pi_\rho^2, \ldots, \pi_\rho^k\} = \Gamma\left(\frac{1}{\rho^*}\right).$$

Face Dual Point __face_dual_point(CFace* pf)

Every face on the convex hull $f = [\rho_i x_i, \rho_j x_j, \rho_k x_k]$, is dual to a point $f^* = \pi_i \cap \pi_j \cap \pi_k$, where

$$\pi_i(y) = \frac{1}{\rho_i \langle x_i, y \rangle}, \pi_j(y) = \frac{1}{\rho_i \langle x_j, y \rangle}, \pi_k(y) = \frac{1}{\rho_i \langle x_k, y \rangle},$$

Then $f^* = \lambda n$, where n is the normal to the face, and

$$\lambda = \pi_i(n) = \pi_j(n) = \pi_k(n).$$

Assume the intersection point is d, then

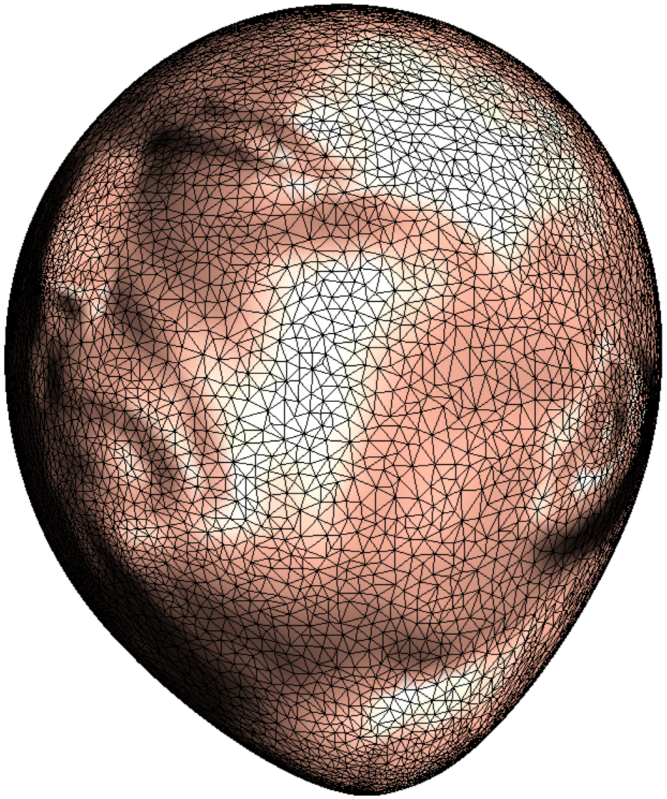
$$\langle \rho_i x_i, d \rangle = \langle \rho_j x_j, d \rangle = \langle \rho_k x_k, d \rangle$$

hence

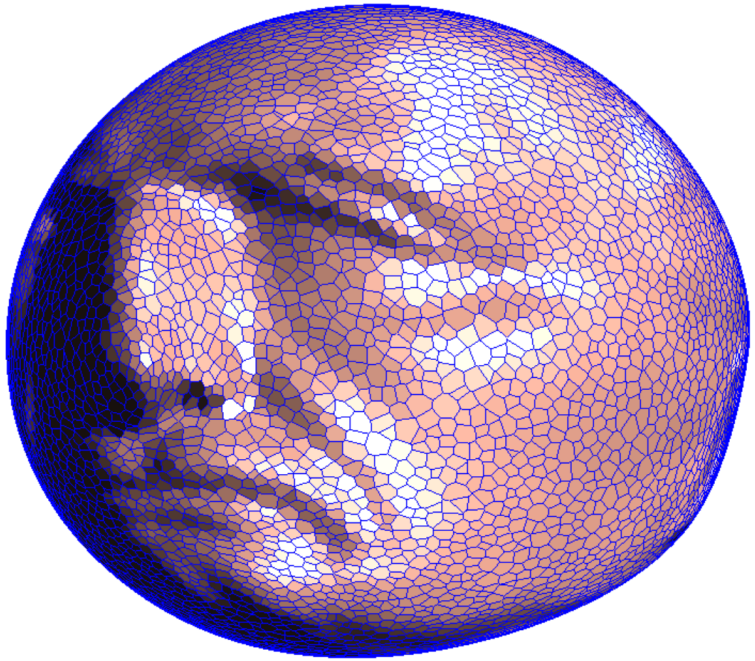
$$d \perp (\rho_i x_i - \rho_j x_j) \quad d \perp (\rho_j x_j - \rho_k x_k),$$

d is along the normal direction. f^* is recorded as $face \to dual\_point()$.

Legendre Dual



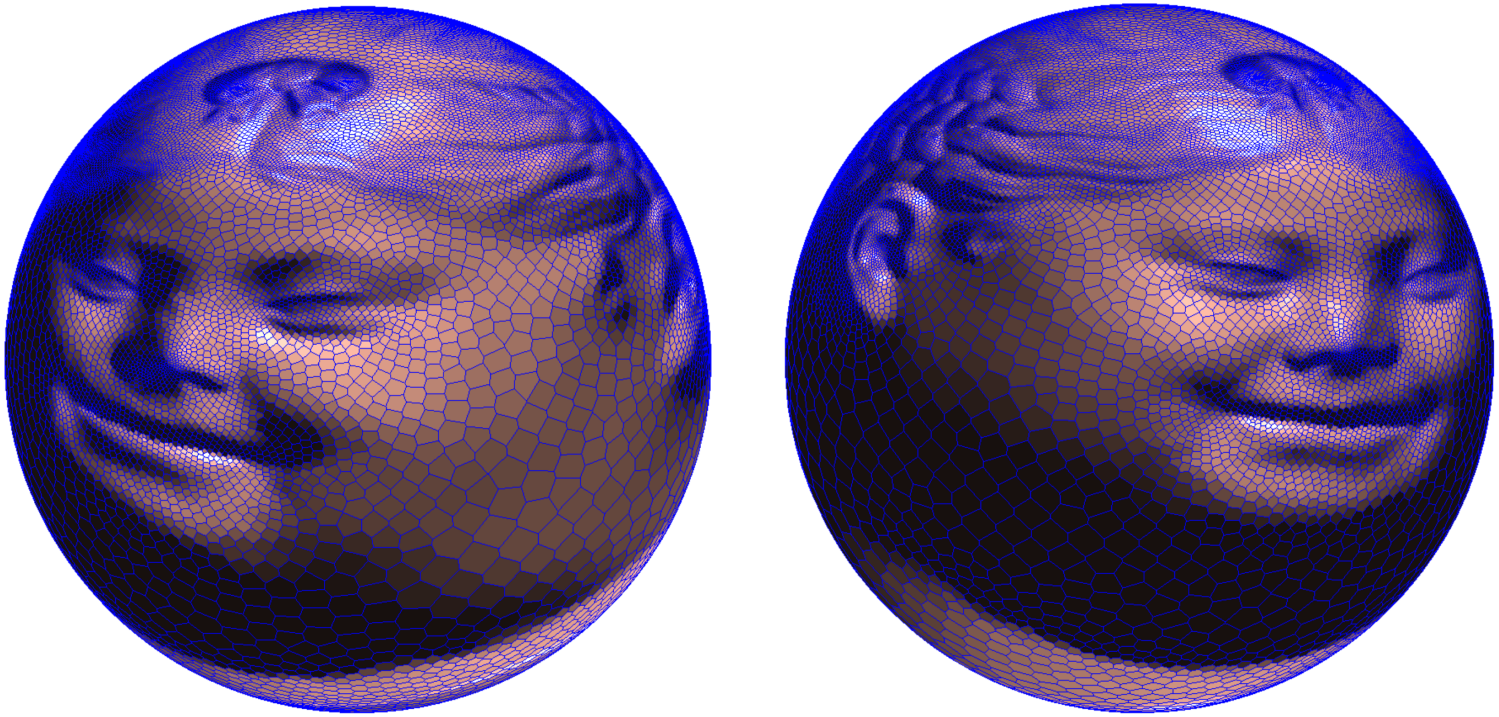
$\mathrm{Conv}(\{\rho_i x_i\})$



$\mathrm{Env}(\{\rho(y) = \frac{1}{\rho_i}\frac{1}{\langle x_i, y\rangle}\})$

Figure: Convex hull and envelope. For each vertex v_i on the left convex hull, the dual points $(f_i^{jk})^*$ of the surrounding faces f_i^{jk} gives the dual face of the envelope on the right side.

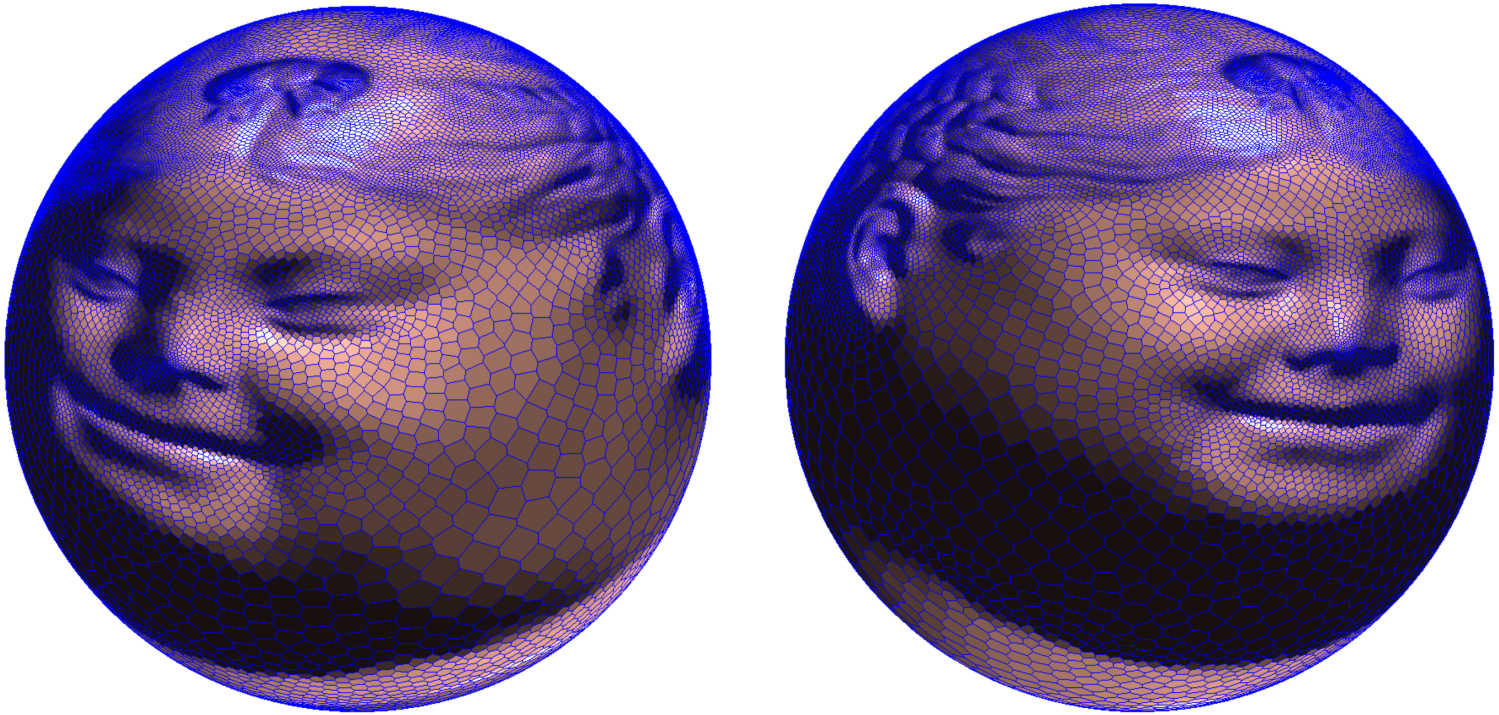
Spherical Power Diagram



Each face of the envelope is recorded as $vertex \to dual\_cell3D()$. The central projection of the envelope to the sphere, induces a spherical power diagram,

$$\mathbb{S} = \bigcup_{i=1}^{k} W_\rho(i), \quad W_\rho(i) := \{y \in \mathbb{S}^2 | \pi_\rho^i(y) \leq \pi_\rho^j(y)\}.$$

