Geometric Algorithm for Optimal Transportation Map

David Gu

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

gu@cs.stonybrook.edu

November 5, 2020

Convex Geometric View

Monge Problem

Monge Problem

Given planar domains with probability measures (Ω, μ) and (Ω^*, ν), Ω is convex, total measures are equal $\mu(\Omega) = \nu(\Omega^*)$, the density functions are bounded $d\mu = f(x)dx$ and $d\nu = g(y)dy$, the transportation cost function is $c(x,y) = \frac{1}{2}|x-y|$, the Monge problem aims at finding the optimal transportation map,

$$\min_{T_\#\mu=\nu} \int_\Omega c(x,y)d\mu(x).$$

Brenier Theorem

Theorem (Brenier)

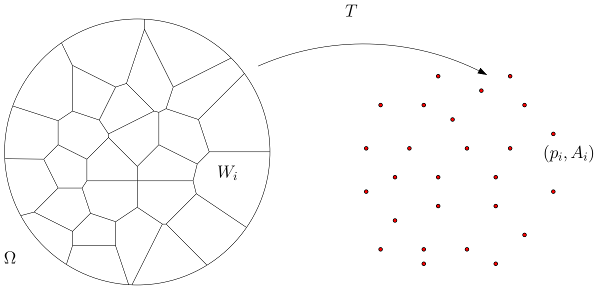
Given the above conditions, assume the density functions satisfy appropriate regularity conditions, $f, g \in L^1(\mathbb{R}^d$, Ω, Ω^* are compact, then the optimal transportation map exists and is unique, it is the gradient of a convex function $u : \Omega \to \mathbb{R}$, $T = \nabla u$, where u is the Brenier potential function.

The Brenier potential satisfies the Monge-Ampere equation,

$$\boxed{\det(D^2 u) = \frac{f(x)}{g \circ \nabla u(x)}}$$

with boundary condition $\nabla u(\Omega) = \Omega^*$.

Semi-Discrete Optimal Transportation Problem

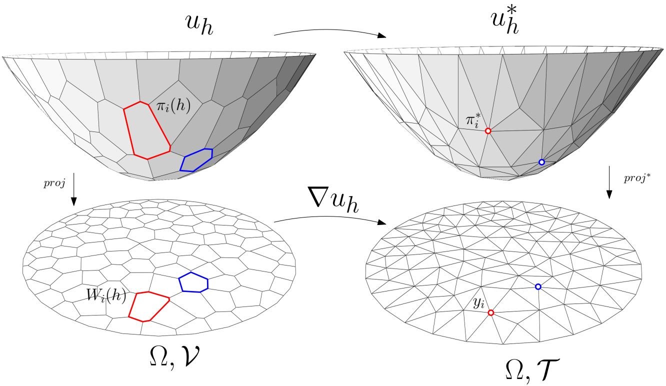


Problem (Semi-discrete OT)

Given a compact convex domain Ω in $\mathbb{R}^d$, and $y_1, y_2, \cdots, y_k$ and weights $\nu_1, \nu_2, \cdots, \nu_k > 0$, find a transport map $T : \Omega \to \{y_1, \ldots, y_k\}$, such that $vol(T^{-1}(p_i)) = \nu_i$, so that T minimizes the transportation cost:

$$\mathcal{C}(T) := \frac{1}{2} \int_\Omega |x - T(x)|^2 dx$$

Semi-Discrete Optimal Transportation Map

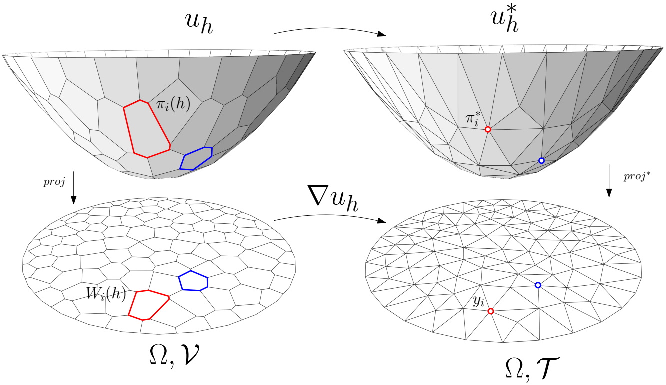


(Ω^*, ν) is discretized as $\{(y_i, \nu_i)\}_{i=1}^k$. Each sample y_i corresponds to a plane $\pi_i(x) = \langle x, y_i \rangle - h_i$, the Brenier potential is

$$u_h(x) := \max_{i=1}^{k} \{\langle x, y_i \rangle - h_i\},$$

where the height vector $h = (h_1, h_2, \cdots, h_k)$. u_h^* is the Legendre dual of

