École Polytechnique Fédérale de Lausanne

Semester Project 2

Professor: Daniel Kuhn

Supervisor: Bahar Taskesen

2D/3D Geometry Morphing via L_2 Semi-discrete Optimal Transport

---

Author

Ju Wu

EPFL

Abstract

This project accomplishes numerical algorithms to compute the L_2 optimal transport map between the continuous measure μ and discrete one ν, where μ represents the density of the continuous distribution supported by a triangular/tetrahedral mesh (assigned values in each point), and where ν is a sum of Dirac masses. Since the main task is to do 2D/3D geometry morphing via semi-discrete optimal transport, it is convenient to comprise complex non-convex geometries with sets of convex polygons and polytopes and do multiple computational graphics operations on basic elements i.e., triangles and tetrahedra. The density of density-varying continuous distribution μ can be configured by assigning different weights on supporting points of meshes, and the area or volume represents measures for 2D or 3D geometries with constant density respectively.

The code will be released publicly and it is written down in Matlab since 1). It is powerful to process large-scale and sparse matrices, which are very common in our implementations, 2). Most of the element-based operations, which usually occupy most of computational resources, can be handled with built-in functions compiled as executable files already, which decreases the performance gap with pure CPP/C based codes. 3). It provides batch run and parallel computing options (in needs of modifying parts of codes accordingly). 4). We can interact with it and visualize instances easily in spite of compiler environments settings and operation system differences, which is not easy job for "geogram" library written by CPP. [1] 5). Last but not the least, there was no open-sourced 2D/3D geometry morphing codes based on semi-discrete optimal transport method (OTM) written in Matlab currently.

The computation of 2D/3D power diagram and regular triangulation constrained by meshes, which are the fundamentals for the following works, is discussed and accomplished. The Barzilai-Borwein (BB) method gradient descent and Newton method are developed to optimize the dual variable of the Dirac mass in the transport functions i.e., the weights of power diagrams.

The numerical algorithm is experimented and evaluated on several instances, with up to hundreds of triangles in 2D cases, hundreds of tetrahedra in 3D case and thousands of Dirac masses. And then multiple techniques including the parallel computing toolbox in Matlab are used to speed up algorithms. Finally, the limitation of the code is analyzed by profiler and expects improvements regarding the costs of time and memory in the pinpointed functions in the future.

Table of contents

1 Introduction

Collision-free and mass conservative geometry morphing has wide applications on swarm path planning, computational graphics and so on. Generally, the geometries in reality are non-convex, and they usually have uneven smoothness on their boundaries and non-constant density distributions. Thus, in practice the 3-dimensional (3D) geometries are represented as meshes comprised of varying regular polytopes e.g., tetrahedron, pyramid, triangular prism and hexahedron shown in Fig. (1). The local density distribution within each cell can be parameterized by values assigned to each vertex. We are mainly concerned about mesh element types of triangles for 2D geometries and tetrahedra for 3D geometries, and the density distribution is considered as a piecewise linear function.

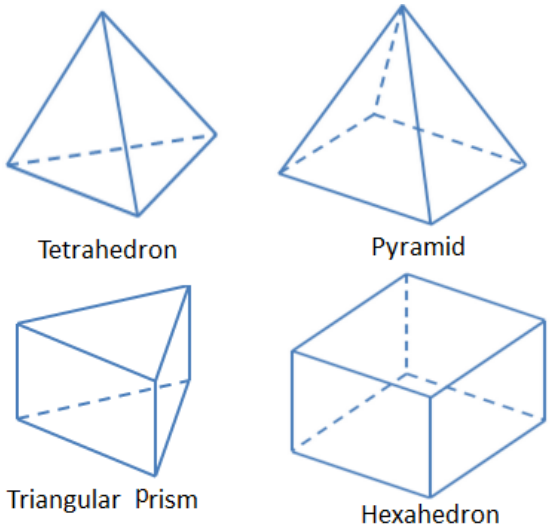


Figure 1: Different types of mesh elements

To perform the geometry morphing, we need to figure out the correspondence relationship between two distributions i.e., the transport map. We use L_2 metrics i.e., $c(x, y) = 1/2||x - y||^2$ to compute the optimal transport function, where (x, y) represents a pair of points in two distributions. We can also derive the relation between the power diagrams and the weights of the semi-discrete optimal transport functions. Then the optimal weights of the transport functions are computed by minimizing a convex objective function with varying methods. And at each iteration the intersection between a power diagram and the triangular/tetrahedral meshes that defines the measure μ is computed to evaluate the value and gradient of this objective function.