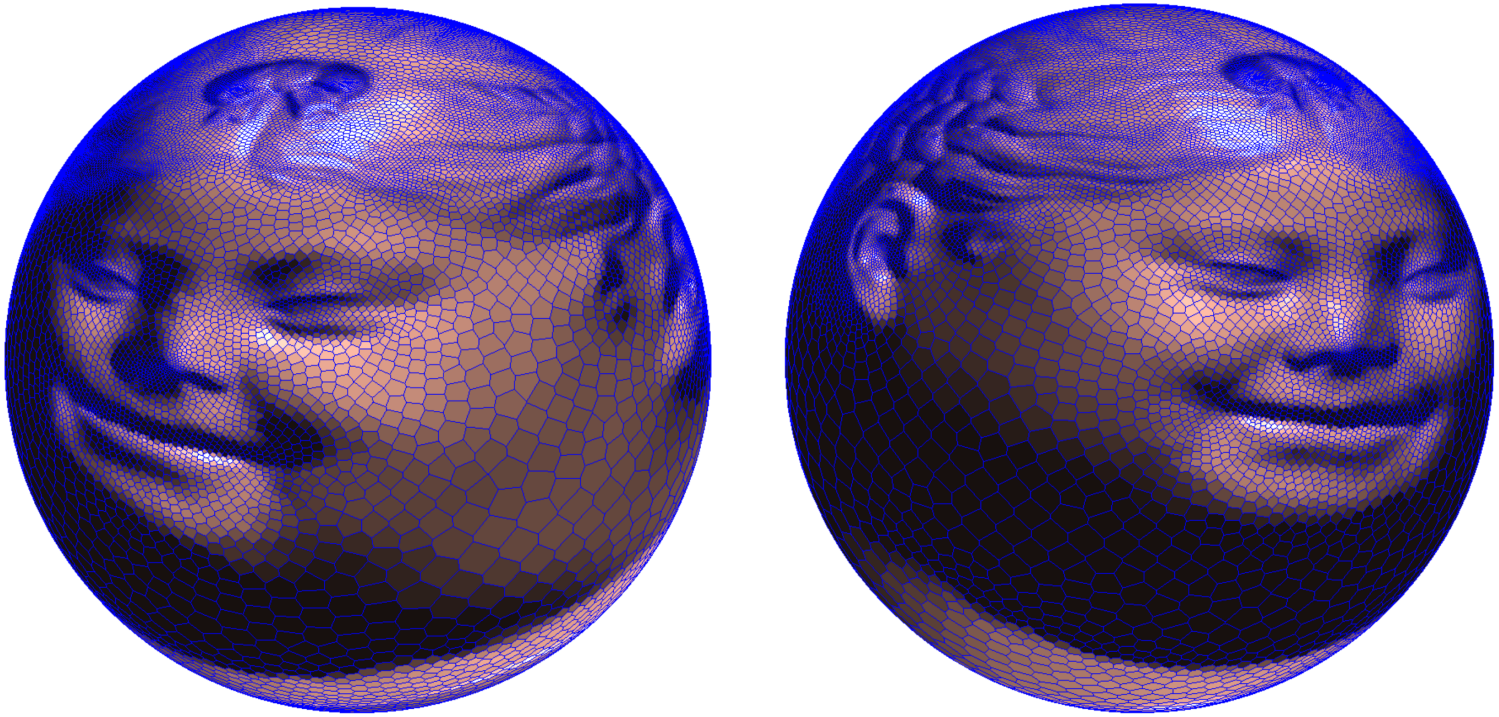
Spherical Power Diagram



Define $h_i = \log \rho_i$, suppose $w_i(\mathbf{h})$ is the spherical area of the cell $W_\rho(i)$, the convex energy

$$E(\mathbf{h}) := \int^{\rho} \sum_{i=1}^{k} w_i(\mathbf{h}) dh_i - \sum_{i=1}^{k} \nu_i h_i.$$

Optimization



The optimization is performed in the admissible space

$$\mathcal{H} := \left\{ \mathbf{h} \in \mathbb{R}^k : w_i(\mathbf{h}) > 0, \forall i \right\} \bigcap \left\{ \sum_{i=1}^{k} h_i = 0 \right\}.$$

Gradient

The gradient of the energy is given by

$$\nabla E(\mathbf{h}) = (w_1(\mathbf{h}) - \nu_1, w_2(\mathbf{h}) - \nu_2, \cdots, w_k(\mathbf{h}) - \nu_k).$$

Note that, all the power cells are convex spherical geodesic polygons. We subdivide the polygon into geodesic triangles, according to Gauss-Bonnet theorem, the area of the geodesic triangle is given by

$$A + B + C - \pi = \text{Area}(\Delta).$$

Suppose the edge lengths of the Δ are $\{a, b, c\}$, inner angles $\{A, B, C\}$, the spherical cosine law is

$$\cos c = \cos a \cos b + \sin a \sin b \cos C.$$

The area of each spherical power cell is recorded as $vertex \to dual\_area()$.

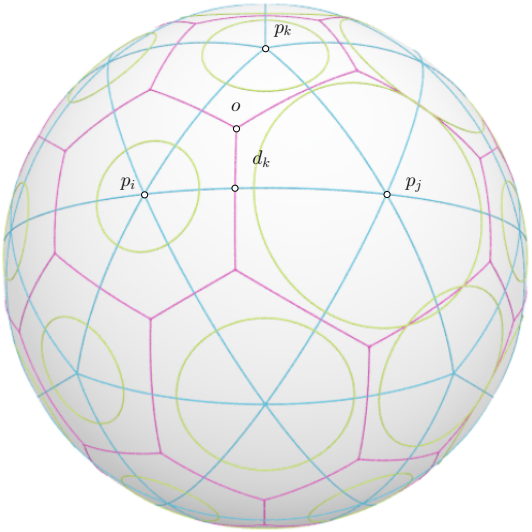
Face Power Center o_f __face_power_center(CFace* pf)

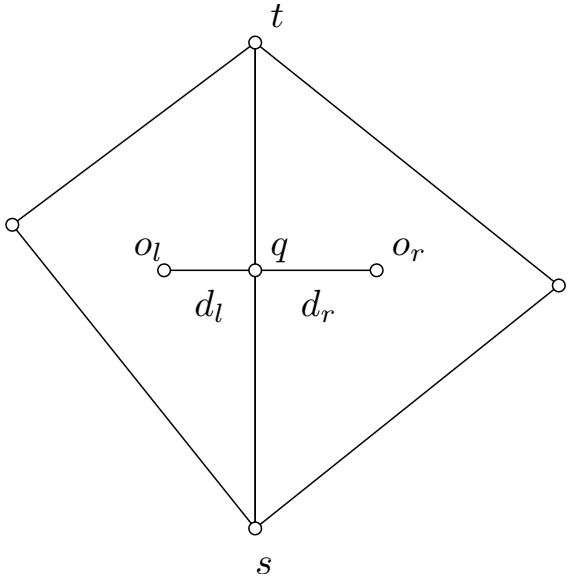
Every face on the convex hull $f = [\rho_i x_i, \rho_j x_j, \rho_k x_k]$, the power center $o_f \in \mathbb{S}^2$ satisfies

$$R_f = \langle \rho_i x_i, o_f \rangle = \langle \rho_j x_j, o_f \rangle = \langle \rho_k x_k, o_f \rangle$$

hence o_f is the normal n_f to the face f, R_f is the power of f. The face power center o_f is recorded as $face \to spherical\_power\_center()$, the face power R_f is recorded as $face \to spherical\_power\_radius()$.

Distance from power center to edge d_l __halfedge_spherical_height()



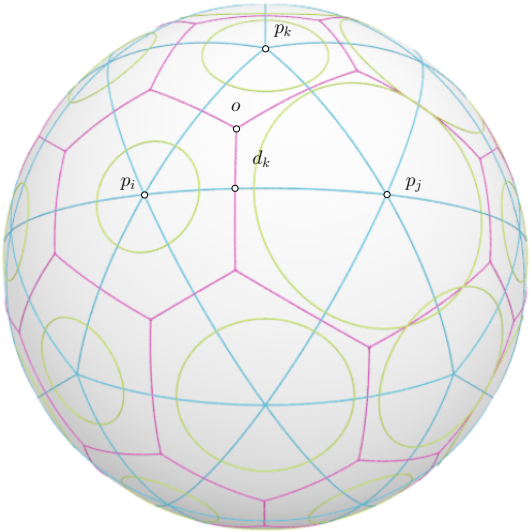


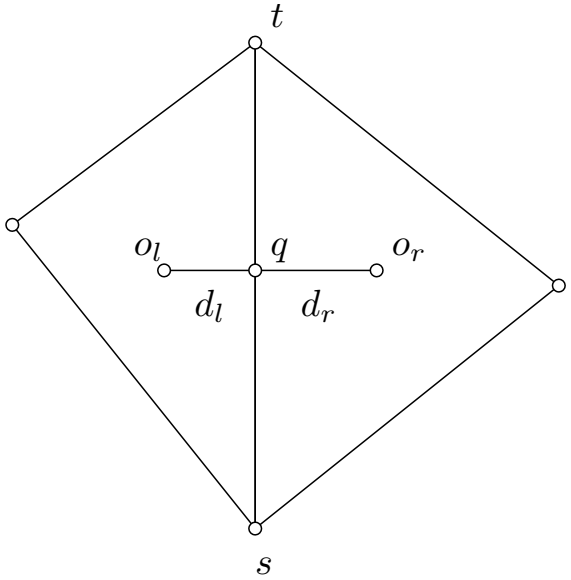
The perpendicular foot q is the intersection of the plane through the sphere center and s, t and the plane through the sphere center and the power centers o_l, o_r, hence

$$q = \frac{(t \times s) \times (o_l \times o_r)}{|(t \times s) \times (o_l \times o_r)|},$$

Note that d_l is an oriented distance, if o_l is outside the left triangle, then $d_l < 0$, recorded as $halfedge \to spherical\_height()$.