Semi-Discrete Optimal Transportation Map



u_h is the upper envelope of plane π_i's; u_h^* is the convex hull of points $\{(y_i, h_i)\}_{i=1}^k$; the projection of u_h^* is a power Delaunay triangulation of $\{y_i\}_{i=1}^k$; the projection of u_h is the dual power diagram of Ω.

Variational Proof

Theorem (Gu-Luo-Sun-Yau 2013)

Ω is a compact convex domain in $\mathbb{R}^n$, $y_1, \cdots, y_k$ distinct in $\mathbb{R}^n$, μ a positive continuous measure on Ω. For any $\nu_1, \cdots, \nu_k > 0$ with $\sum \nu_i = \mu(\Omega)$, there exists a vector $(h_1, \cdots, h_k)$ so that

$$u(\mathbf{x}) = \max\{\langle \mathbf{x}, \mathbf{p}_i \rangle + h_i\}$$

satisfies $\mu(W_i \cap \Omega) = \nu_i$, where $W_i = \{\mathbf{x} | \nabla f(\mathbf{x}) = \mathbf{p}_i\}$. Furthermore, $\mathbf{h}$ is the maximum point of the concave function

$$E(\mathbf{h}) = \sum_{i=1}^{k} \nu_i h_i - \int_{\mathbf{0}}^{\mathbf{h}} \sum_{i=1}^{k} w_i(\eta) d\eta_i,$$

where $w_i(\eta) = \mu(W_i(\eta) \cap \Omega)$ is the μ-volume of the cell.

The energy $E(\mathbf{h})$ is called the Alexandrov's energy.

Admissible Height Space

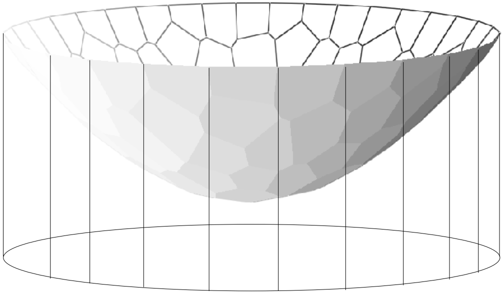
Definition (Admissible Height Space)

The admissible height space is defined as

$$\mathcal{H}(Y) := \left\{ \mathbf{h} \in \mathbb{R}^k : w_i(h) > 0, i = 1, 2, \dots, k \right\} \bigcap \left\{ \sum_{i=1}^{k} h_i = 1 \right\}.$$

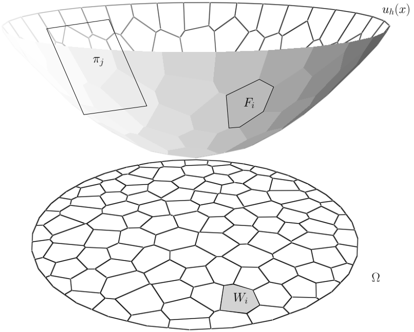
The admissible height space is a non-empty convex space. The optimization is to maximize the energy $E(\mathbf{h})$ in the admissible height space $\mathcal{H}$, using Newton's method.

Geometric Interpretation



One can define a cylinder through $\partial\Omega$, the cylinder is truncated by the xy-plane and the convex polyhedron. The energy term $\int^{\mathbf{h}} \sum w_i(\eta) d\eta_i$ equals to the volume of the truncated cylinder.