2 L_2 semi-discrete optimal transport

The initial formation of the optimal transport was proposed by Monge as follows:

$$\begin{aligned} \min \quad & \int_{\Omega} c(x, T(x))\, d\mu \\ \text{s.t.} \quad & \nu = T_{\#\mu} \end{aligned} \tag{1}$$

where Ω is a Borel set and its two measures μ and ν satisfy that $\mu(\Omega) = v(\Omega)$. $T_{\#\mu}$ is the optimal transport that pushes forward μ to ν and it is defined by $T_{\#\mu}(\omega) = \mu(T^{-1}(\omega))$ for any measurable subset ω of Ω. c is the convex distance function. Note that there only exists option to merge the mass of the source distribution but not to split them so that the relaxation of Monge's problem i.e., Kantorovich formation is introduced, in which the mass of source distribution can be both splitted and merged. The optimal transport mapping of μ and ν measures on $\Omega \times \Omega$ is as follows.

$$\begin{aligned} \min \quad & \int_{\Omega\times\Omega} c(x,y)d\gamma, \gamma \in \Pi(\mu,\nu) \\ \text{s.t.} \quad & \Pi(\mu,\nu) = \{\gamma \in P(\Omega\times\Omega)|(P_1)_{\#\gamma} = \mu, (P_2)_{\#\gamma} = \nu\} \end{aligned} \tag{2}$$

where $(P_1)_{\#\gamma}, (P_2)_{\#\gamma}$ represent the two projections such that $(x,y) \in \Omega\times\Omega \mapsto x$ and $(x,y) \in \Omega\times\Omega \mapsto y$ respectively. And $(Id \times T)_{\#\mu}$ corresponds to the optimal transport map.

Then for the transport plan $\gamma = (Id \times T)_{\#\mu} \in \Pi(\mu, v)$ we can obtain the Monge–Kantorovich equivalence from the Brenier theorem in [2] as follows,

$$\min \int_{\Omega\times\Omega} c(x,y)d(Id\times T)_{\#\mu} = \min \int_{\Omega} c(x, T(x))\, d\mu \tag{3}$$

Then we obtain the duality of the above formation as

$$\begin{aligned} \max \quad & \int_{\Omega} \psi d\mu + \int_{\Omega} \phi d\nu \\ \text{s.t.} \quad & \phi(x) + \psi(y) \le c(x,y), \forall (x,y) \in \Omega\times\Omega \end{aligned} \tag{4}$$

where ϕ and ψ are admissible dual potentials, from the Brenier theorem in [2], with $c(x,y) = 1/2\|x-y\|^2$, we have $\forall(x,y) \in \partial_c\psi, \nabla\psi(x) + y - x = 0$, we have the optimal transport $y = T(x) = x - \nabla\psi(x) = \nabla(\|x\|^2/2 - \psi(x))$.

We have the following observations:

1. The dual potential ψ in Eq. (4) is c-concave and can be replaced by $\phi^c(x) = \inf_y c(x,y) - \phi(y)$.
2. Define $\Phi(x) = \|x\|^2/2 - \psi(x)$, Φ is convex.
3. For two source points x_1 and x_2, if $x_1 \neq x_2$ and $0 < t < 1$, two particles at the trajectories $t_1 = (1-t)x_1 + tT(x_1)$ and $t_2 = (1-t)x_2 + tT(x_2)$ cannot collide at the same time t.

If we consider that μ is continuous distribution and $\nu = \sum_{i=1}^{k} \nu_i\delta_{p_i}$ is a sum of k Dirac masses, and they conform to the mass conservation such that $v(\Omega) = \sum_{i=1}^{k} v_i = \mu(\Omega)$. We can parameterize the Eq. (4) by letting $\phi(y_i) = 1/2\omega_i$, so that combined with Observation. 1, we have $\psi(x) = 1/2(\inf_{y_i} ||x - y_i||^2 - \omega_i)$. Thus Eq. (4) can be written as

$$\max \int_{\Omega} 1/2(\inf_{y_i} ||x-y_i||^2 - \omega_i)d\mu + \int_{\Omega} 1/2\omega d\nu \tag{5}$$