Spherical Edge Length γ_{ij} __edge_spherical_length()



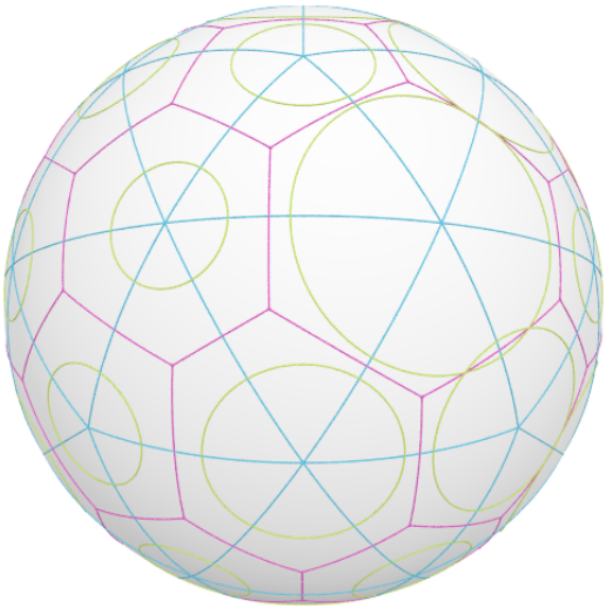


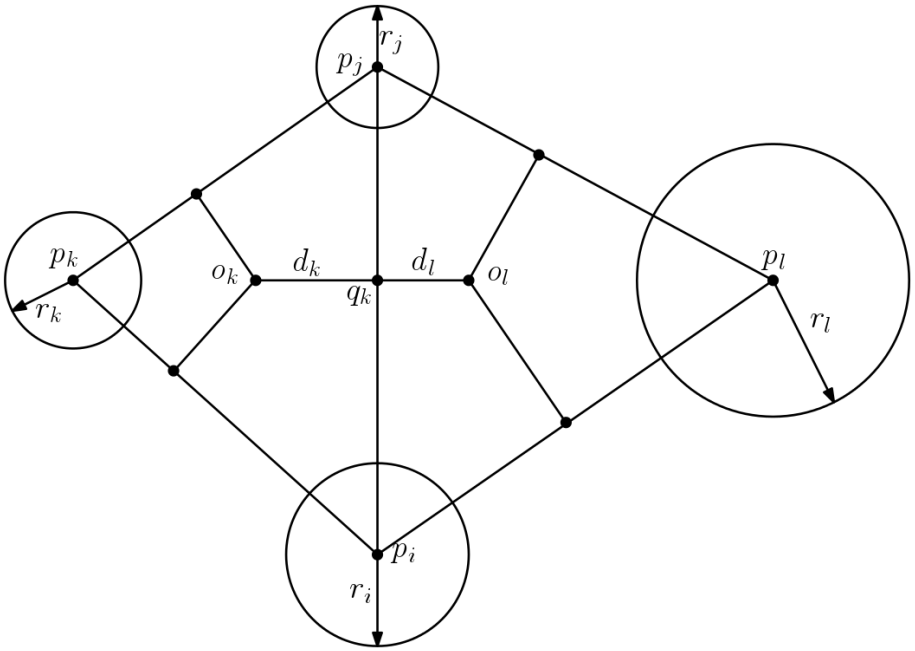
The spherical edge length of $[\rho_i x_i, \rho_j x_j]$,

$$\gamma_{ij} = \cos^{-1}\langle x_i, x_j \rangle,$$

γ_{ij} is recorded as *edge* $\rightarrow$ *spherical_length*().

Edge weight w_{ij} _edge_interior_weight(CEdge* pe)





$$w_{ij} = \frac{\partial w_i}{\partial h_j} = \frac{\partial w_j}{\partial h_i} = -\frac{1}{\rho_i \rho_j \sin \gamma_{ij}} \left(\frac{R_l^2 \sin d_l}{\cos^2 d_l} + \frac{R_k^2 \sin d_k}{\cos^2 d_k} \right) \tag{1}$$

$$\frac{\partial w_i}{\partial h_i} = -\sum_{j \neq i} \frac{\partial w_i}{\partial h_j} \tag{2}$$

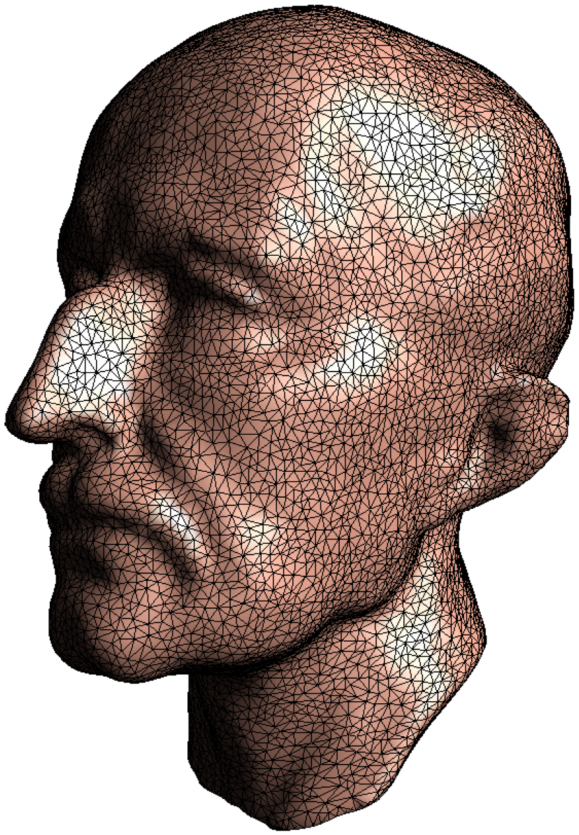
where $h_i = \log \rho_i$. w_{ij} is recorded as $edge \to weight()$.

Computational Geometric Algorithms

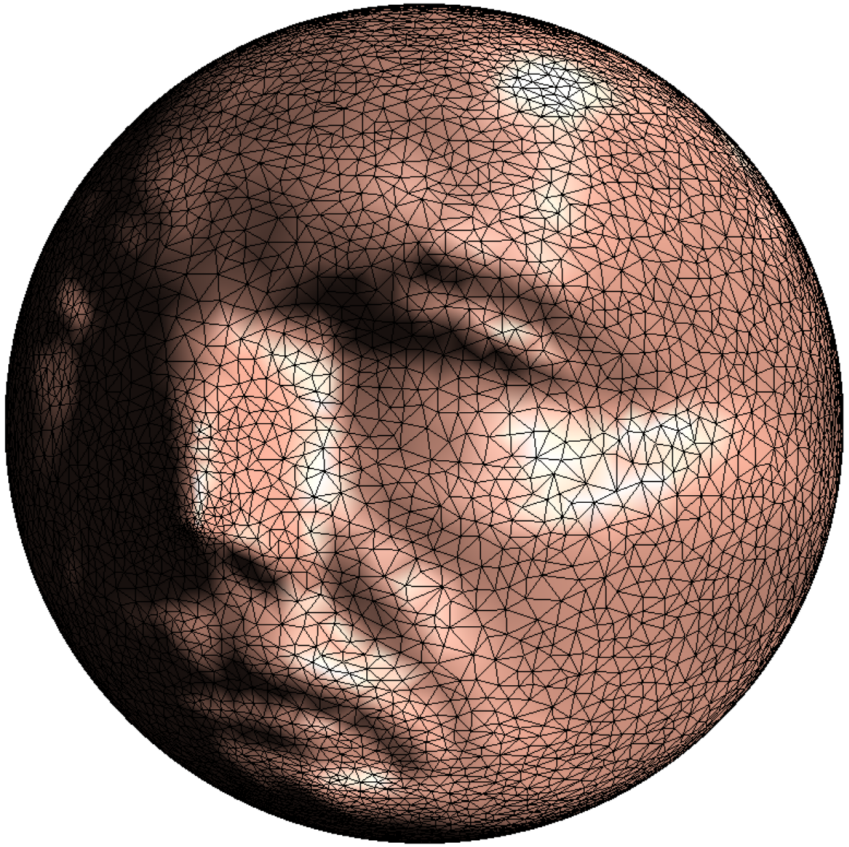
File Format

- The source measure is the uniform distribution on the unit sphere $\mathbb{S}^2$.
- The target measure is represented as a triangle mesh (obj or m format), each vertex has both (x, y, z) coordinates and (u, v, w) parameters. Each vertex v_i represents a sample $x_i = (u_i, v_i, w_i)$, (u_i, v_i, w_i) specifying the spherical position in $\mathbb{S}^2$. The summation of the areas of all triangular faces adjacent to v_i is treated as ν_i, (after normalization such that the total area is 4π).

File IO



(a). source mesh



(b). target mesh

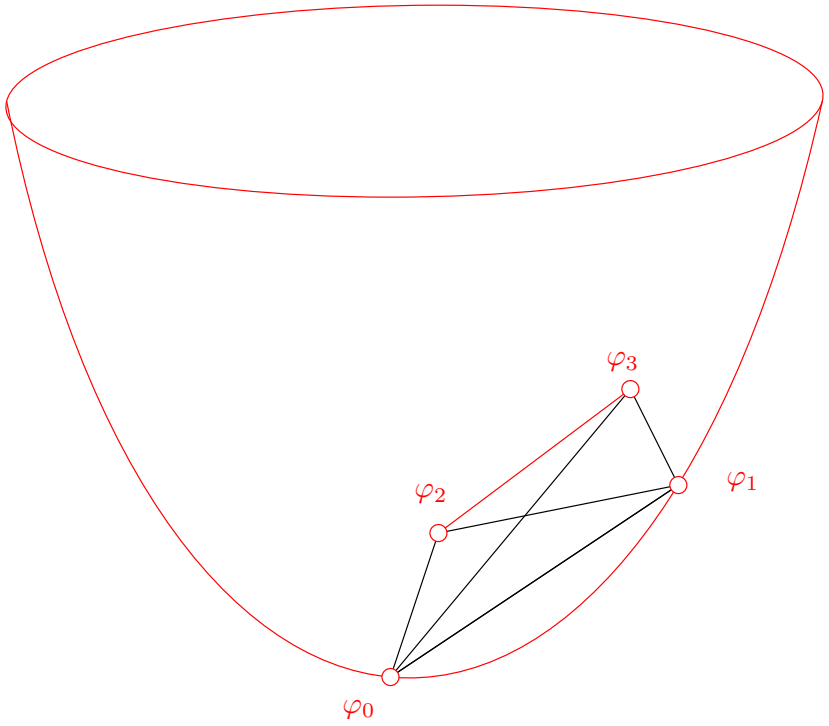
Figure: Input files, source file specifies the vertex positions $(x, y, z) \in \mathbb{R}^3$ and ν, the target file specifies the positions on the sphere $(u, v, w) \in \mathbb{S}^2$.

Data Structure & Algorithms

1. The combinatorial data structure to represent the convex hull and the dual envelope is half-edge;
2. The linear numerical solver is Eigen library;
3. The geometric computation is based on adaptive arithmetic method.
4. The convex hull is based on Lawson's edge flip algorithm.
5. The optimization of Alexandrov energy is based on damping algorithm.