Computational Algorithm

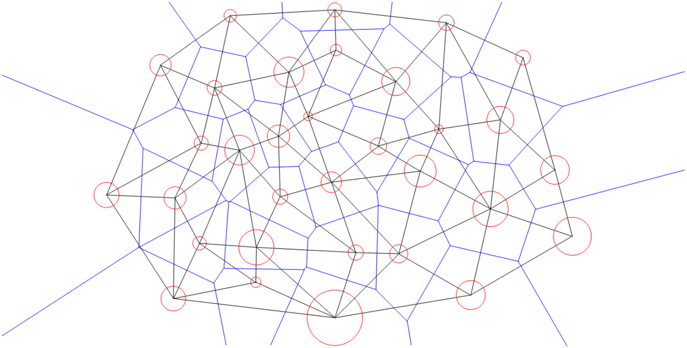


Definition (Alexandrov Potential)

The concave energy is

$$E(h_1, h_2, \cdots, h_k) = \sum_{i=1}^{k} \nu_i h_i - \int_{\mathbf{0}}^{\mathbf{h}} \sum_{j=1}^{k} w_j(\eta) d\eta_j,$$

Semi-Discrete Optimal Transportation Map

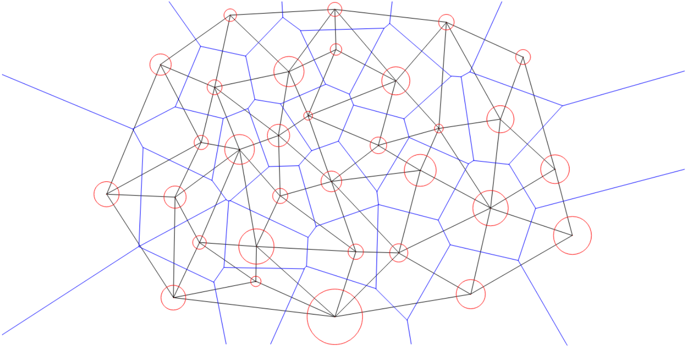


The gradient is $\nabla u_h = (\nu_i - w_i(\mathbf{h}))$; the element of the Hessian matrix is the ratio between the power voronoi edge length and the power Delaunay edge length,

$$a_{ij} = -\frac{1}{|y_i - y_j|} \int_{W_i \cap W_j} f(x) dx$$

and the diagonal element equals $a_{ii} = -\sum_{j \neq i} a_{ij}$.

Computational Algorithm



The Hessian of the energy is the length ratios of edge and dual edges,

$$\frac{\partial w_i}{\partial h_j} = -\frac{|e_{ij}|}{|\bar{e}_{ij}|}$$

Optimal Transport Map

Input: A set of distinct points $Y = \{y_1, y_2, \cdots, y_k\}$, and the weights $\{\nu_1, \nu_2, \cdots, \nu_k\}$;A convex domain Ω, $\sum \nu_j = \mathrm{Vol}(\Omega)$;
Output: The optimal transport map $T : \Omega \to Y$

1. Scale and translate Y, such that $Y \subset \Omega$;
2. Initialize $\mathbf{h}^0 \leftarrow \frac{1}{2}(|y_1|^2, |y_2|^2, \cdots, |y_k|^2)^T$;
3. Compute the Brenier potential $u(\mathbf{h}^k)$ (envelope of π_i's) and its Legendre dual $u^*(\mathbf{h}^k)$ (convex hull of π_i^*'s);
4. Project the Brenier potential and Legendre dual to obtain weighted Delaunay triangulation $\mathcal{T}(\mathbf{h}^k)$ and power diagram $\mathcal{D}(\mathbf{h}^k)$;

5. Compute the gradient of the energy

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

6. If $\|\nabla E(\mathbf{h}^k)\|$ is less than ε, then return $T = \nabla u(\mathbf{h}^k)$;
7. Compute the Hessian matrix of the energy

$$\frac{\partial w_i(\mathbf{h})}{\partial h_j} = -\frac{|e_{ij}|}{|\bar{e}_{ij}|}, \quad \frac{\partial w_i}{\partial h_i} = -\sum_j \frac{\partial w_i(\mathbf{h})}{\partial h_j}.$$

8. Solve linear system

$$\nabla E(\mathbf{h}) = \mathrm{Hess}(\mathbf{h}^k)\mathbf{d};$$

Optimal Transport Map

Damping Algorithm

9. Set the step length $\lambda \leftarrow 1$;
10. Construct the convex hull $\mathrm{Conv}(\mathbf{h}^k + \lambda \mathbf{d})$;
11. if there is any empty power cell, $\lambda \leftarrow \frac{1}{2}\lambda$, repeat step 3 and 4, until all power cells are non-empty;
12. set $\mathbf{h}^{k+1} \leftarrow \mathbf{h}^k + \lambda \mathbf{d}$;
13. Repeat step 9 through 12.