Considering that ν is the discrete distribution, by putting inf to the outside of the integral and eliminating 1/2, we have,

$$\max \sum_i \int_{\Omega_{y_i}} \|x - y_i\|^2 - w_i d\mu + \sum_i v_i w_i \tag{6}$$

where Ω_{y_i} is one of the subsets of the power diagram $Pow_W(Y)$ that partitions Ω and defined as

$$Pow_W(y_i) := \{x | \|x - y_i\| - w_i < \|x - y_j\| - w_j, \forall i \neq j\} \tag{7}$$

from Eq. (7), the Voronoi diagram is the special case of power diagram in which $\forall i, w_i = 0$ and defined as

$$Vor(y_i) := \{x | \|x - y_i\| < \|x - y_j\|, \forall i \neq j\} \tag{8}$$

We define the following function derived from the first term of Eq. (6)

$$f_T(W) = \int_\Omega \|x - T(x)\|^2 - w_{T(x)} d\mu. \tag{9}$$

where the transport function $T(x)$ is determined by the weights vector W in power diagram in Eq. (7) for the fixed discrete points y_i in ν and $w_{T(x)}$ is the fixed weight w_i in each corresponding power diagram cell determined by x. For the fixed weights W, we have,

$$f_T(W) = \int_\Omega \|x - T(x)\|^2 d\mu - \sum_{i=1}^{k} w_i \mu(T^{-1}(y_i)). \tag{10}$$

where $T^{-1}(y_i)$ denotes the pre-image of T and $\mu(T^{-1}(y_i))$ represents the measure of the region covered by a subset of power diagrams supported by y_i, by envelope theorem, the variation of T incurred by W can be ignored to obtain the gradient of $f_T(W)$ as

$$\frac{\partial f_T(W)}{\partial w_i} = -\mu(T^{-1}(y_i)) = -\mu(Pow_W(y_i)) \tag{11}$$

Combined with Eq. (7) and (10), the global maximum optimization loss function can be derived from Eq. (4) as follows

$$g(W) = f_T(W) + \sum_{i=1}^{k} \omega_i \nu_i \tag{12}$$

And its gradient is given by

$$\frac{\partial g}{\partial w_i} = -\mu(Pow_W(y_i)) + \nu_i \tag{13}$$

Note that g is comprised of a concave function and a linear one, g has an unique global maximum when the gradient is zero, i.e., the measure $\mu(Pow_W(y_i))$ is equal to the prescribed measure v_i for each power diagram cell. Finally we obtain the following theorem, and its proof is the above

derivations.

Theorem 2.1 *Given a measure μ with density, a set of discrete points y_i and prescribed masses ν_i such that $\sum_i \nu_i = \mu(\Omega)$, there exists a weight vector W such that $\mu(Pow_W(p_i)) = \nu_i$ and the T_W defined by the power diagram parameterized by W is an optimal transport map.*

3 Power diagram construction

In this semi-discrete setting of L_2 cases, the optimal transport map is determined by the weights of a power diagram. It is essential to construct the power diagrams in 2D/3D cases, which are the foundations for our implementations.

3.1 Computational geometry backgrounds

The Delaunay triangulation is the geometric dual of Voronoi diagram. Given a set of weighted points S in 2D/3D cases, its power diagram $PD(S)$ is a tessellation of the space and the regular triangulation $RT(S)$ is the geometric dual of power diagram. They have the following relationships applied for 2D/3D cases [3]:

1. A vertex y_i of $RT(S)$ is the the dual of power diagram cell $Pow_W(y_i)$, and y_i is the support of that. y_i is redundant in $RT(S)$ if and only if the power diagram cell $Pow_W(y_i)$ is an empty set.
2. An edge of $RT(S)$ is the dual of a face of $PD(S)$ and the edge is perpendicular to the face. Two vertices $y_i, y_j \in S$ are connected by an edge in $RT(S)$ if and only if the power diagram cell $Pow_W(y_i)$ and the power diagram cell $Pow_W(y_j)$ share a face.
3. A face of $RT(S)$ is the dual of an edge of $PD(S)$. The face is perpendicular to the edge and the edge goes through the orthogonal center of the face.
4. A triangle/tetrahedron of $RT(S)$ is the dual of a vertex of $PD(S)$. The vertex lies in the orthogonal center of the triangle/tetrahedron.

There is an useful relation between regular triangulation in d-dimensional space and convex hulls in $(d+1)$-dimensional space introduced in [4] as follows

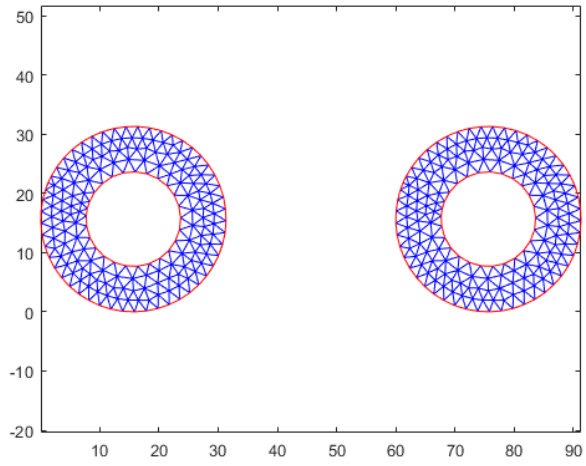
Lemma 3.1 *Let S be a finite set of weighted points in 3D. The vertical projection of the lower facets of the convex hull $CH(S^+)$ into 3D gives tetrahedra of $RT(S)$.*