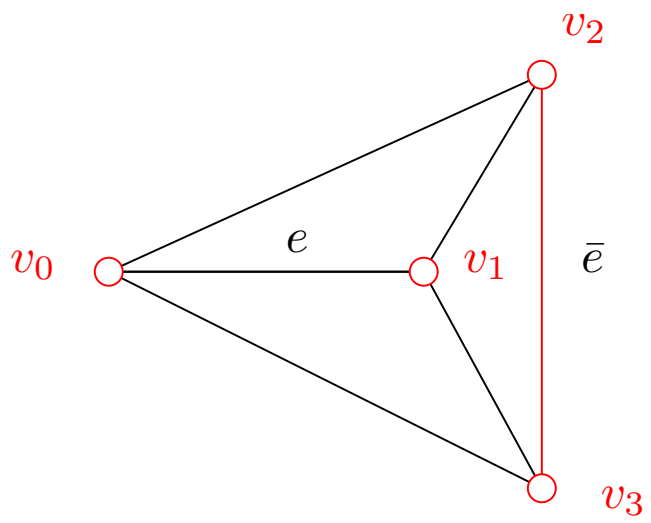
Edge Local convex

Given an edge e in the triangulation $\mathcal{T}$, find the two neighboring faces, suppose vertex v_i is represented as $\varphi_i := \rho_i x_i$, compute the volume of the tetrahedron $[\varphi_0, \varphi_1, \varphi_2, \varphi_3]$. If the volume is positive, then e is locally convex, if the volume is negative, then e is non-locally-convex.



Edge Flippable

Given an edge $e = [v_0, v_1]$ in the triangulation $\mathcal{T}$, if $[v_2, v_3]$ is connected by another edge $\bar{e}$, then the edge is not flippable.



Lawson Edge Flip Algorithm

Input is a set of points S in $\mathbb{R}^3$, the output is the convex hull of S.

1. Construct an initial triangulation of the point set S;
2. Push all non-locally convex edges of $\mathcal{T}$ on stack and mark them;
3. While the stack is non-empty do
    1. $e \leftarrow \text{pop}()$;
    2. unmark e;
    3. if e is locally convex then continue;
    4. if e can't be flipped then continue;
    5. flip edge e;
    6. push other four edges of the two triangles adjacent to e into the stack if unmarked;
4. If there is an edge e, which is not locally convex, then there is some point p_i that is not on the convex hull of S.

Lawson Edge Flip for Convex Hull

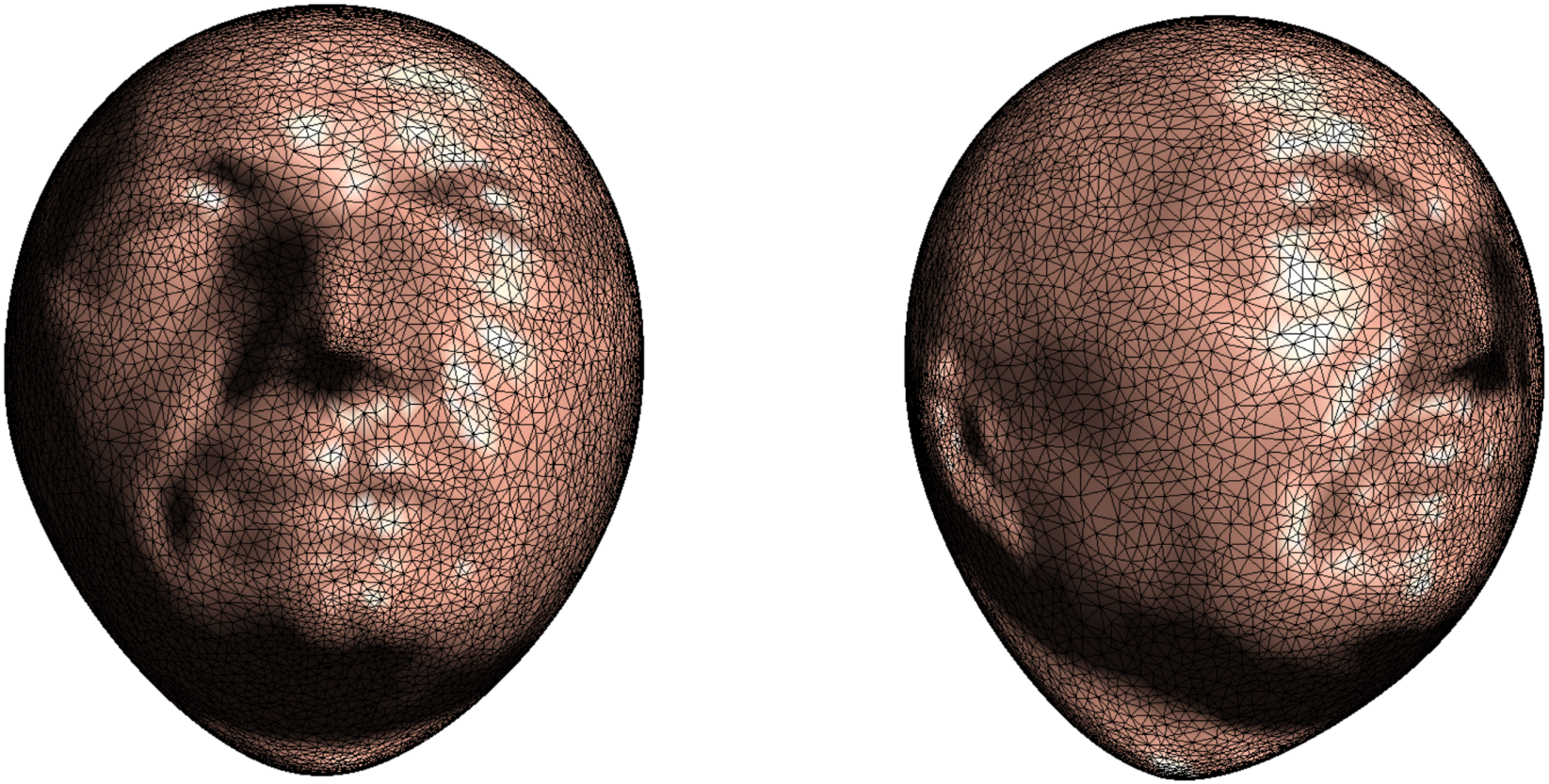


Figure: Construct convex hull of $\{\rho_i x_i\}$, using Lawson Edge Flip algorithm.

Legendre Dual

Given a convex hull, which is the radial graph of a convex function ρ, we compute its Legendre dual $1/\rho^*$. Each point $\rho_i x_i$ on the convex hull represents a plane π_i,

$$\pi_i(y) = \frac{1}{\rho_i} \frac{1}{\langle x_i, y \rangle}.$$

Each face $[\rho_i x_i, \rho_j x_j, \rho_k x_k]$ is dual to a point f^* satisfying the linear equation group,

$$\langle \rho_i x_i, f^* \rangle = \langle \rho_j x_j, f^* \rangle = \langle \rho_k x_k, f^* \rangle$$

Envelope

Given the convex hull $\{\rho_i x_i\}$, each face f_α is dual to a point f_α^*; each vertex v_i is dual to a supporting plane v_i^*.

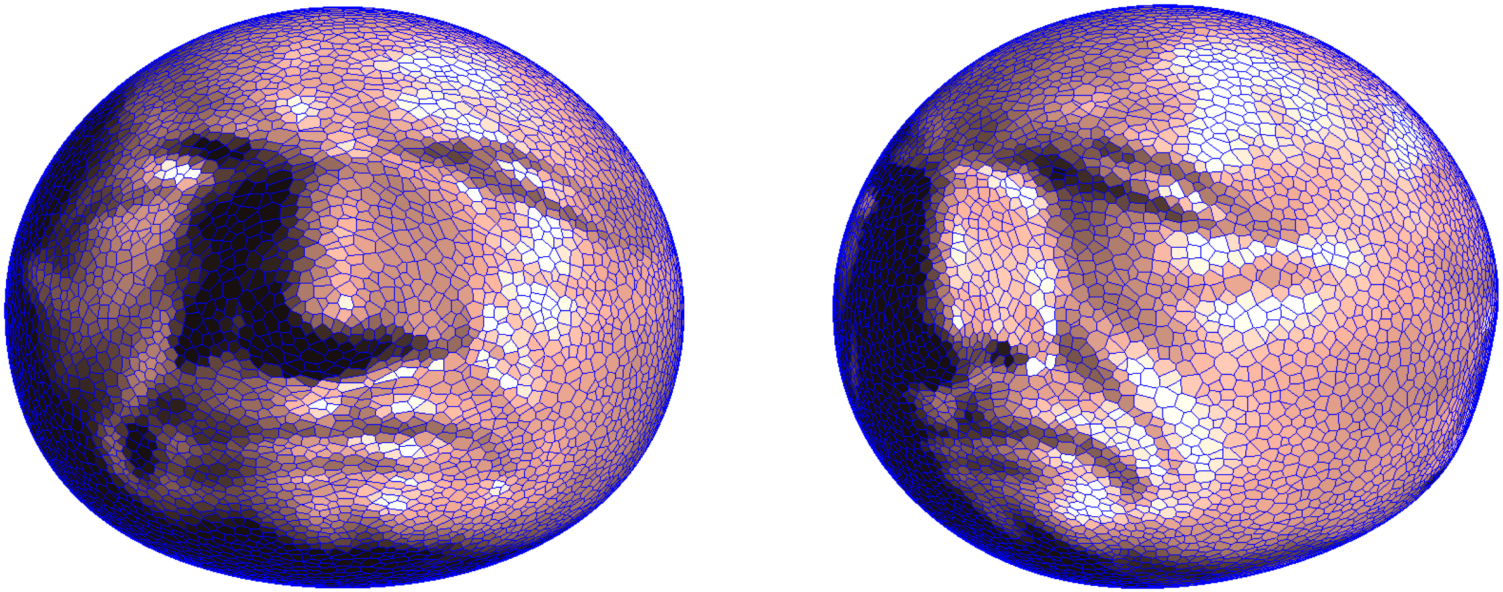
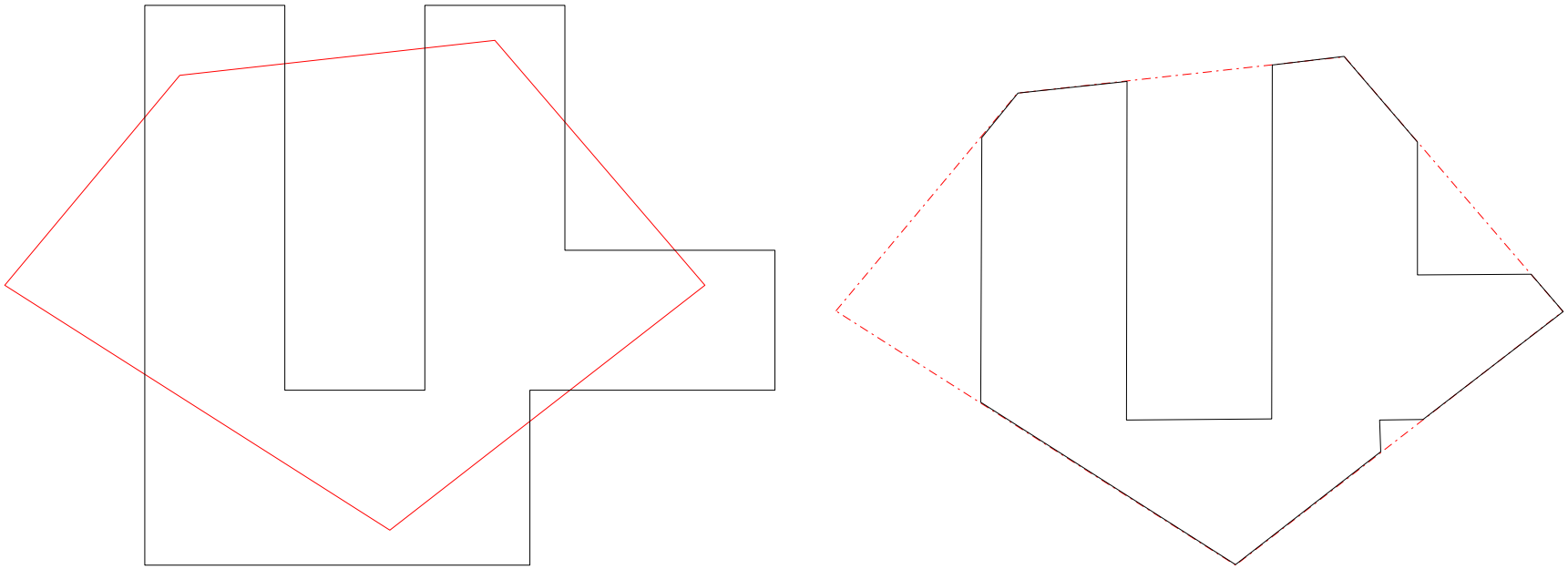


Figure: Legendre dual of the convex hull is the envelope.