Optimal Transportation Map

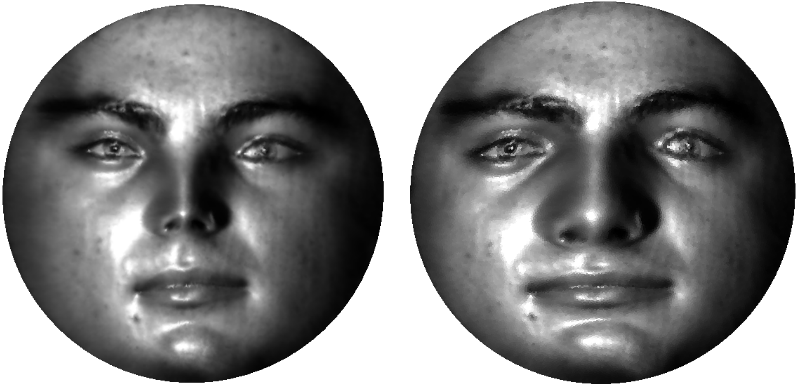


Figure: Optimal transportation map.

Optimal Transportation Map

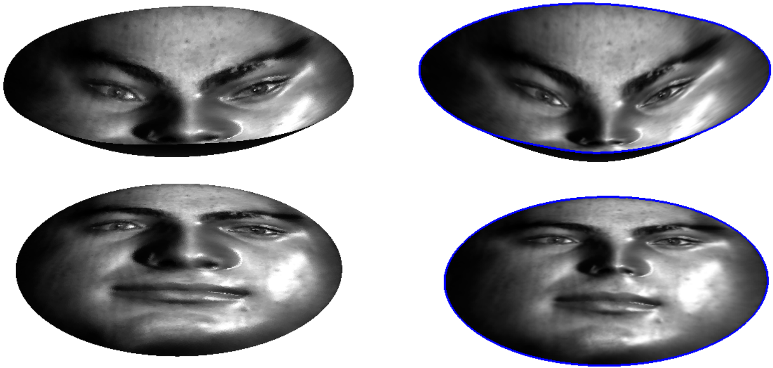


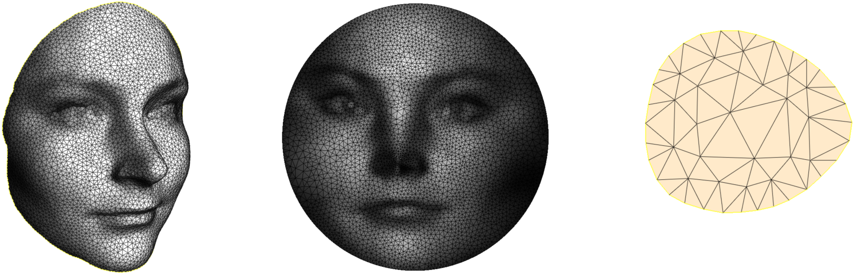
Figure: Optimal transportation map.

Computational Geometric Algorithms

File Format

- (Ω^*, ν) is represented as a triangle mesh (obj format), each vertex has both (x, y, z) coordinates and (u, v) parameters. Each vertex v_i represents a sample $y_i = (u_i, v_i)$, (u_i, v_i) specify the planar position in Ω^*. The summation of the areas of all triangular faces adjacent to v_i is treated as ν_i, (after normalization).
- (Ω, μ) is represented as another triangle mesh (obj format), its boundary gives the boundary of Ω. For current version, μ is the uniform distribution.

File IO



(a) Y and ν (b) planar positions $\{y_i\}$ (c) convex Ω

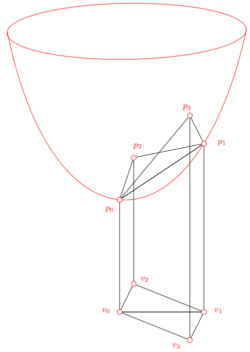
Figure: Input files.

Data Structure & Algorithms

1. The combinatorial data structure to represent the Delaunay triangulation and the dual voronoi diagram is either half-edge or Dart data structure;
2. The linear numerical solver is Eigen library;
3. The geometric computation is based on adaptive arithmetic method.
4. The power Delaunay is based on Lawson's edge flip algorithm.
5. The polygon clipping is based on Sutherland–Hodgman algorithm.
6. The optimization of Alexandrov energy is based on damping algorithm.