Where S^+ represents the lifted point set of S. The above lemma takes 3D cases for instance, and for a point $p = (p_x, p_y, p_z) \in R^3$ with weight ω_p in 3D space, its lifted point is defined as

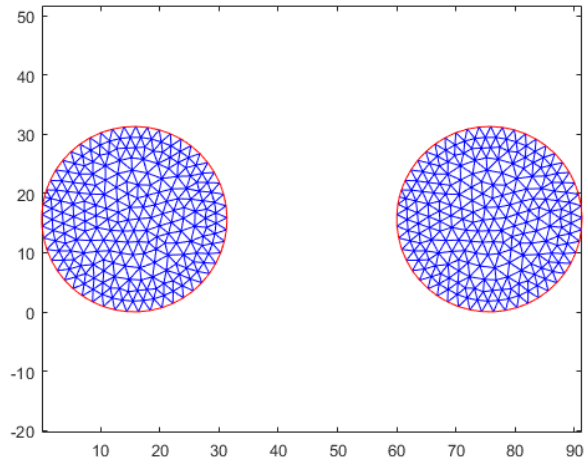
$$p^+ = (p_x, p_y, p_z, p_x^2 + p_y^2 + p_z^2\text{-}\omega_p). \tag{14}$$

To demonstrate the relation between S^+ and S derived from Lemma 3.1, first we take two instances of 2D meshes as follows. Both of them have the supporting discrete points sets as the vertices of

triangles.

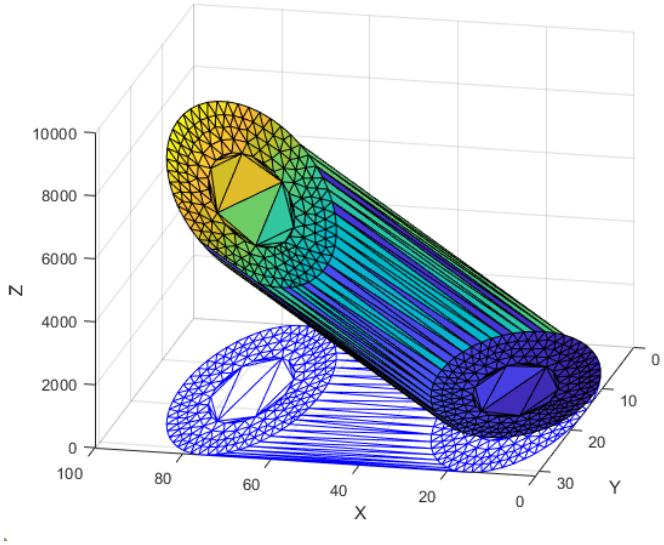


2D mesh of the "2-rings"

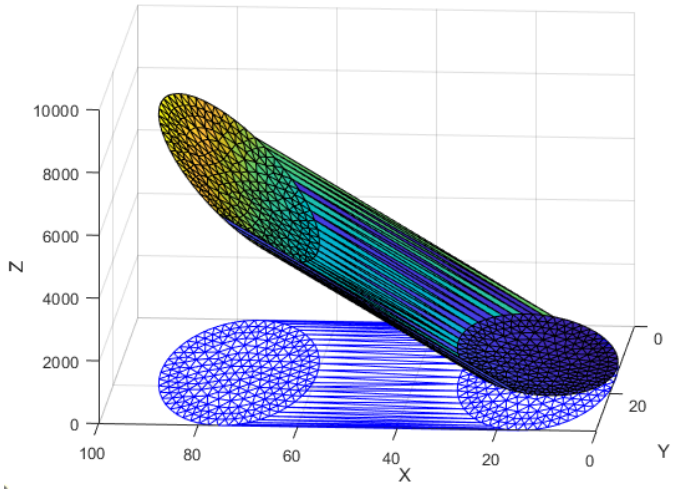


2D mesh of the "2-circles"

Then we assign the weights w_p of all points in the set as zero. We obtain the lower facets of the convex hull $CH(S^+)$ and its projections into X-Y plane as $RT(S)$.