Sutherland–Hodgman algorithm

Given a subject polygon S and a convex clipping polygon C, we use C to clip S. Each time, we use one edge e of C to cut off a corner of S.



Sutherland–Hodgman algorithm

```
foreach Edge clipEdge in clipPolygon do
    List inputList ← outputList;
    outputList.clear();
    foreach Edge [p_{k-1}, p_k] in inputList do
        Point q ← ComputeIntersection(p_{k-1}, p_k, clipEdge);
        if p_k inside clipEdge then
            if p_{k-1} not inside clipEdge then
                outputList.add(q);
            end
            outputList.add(p_k);
        end
        else if p_{k-1} inside clipEdge then
            outputList.add(q)
        end
    end
```

Spherical Power Diagram Algorithm

1. Compute the convex hull using Lawson edge flipping;
2. Compute the envelope using Legendre dual algorithm and project the envelope to the spherical power diagram $\mathcal{D}$;
3. Clip the power cells using Sutherland-Hodgman algorithm, if necessary;

Damping Algorithm

1. Initialize the step length λ;
2. $\varphi_i \leftarrow \varphi_i e^{\lambda d_i}$;
3. Compute the convex hull using Lawson edge flipping;
4. If the convex hull misses any vertex, then $\lambda \leftarrow \frac{1}{2}\lambda$, repeat step 2 and step 3;
5. Compute the upper envelope using Legendre dual algorithm, project to the power diagram $\mathcal{D}$;
6. If necessary, clip the power cells using Sutherland-Hodgman algorithm;
7. If any power cell is empty, then $\lambda \leftarrow \frac{1}{2}\lambda$, repeat step 5 and step 6;

Newton's Method

Input: $\{x_1, x_2, \ldots, x_k\} \subset \mathbb{S}^2$, $\{\nu_1, \nu_2, \ldots, \nu_k\}$, $\sum_{i=1}^{k} \nu_i = 4\pi$, $\nu_i > 0$;
Output: $\mathrm{Conv}\{\rho_1 x_1, \rho_2 x_2, \ldots, \rho_k x_k\}$ realizing discrete curvature ν_i's.

1. Initialize φ as $\varphi_i \leftarrow x_i$;
2. Call the spherical power diagram algorithm;
3. Compute the gradient ∇E, the target area minus the current power cell area;
4. Compute the Hessian matrix H, using the power diagram edge length;
5. Compute the update direction $Hd = \nabla E$;
6. Call the damping algorithm, set $\varphi \leftarrow \varphi e^{\lambda d}$, such that φ is admissible;
7. Repeat step 2 through step 6, until the gradient is close to 0.

Computational Examples

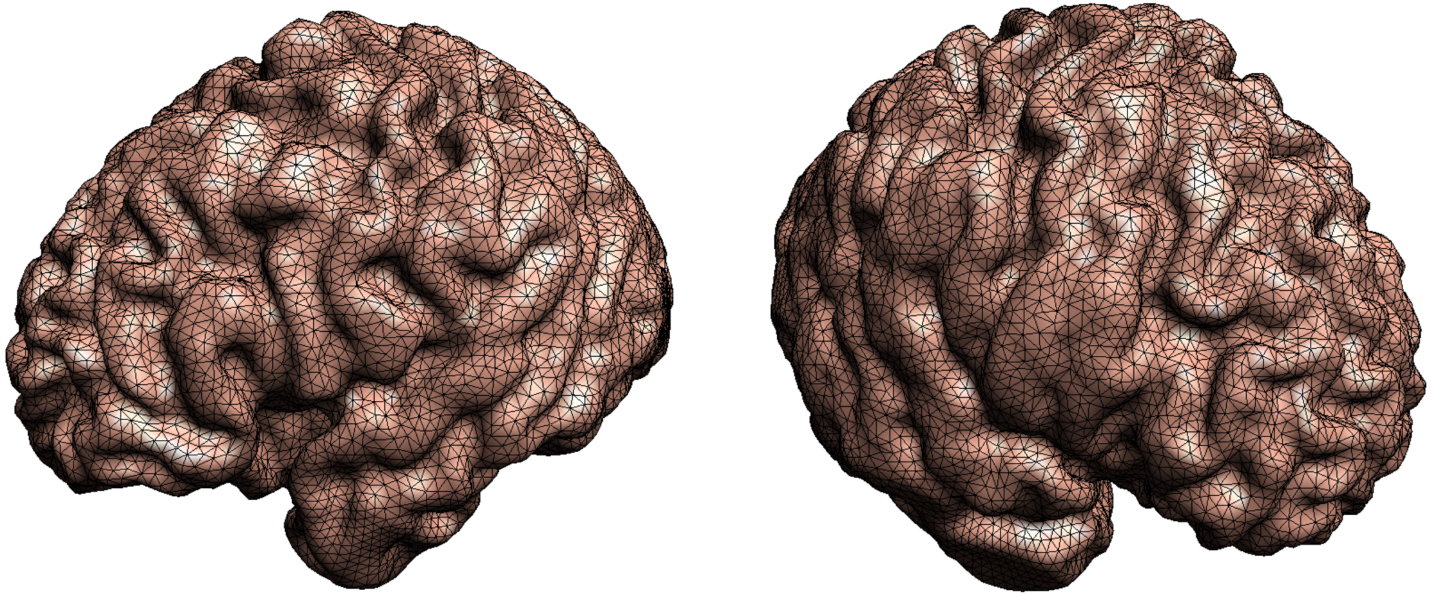


Figure: Input brain mesh.

Computational Examples

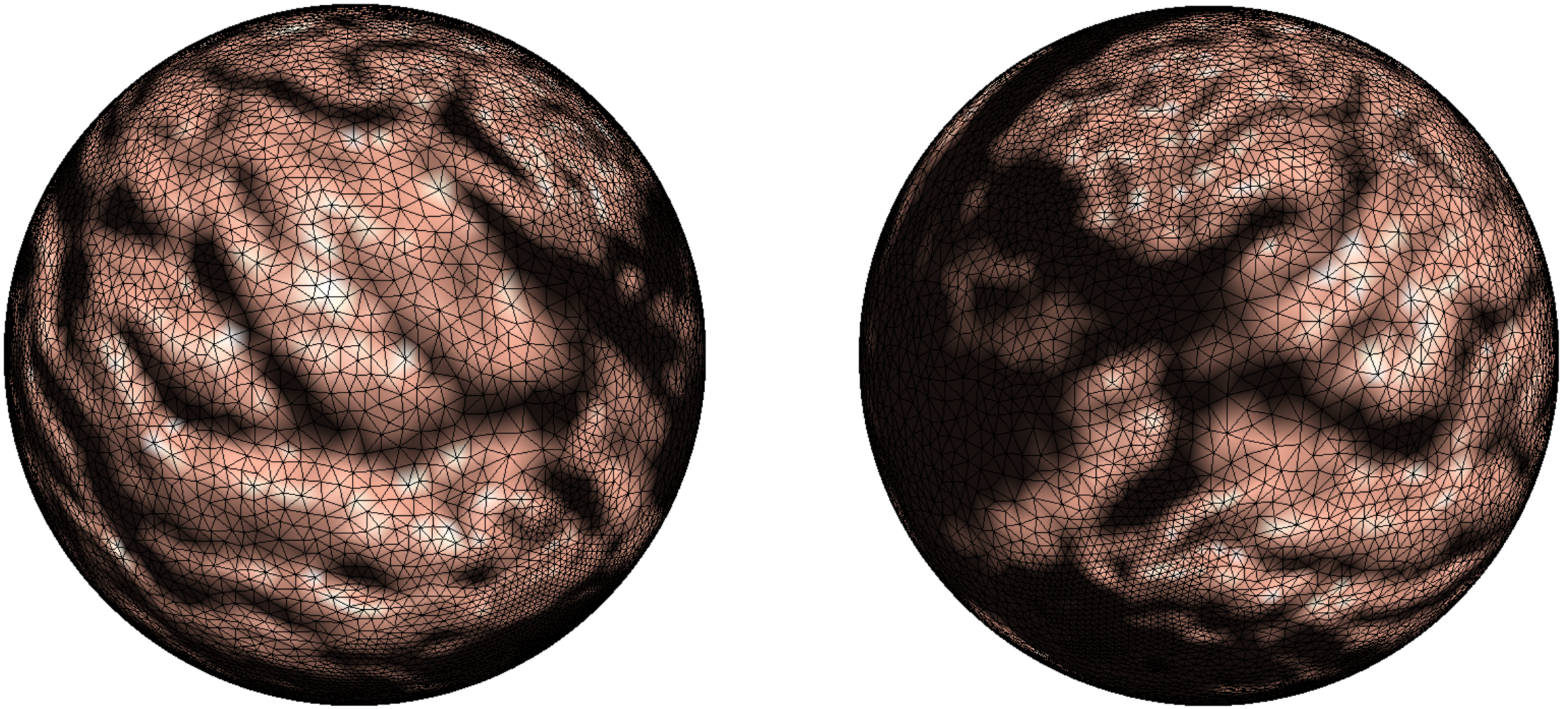


Figure: Initial harmonic map.