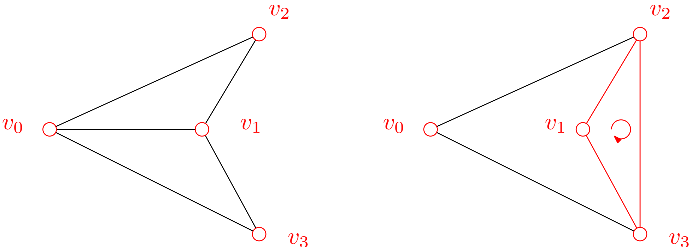
Edge Local Power Delaunay

Given an edge e in a planar triangulation $\mathcal{T}$, find the two neighboring faces, lift the four vertices to the convex hull φ, suppose vertex v_i is represented as $p_i(u_i, v_i, \varphi(u_i, v_i))$, compute the volume of the tetrahedron $[p_0, p_1, p_2, p_3]$. If the volume is positive, then e is locally powerd Delaunay, if the volume is negative, then e is non-locally-power-Delaunay.



Edge Flippable

Given an edge $e = [v_0, v_1]$ in a planar triangulation $\mathcal{T}$, if $[v_0, v_3, v_2]$ or $[v_1, v_2, v_3]$ is clockwise, then the edge is not flippable.



Lawson Edge Flip Algorithm

Input is a set of points S on the plane with the powers, the output is the power Delaunay triangulation.

1. Construct an arbitrary triangulation of the point set S;
2. Push all non-locally intrior edges of $\mathcal{T}$ on stack and mark them;
3. While the stack is non-empty do
   1. $e \leftarrow$ pop();
   2. unmark e;
   3. if e is locally power Delaunay 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 local power Delaunay, then there is some point p_i that is not on the convex hull of all p_k's.

Lawson Edge Flip for Convex Hull

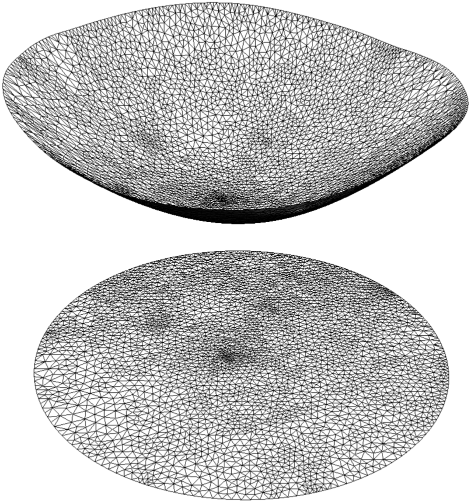


Figure: Construct convex hull of the graph of φ, using Lawson Edge Flip algorithm.

Legendre Dual

Given a convex hull, which is the graph of a convex function φ, we compute its Legendre dual φ^*. Each point $p_i = (a_i, b_i, c_i)$ on the convex hull represents a plane π_i,

$$\pi(x, y) = a_i x + b_i y - c_i.$$

Each face $[p_i, p_j, p_k]$ is dual to a point (x, y, z) satisfying the linear equation group,

$$\begin{pmatrix} c_i \\ c_j \\ c_k \end{pmatrix} = \begin{pmatrix} a_i & b_i & -1 \\ a_j & b_j & -1 \\ a_k & b_k & -1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Upper Envelope-Brenier Potential

Given the convex hull $\{p_1, p_2, \cdots, p_k\}$, where $p_i(u_i, v_i, \varphi(u_i, v_i))$, add one more point as infinity point $(0, 0, -h)$, h is big enough to be above all other points. Each face f_α is dual to a point f_α^*; each vertex v_i is dual to a supporting plane v_i^*.

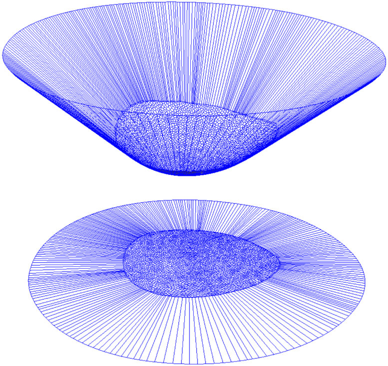
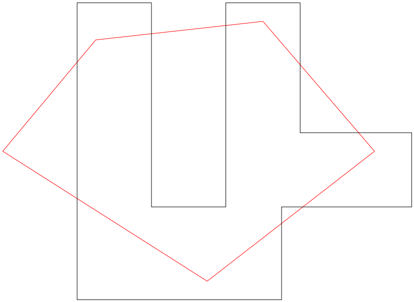


Figure: Legendre dual of the convex hull is the upper envelope.