$CH(S^+)$ and $RT(S)$ for set S supporting the "2-rings"



$CH(S^+)$ and $RT(S)$ for set S supporting the "2-circles"

3.2 Numerical implementations

The essential and elemental steps for computing power diagram and its dual regular triangulation is to compute the convex hulls. The Qhull library is used in Matlab to compute and construct the convex hulls by implementing the Quickhull algorithms. The Quickhull is a method of computing the convex hull of a finite set of points in the plane. It uses a divide and conquer approach similar to that of QuickSort and its average case time complexity is considered to be $O(n\log(n))$, whereas in the worst case it takes $O(n^2)$. Given a finite set of the weighted points S, the numerical algorithm to obtain $RT(S)$ and $PD(S)$ is as follows.

Algorithm 1: To compute $RT(S)$ and $PD(S)$ given a finite set of the weighted points S

Data: A finite set of the weighted points S, and the weight w_i assigned to each point s_i

Result: The regular triangulation $TR(S)$, the power diagram $PD(S)$ and the vertices of $PD(S)$.

Initialization:

$S\_lifted \leftarrow$ `Compute_LiftedPoints(`$S,\ W$`)`;

$ConvexHull(S\_lifted) \leftarrow$ `Compute_ConvexHull(`S_lifted`)`;

Compute $RT(S)$:

$[Facets,\ S\_lifted] \leftarrow ConvexHull(S\_lifted)$;

$Facets\_normals \leftarrow$ `Compute_Normals(`$Facets,\ S\_lifted$`)`;

$Normals\_flipped \leftarrow$ `Flip_Normals(`$Facets\_normals,\ S\_lifted$`)`;

$LowerFacets \leftarrow$ `Filter_LowerFacets(`$Normals\_flipped,\ Facets$`)`;

$RT(S) \leftarrow [LowerFacets,\ S]$;

Compute $PD(S)$:

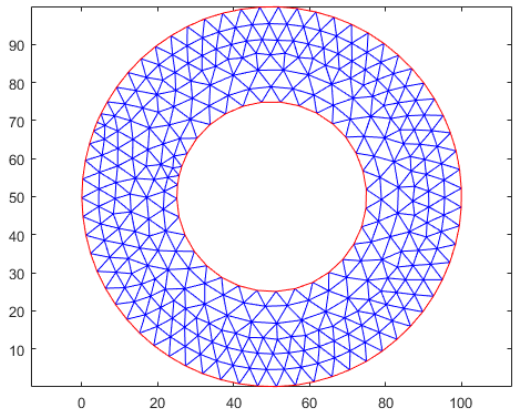
$PD\_pieces \leftarrow$ `PowerDiagram_Pieces(`$RT(S)$`)`;

$PD\_centers \leftarrow$ `PowerDiagram_CellCenters(`$RT(S),\ S,\ W$`)`;

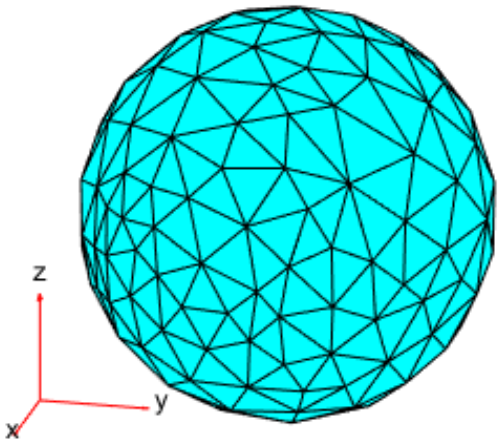
$PD\_bounds \leftarrow$ `Find_Boundaries(`$RT(S),\ PD\_pieces$`)`;

$[PD(S),\ vertices\_PD(S)] \leftarrow$ `Assemble_Subsets(`$RT(S),\ PD\_pieces,\ PD\_centers,\ PD\_bounds$`)`;

To demonstrate the above algorithm to generate the power diagrams from point sets, first we take one 2D mesh comprised of triangles and one 3D mesh comprised of tetrahedra as follows

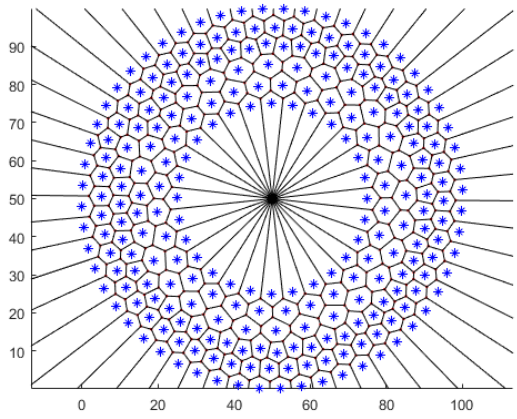


2D mesh of the "ring"