Computational Examples

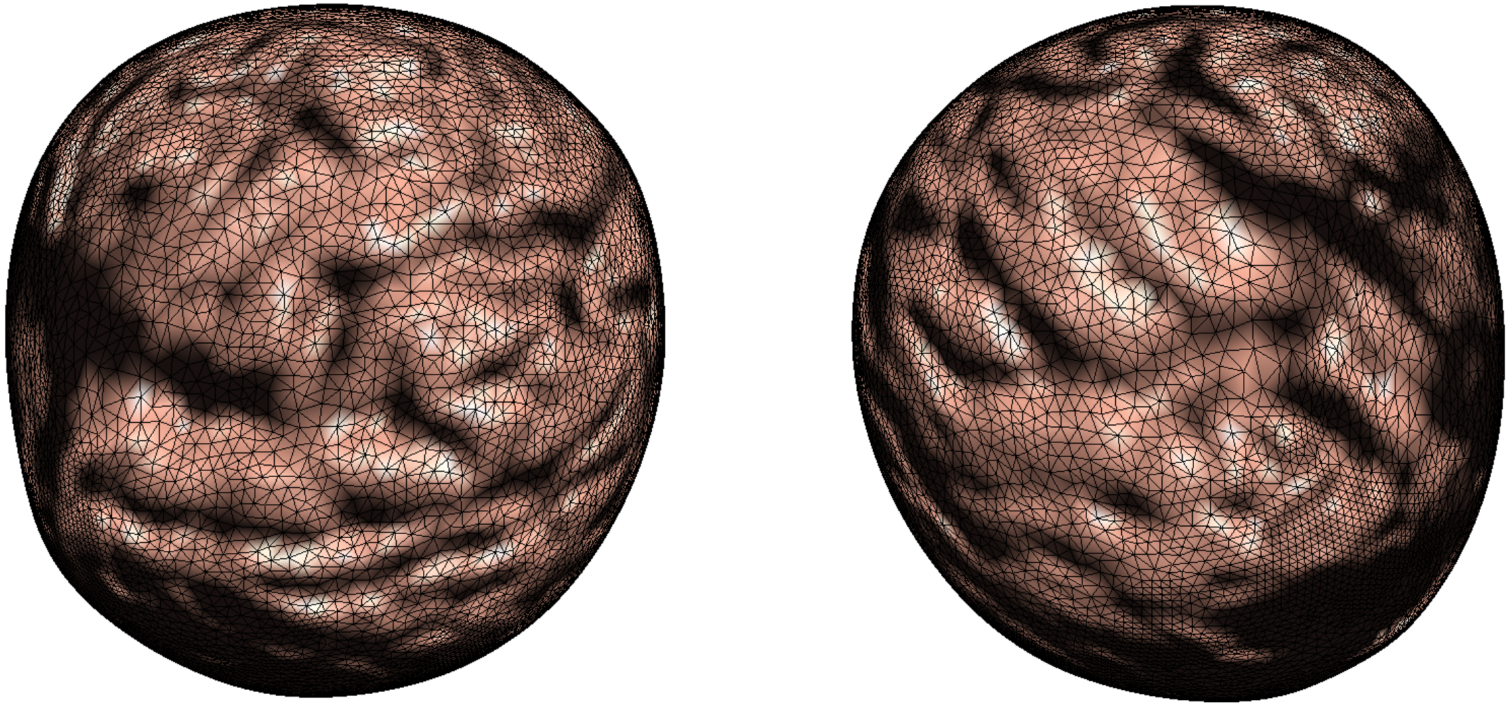


Figure: Final convex hull S_ρ.

Computational Examples

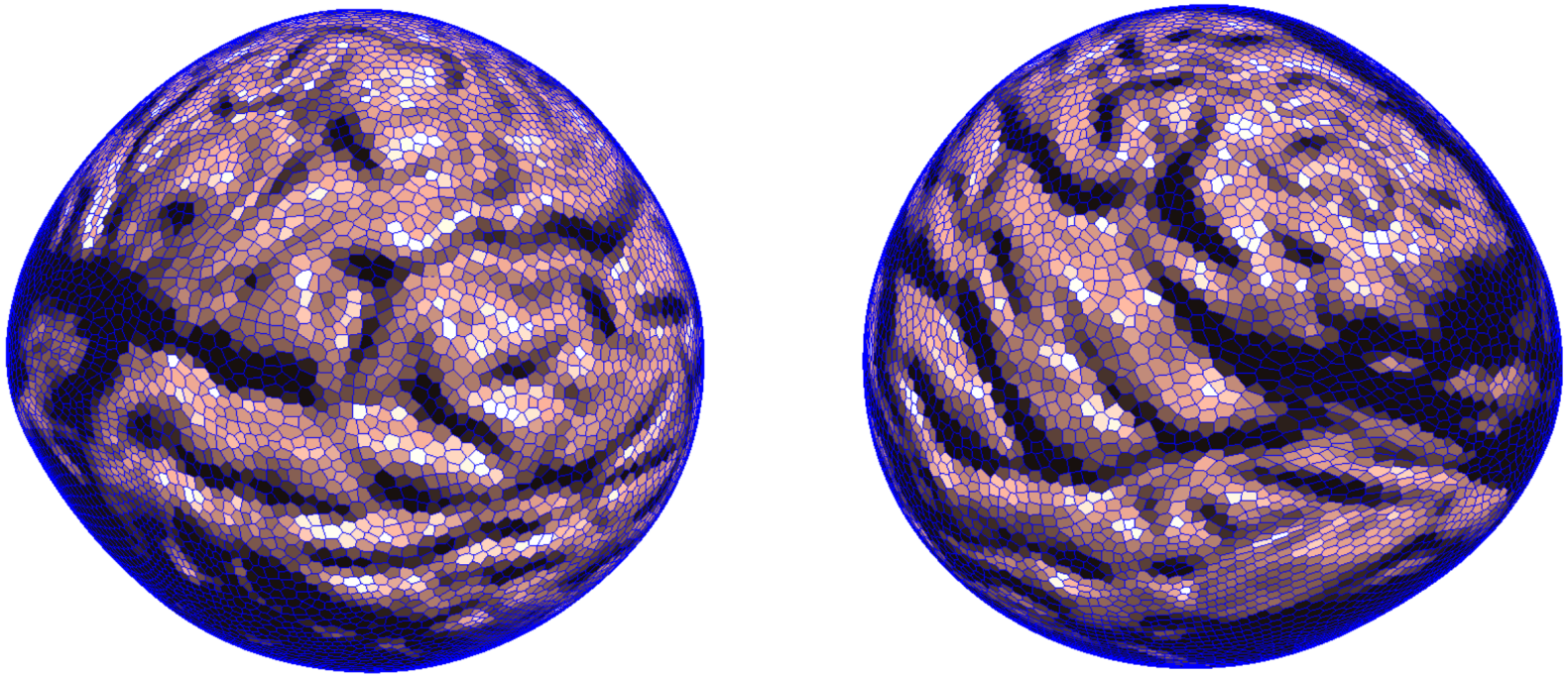


Figure: Final envelope S_{ρ^*}.

Computational Examples

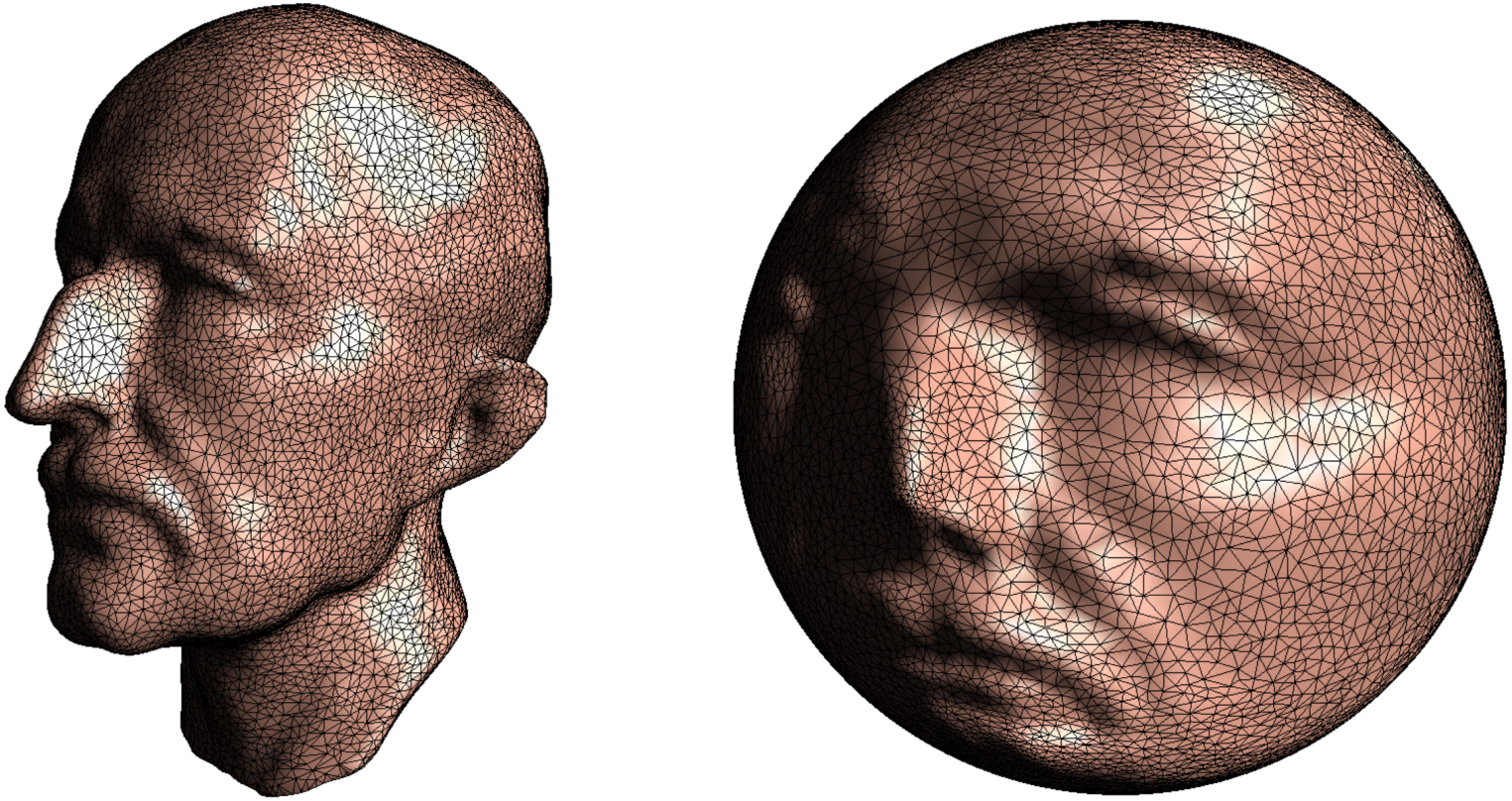


Figure: Input source and target meshes.