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.



```
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
end
```

Upper Envelope - Brenier Potential

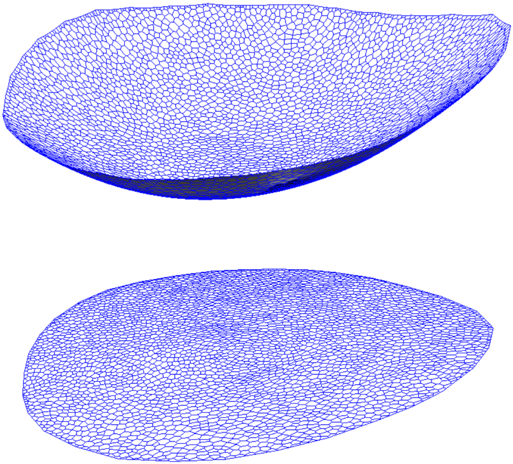


Figure: Brenier potential obtained by clipping the Legendre dual.

Cell Clipping

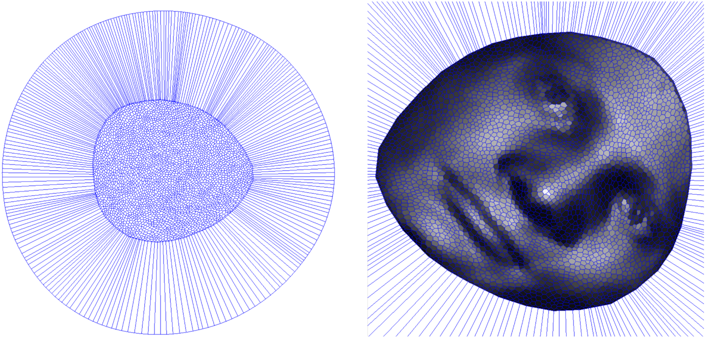


Figure: Boundary cell clipping.

Power Diagram Algorithm

1. Compute the convex hull using Lawson edge flipping, add the infinity vertex $(0,0,-h)$; project the convex hull to power Delaunay triangulation $\mathcal{T}$;
2. Compute the upper envelope using Legendre dualalgorithm and the, project to the power diagram $\mathcal{D}$;
3. Clip the power cells using Sutherland-Hodgman algorithm;

Damping Algorithm

1. Initialize the step length λ;
2. $\varphi \leftarrow \varphi + \lambda d$;
3. Compute the convex hull using Lawson edge flipping, add the infinity vertex $(0, 0, -h)$; project the convex hull to power Delaunay triangulation $\mathcal{T}$;
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. 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

1. Initialize ϕ as $\phi(u,v) = \frac{1}{2}(u^2 + v^2)$;
2. Call the 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 $\phi \leftarrow \phi + \lambda d$, such that ϕ is admissible;
7. Repeat step 2 through step 6, until the gradient is close to 0.

Transportation Map

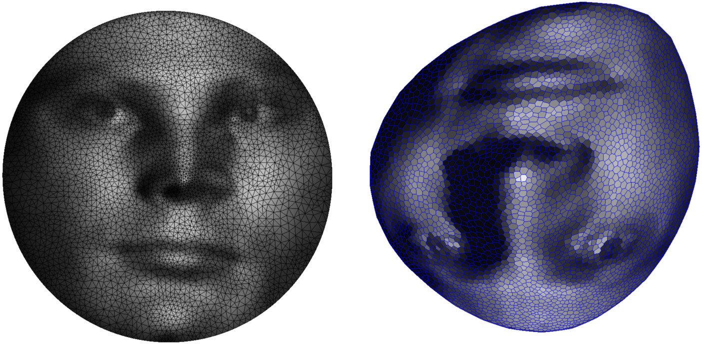


Figure: Transportation map.