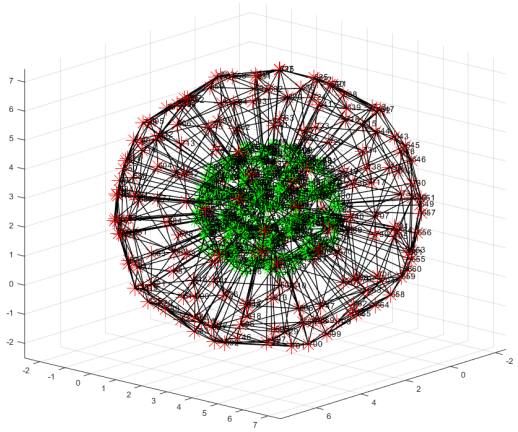


3D mesh of the "sphere"

Then we compute their power diagrams. The blue points are the point set supporting "ring". The green points are the point set supporting "sphere" and the red points are the bounds points computed from the boundary facets of $RT(S)$.



Power diagram of 2D points supporting the "ring"



Power diagram of 3D points supporting the "sphere"

4 Numerical algorithms

In this section, we developed the algorithms to optimize the global loss function in Eq. (12). To compute the gradient in Eq. (13), we need to compute the intersection of the mesh M represented as interconnected 2/3-dimensional simplex and each subset of the power diagram, and this function will take more than 70 percentage of the total running time of the program in the general cases so that it would be focused on later when managing to improve the performance later. After intersection, we obtain the power diagram constrained by the target mesh. The algorithm to compute $Pow_W(Y) \cap M$ is as follows

Algorithm 2: computing $Pow_W(Y) \cap M$ and $\mu(Pow_W(y_i)|M)$

Data: A triangular/tetrahedral mesh M with varying/constant densities, a set of points Y and weights W.

Result: The constrained power diagram $Pow_W(Y)|M$ as the intersection of $Pow_W(Y) \cap M$, and the volume of each of them $\mu(Pow_W(y_i)|M)$

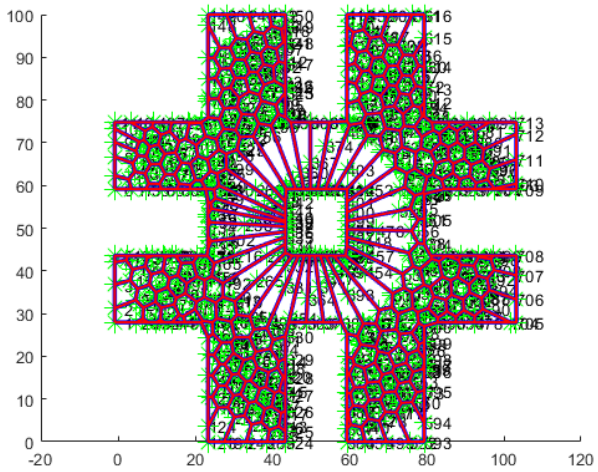
Compute $Pow_W(Y)$ with Algo. 1;

foreach *subset* $Pow_W(y_i)$ *in power diagram* $Pow_W(Y)$ **do**

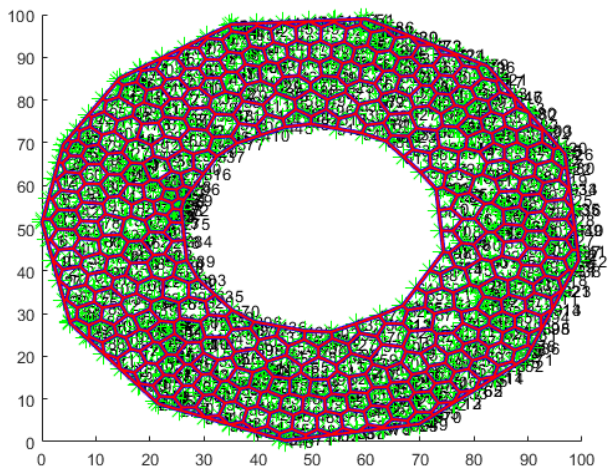
- $kl \leftarrow 0$;
- $Preallocated\_list_i \leftarrow 0$;
- $Cell\_Contri \leftarrow 0$;
- **foreach** *triangle/tetrahedron* $t_j \in M$ **do**
  - $Intersected\_Polyhedron \leftarrow Pow_W(y_i) \cap t_j$;
  - $Cell\_Contri \leftarrow Cell\_Contri + volume(Intersected\_Polyhedron)$;
  - $Min\_V\_Rep \leftarrow extreme(Intersected\_Polyhedron)$;
  - $Preallocated\_list_i(kl+1 : kl+Min\_V\_Rep, :) \leftarrow Min\_V\_Rep$;
- $Pow_W(y_i)|M \leftarrow Unique(Preallocated\_list_i(1 : kl, :))$;
- $\mu(Pow_W(y_i)|M) \leftarrow Cell\_Contri$;

Since each intersection operation is separate from each other, the parallel computing mode that divides the whole task, fully utilizes the CPU resources and overclock core frequencies can be used to speed up the above algorithm. In our setting, $\mu(Pow_W(y_i)|M)$ corresponds to the integral of the piecewise linear density on the intersection between $Pow_W(y_i)$ and the triangle/tetrahedron t_j of M.