Computational Examples

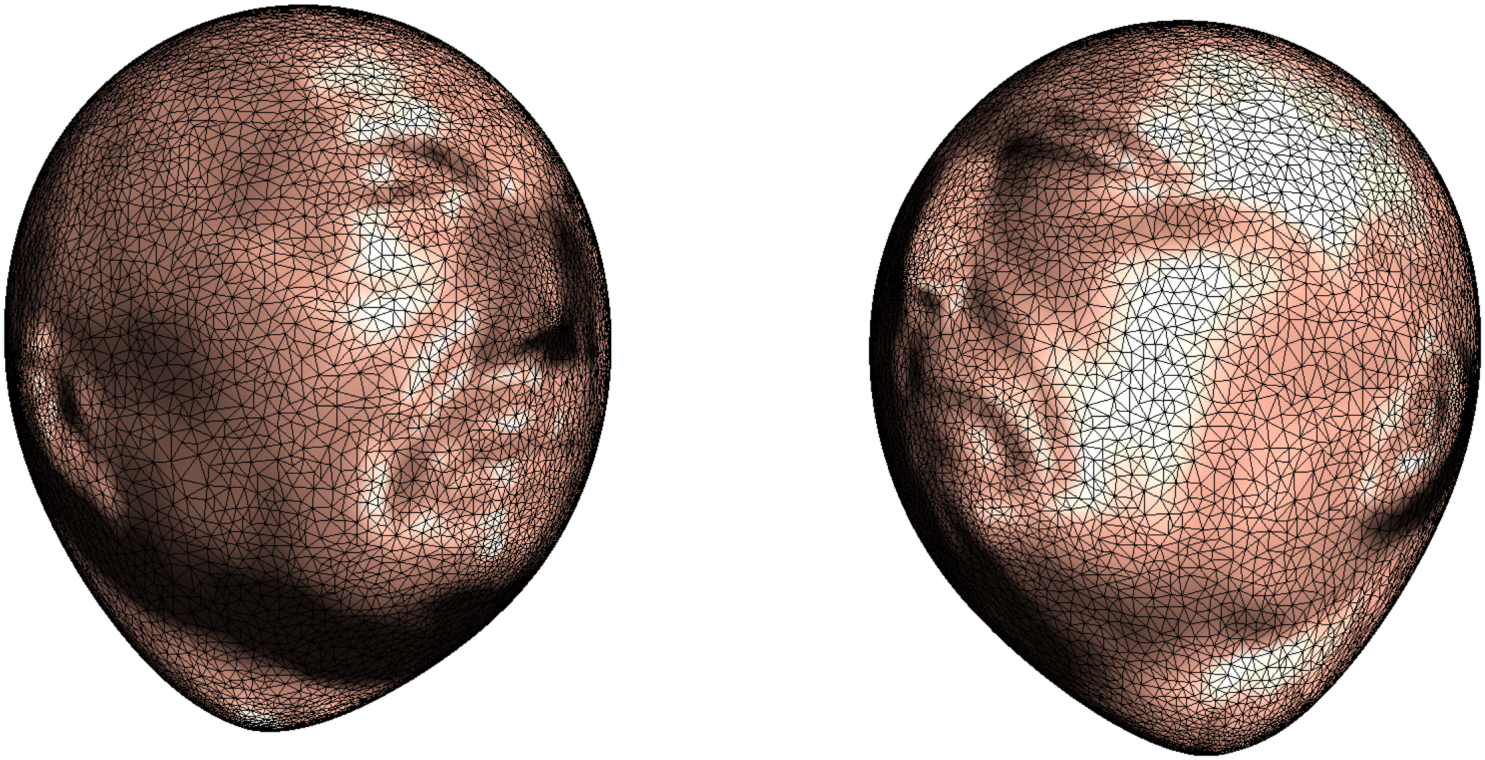


Figure: Final convex hull S_ρ.

Computational Examples

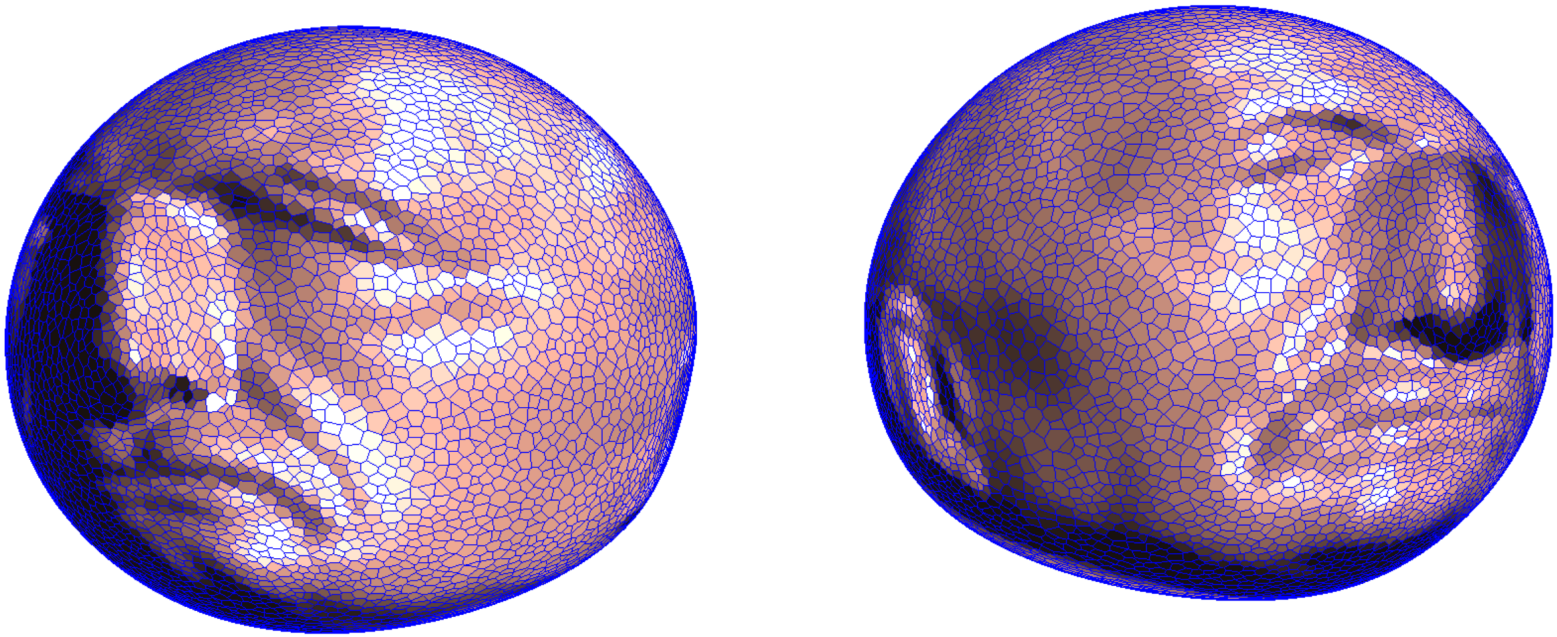


Figure: Final envelope S_{ρ^*}.

Computational Examples

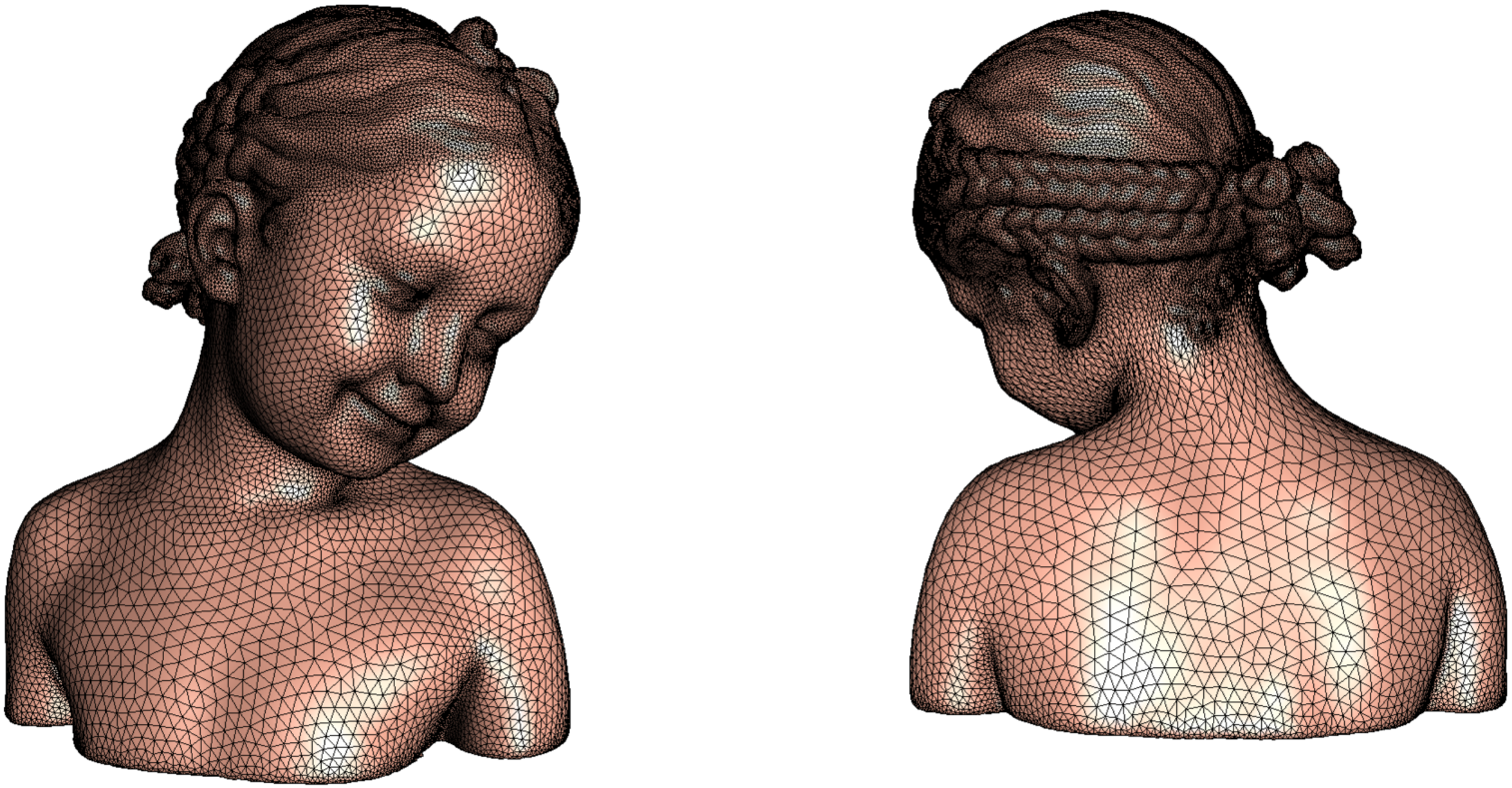


Figure: Input meshes.