Optimal Transportation Map

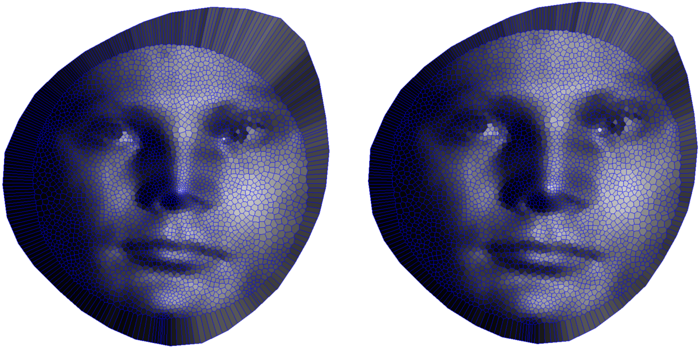


Figure: Optimal transportation map.

Optimal Transportation Map

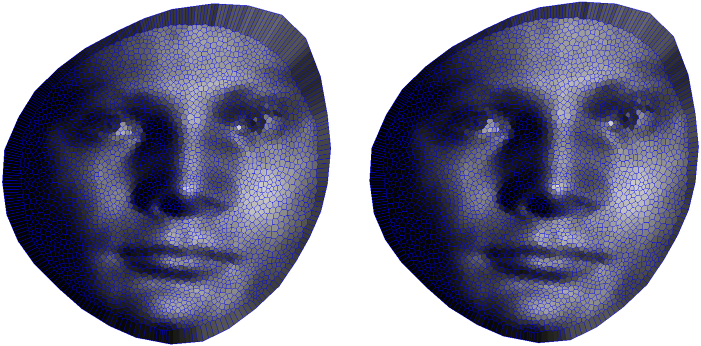


Figure: Optimal transportation map.

Optimal Transportation Map

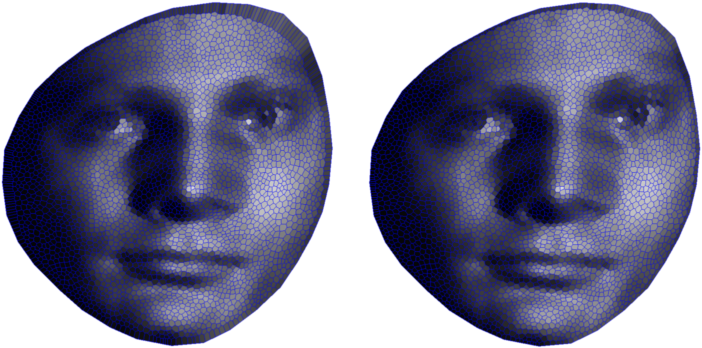


Figure: Optimal transportation map.

Optimal Transportation Map

Figure: Optimal transportation map.

Instruction

Dependencies

1. ‘DartLib‘ or ’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: -target target_mesh -source source_mesh
- '!': Newton's method
- 'm': Compute the mass center of power cells
- 'W': output the Legendre dual mesh and the optimal transportation map mesh
- 'L': Edit the lighting
- 'd': Show convex hull or upper envelope; power Delaunay or diagram
- 'g': Show 3D view or 2D view
- 'e': Show edges
- 'c': Show cell centers
- 'o': Take a snapshot
- '?': Help information

PowerDynamicMesh class

Compute the Power Delaunay and Power Diagram.

1. *CPDMesh* :: *_Lawson_edge_swap* Lawson edge swap algorithm to compute convex hull u_h^*, Power Delaunay triangulation;
2. *CPDMesh* :: *_Legendre_transform* Legendre dual transformation compute upper envelope u_h, Power voronoi diagram;
3. *CPDMesh* :: *_power_cell_clip* Clip power cells, based on Sutherland-Hodgman algorithm;

COMTDynamicMesh class

Compute the Optimal Mass Transportation Map.

1. *COMTMesh* :: *_update_direction* compute the update direction, based on Newton's method;
2. *COMTMesh* :: *_calculate_gradient* calculate the gradient of the Alexandrov energy;
3. *COMTMesh* :: *_calculate_hessian* calculate the Hessian matrix of the Alexandrov energy;
4. *COMTMesh* :: *_edge_weight* calculate the edge weight

Directory Structure

- 3rdparty/DartLib or 3rdparty/MeshLib, header files for mesh;
- MeshLib/algorithms/OMT, the header files for Power Diagram Mesh and Optimal Mass Transportation Map Mesh;
- 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-homework3_skeleton
3. mkdir build
4. cd build
5. cmake ..
6. open OTHomework.sln inside the build directory.