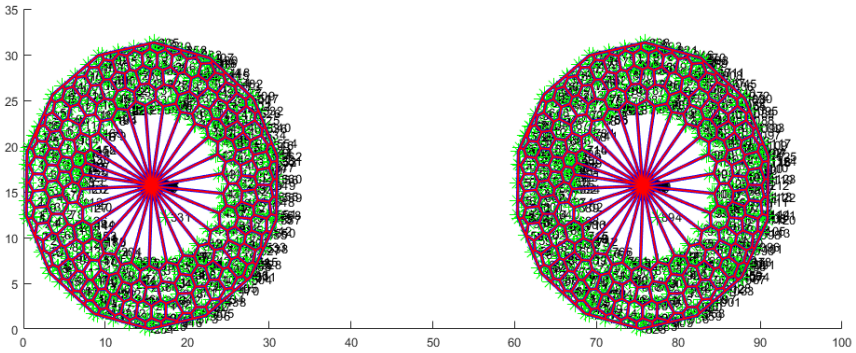
Note that it is notoriously computationally heavy to compute the intersection of the polyhedron, i.e., convex hulls, it is usually comprised of two steps when both the input and output polytopes are of V-representations: 1). generate minimal H-representations of the two polytopes, 2). solve the vertex enumeration problem for the union of the k H-representations. It will be discussed in details in Section. 5. Given the above two algorithms, we take the instances of the 2D power diagrams constrained by "4 cross", "rectangle", "ring" and "2 circles" and of the 3D power diagrams constrained by "cuboid" as follows.



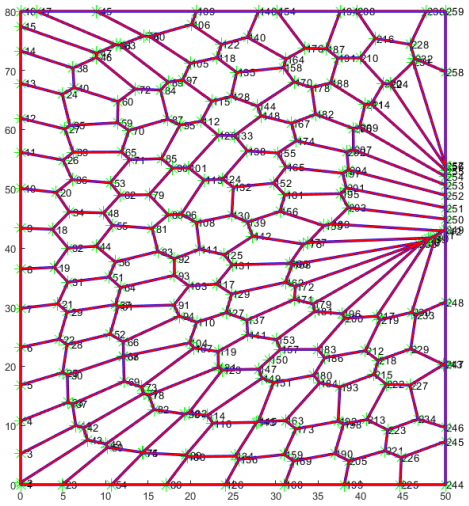
PD constrained by "4 cross"



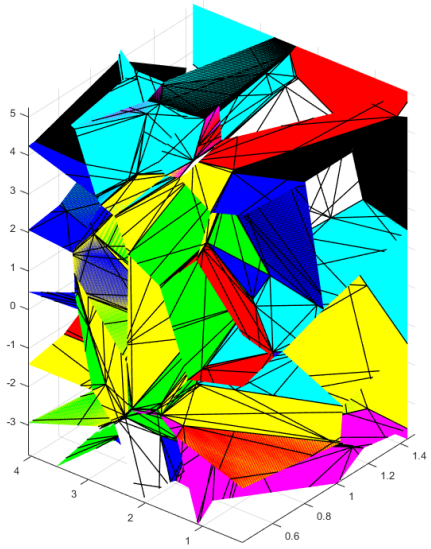
PD constrained by "ring"



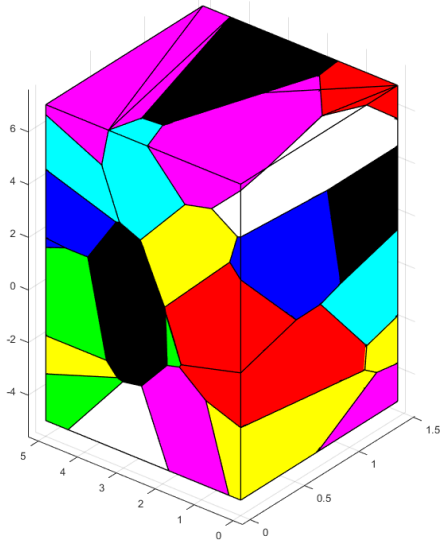
PD constrained by "2 circles"