Computational Examples

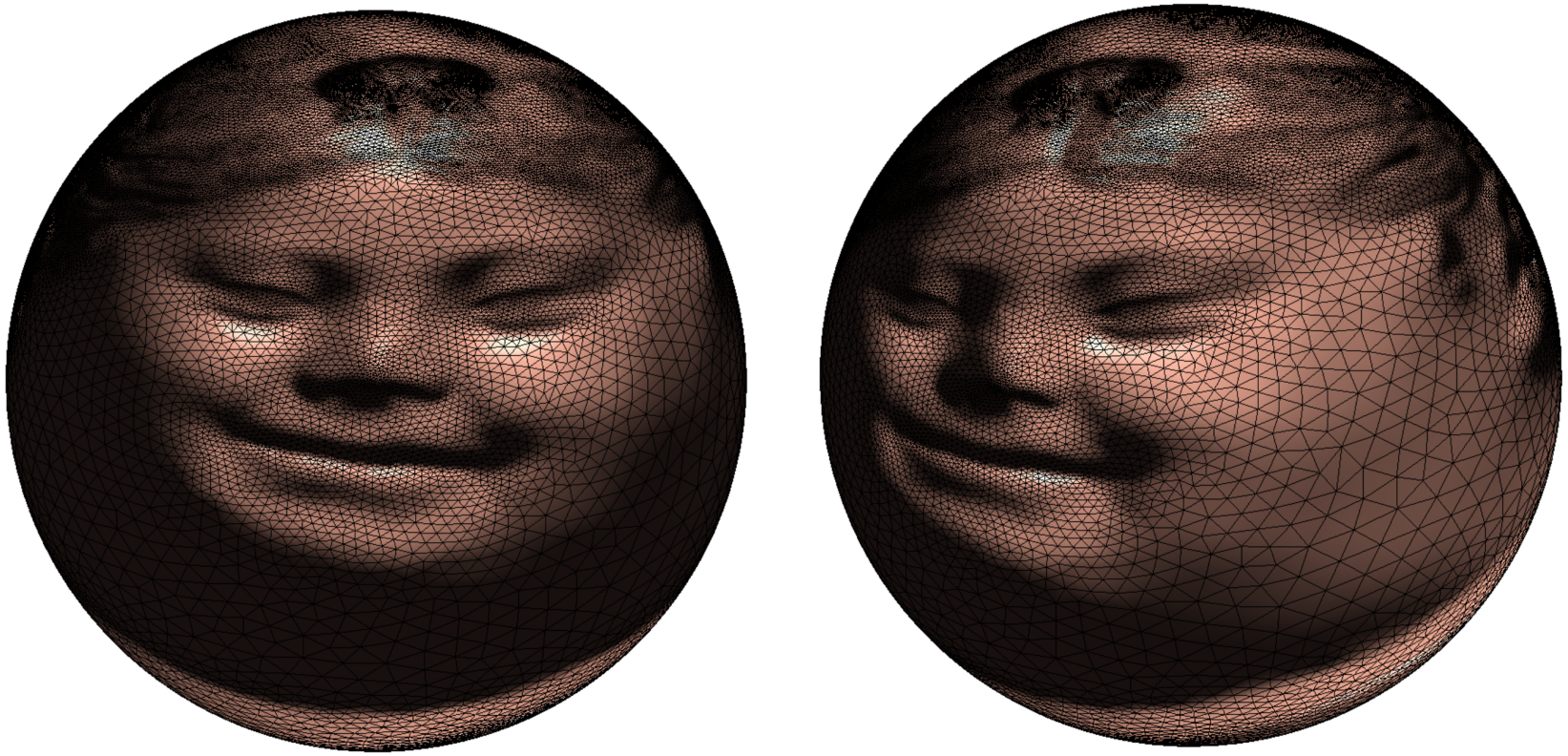


Figure: Initial harmonic maps.

Computational Examples

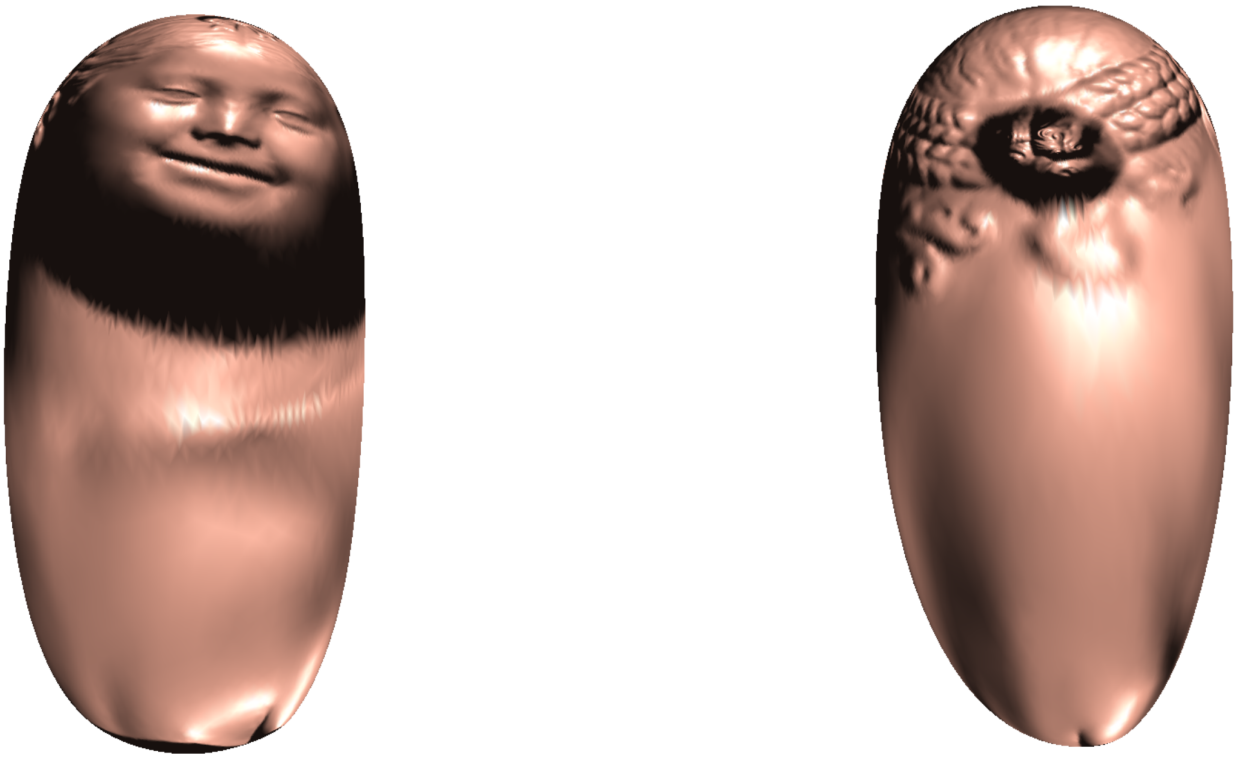


Figure: Final convex hull S_ρ.

Computational Examples

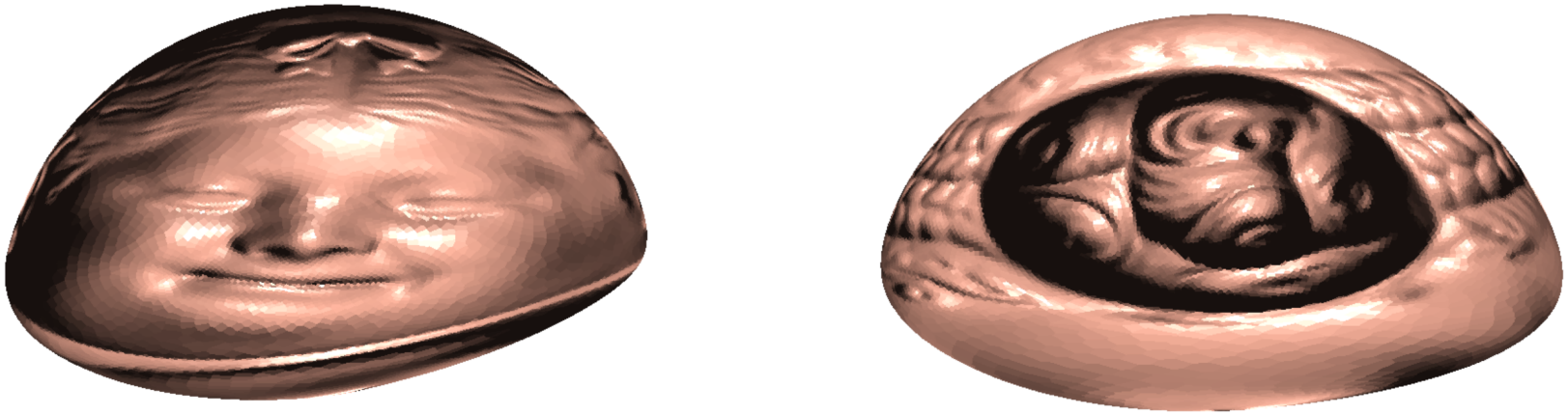


Figure: Final convex hull S_ρ.

Instruction

Dependencies

1. 'MeshLib', a general purpose mesh library based on Dart data structure.
2. 'Eigen', numerical solver.
3. 'freeglut', a free-software/open-source alternative to the OpenGL Utility Toolkit (GLUT) library.

Commands and Hot keys

- Command: -source mesh -target mesh.sphere.m
- '!': Newton's method
- 'L': Edit the lighting
- 'd': Show convex hull or upper envelope;
- 'g': Show original mesh, spherical image or the convex mesh;
- 'e': Show edges
- 'm': Compute the power cell centers;
- 'c': show the power cell centers;
- 'W': save to the output mesh;
- 'o': Take a snapshot
- '?': Help information

SphericalPowerDiagramDynamicMesh class

Compute the spherical Power Delaunay and Power Diagram.

1. *CPDMesh* :: *_Lawson_edge_swap* Lawson edge swap algorithm to compute the convex hull S_ρ;
2. *CPDMesh* :: *_Legendre_transform* Legendre dual transformation compute envelope S_{ρ^*}, spherical power voronoi diagram;
3. *CPDMesh* :: *_power_cell*$_a$*rea* Compute the power cell area;
4. *CPDMesh* :: *__edge_local_convex* verify whether the edge is local convex;
5. *CPDMesh* :: *__edge_flippable* verify whether the edge is flippable;
6. *CPDMesh* :: *_edge_weight* calculate the edge weight;

SphericalPowerDiagramDynamicMesh class

Compute the spherical Power Delaunay and Power Diagram.

1. *CPDMesh* :: *__edge_spherical_length*() Compute the spherical edge lengths γ_{ij};
2. *CPDMesh* :: *__face_power_center*() face power center o_f, and face power R_f;
3. *CPDMesh* :: *__halfedge_spherical_height*() the distance from the power center to the edge;
4. *CPDMesh* :: *__edge_interior_weight*() calculate the weight for each edge;

CSOTDynamicMesh class

Compute the spherical Optimal Mass Transportation Map.

1. *CSOTDynamicMesh* :: *_calculate_gradient* calculate the gradient of the Alexandrov energy;
2. *CSOTDynamicMesh* :: *_update_direction* compute the update direction, based on Newton's method;
3. *CSOTDynamicMesh* :: *_calculate_hessian* calculate the Hessian matrix of the Alexandrov energy;
4. *CSOTDynamicMesh* :: *_solve* solve the linear system;
5. *CSOTDynamicMesh* :: *_error* compute the relative and L^2 error;
6. *_OT_Damping*() damping algorithm;
7. *OT_Newton*() Newton's method;
8. *OT_Initialize*() set the target measure, the initial ρ_i's to be one.

Coding Assignment

Compute the Optimal Mass Transportation Map.

1. *CPDMesh* :: *__edge_local_convex* verify whether the edge is local convex;
2. *CPDMesh* :: *__edge_flippable* verify whether the edge is flippable;
3. *CPDMesh* :: *__edge_spherical_length*() Compute the spherical edge lengths γ_{ij};
4. *CPDMesh* :: *__face_power_center*() face power center o_f, and face power R_f;
5. *CPDMesh* :: *__halfedge_spherical_height*() the distance from the power center to the edge d_l;
6. *CPDMesh* :: *__edge_interior_weight*() calculate the weight for each edge w_{ij};

Directory Structure

- 3rdparty/MeshLib, header files for mesh;
- OT/include, OT/src, the source files for optimal transportation map;
- CMakeLists.txt, CMake configuration file;

Configuration

Before you start, read README.md carefully, then go three the following procedures, step by step.

1. Install [CMake](https://cmake.org/download/).
2. Download the source code of the C++ framework.
3. Configure and generate the project for Visual Studio.
4. Open the .sln using Visual Studio, and complie the solution.
5. Finish your code in your IDE.
6. Run the executable program.

Configure and generate the project

1. open a command window
2. cd ot-homework4_skeleton
3. mkdir build
4. cd build
5. cmake ..
6. open OTHomework.sln inside the build directory.