PD constrained by "rectangle"



PD constrained by "cuboid" (interior)



PD constrained by "cuboid" (exterior)

Now that it is much more complex to obtain the Hessian matrix of Eq. (12) if we take Newton's method to optimize Eq. (12), so in kth iteration we can consider use $\alpha_k \nabla g^{(k)}$ to approximate $(H^{(k)})^{-1}\nabla g^{(k)}$ without computing $H^{(k)}$, which is called Barzilai-Borwein (BB) method, where α_k is a scalar value. This method is proposed in [5], by solving least-squares problems, we get two step size α_k as follows

$$\alpha_k^1 = \frac{(s^{(k-1)})^T s^{(k-1)}}{s^{(k-1)})^T y^{(k-1)}} \tag{15}$$

$$\alpha_k^2 = \frac{(s^{(k-1)})^T y^{(k-1)}}{y^{(k-1)})^T y^{(k-1)}} \tag{16}$$

where $s^{(k-1)} := W^{(k)} - W^{(k-1)}$ and $y^{(k-1)} := \nabla g^{(k)} - \nabla g^{(k-1)}$. Since $W^{(k-1)}$ and $\nabla g^{(k-1)}$ and thus $s^{(k-1)}$ and $y^{(k-1)}$ are unavailable at $k = 1$, we apply the standard gradient descent at $k = 1$ with the

step size $\alpha^{(1)} = 0.867$ and start BB from $k = 2$. In our setting, we fix $\alpha_k = \alpha_k^1$ and $\alpha_k = \alpha_k^2$ for a few consecutive steps T_s alternatively to alleviate oscillating across a valley and falling in local minimum regions.

Note that the Newton's method introduces the knowledge of the second derivative of the loss function and converges quadratically in general. Given $c(x, y) = 1/2||x - y||^2$, it is necessary to construct large-scale Hessian matrices with the following equations. [6]

$$\frac{\partial^2 g}{\partial w_i \partial w_j} = \int_{Pow_W(y_i)|M \cap Pow_W(y_j)|M} \frac{1}{||y_i - y_j||} d\mu \tag{17}$$

$$\frac{\partial^2 g}{\partial {w_i}^2} = -\sum_{k \neq i} \frac{\partial^2 g}{\partial w_i \partial w_k} \tag{18}$$

where $Pow_W(y_i)|M$ is one of the subsets of the power diagrams constrained by the target mesh M. Since Eq. (12) is concave, its Hessian matrix is semi-negative definite matrix, so we add an empirical term δI to H. Finally we get $H_n = H + \delta I$, where δ is a very small term, we take -0.665 at default.

With above discussions, we have the following algorithm to compute the semi-optimal transport between a finite set of points Y and mesh M with the density distribution ρ.

Algorithm 3: Semi-discrete optimal transport between a finite set of points Y and mesh M

Data: A triangular/tetrahedral mesh M with density distribution ρ, a set of points Y of the number k and the prescribed masses ν_i for each point satisfying that $\sum_{i=1}^{k} \nu_i = \mu(M)$

Result: The weights W that determines the optimal transport from M to $\sum_{i=1}^{k} \nu_i \delta_{y_i}$

Initialization:

$W \leftarrow 0$;

$\epsilon \leftarrow 0.01 * \mu(M)/\sqrt{k}$;

while $\|\nabla g(W)\|^2 < \epsilon$ **do**

Compute $Pow_W(Y) \cap M$ with Algo. 2;

Compute $\nabla g(W)$ and $\nabla^2 g(W)$;

Update W with either BB gradient descent or Newton's method.

5 Implementations

In this section, we approximate the 2D/3D geometries morphing between two triangular/tetrahedral meshes M_{init} and M_{end} with the help of Algo. 3. We have

Algorithm 4: Semi-discrete optimal transport based 2D/3D geometries morphing

Data: A initial mesh M_{init} with density distribution ρ_{init} and a destination mesh M_{end} with density distribution ρ_{end}

Result: The initial point set P_{init} and $RT(P_{init})$ and destination point set P_{end} and $RT(P_{end})$

Initialization:

Sample k points from M_{end} as Y;

$\nu_i \leftarrow \mu(M_{init})/k$;

Perform Algo. 3 to get W^*;

Perform Algo. 2 to get $Pow_{W^*}(Y)|M_{init}$;

$ConstrainedPD\_init \leftarrow Pow_{W^*}(Y)|M_{init}$;

$P_{init} \leftarrow$ `Compute_MassCenter(`$ConstrainedPD\_init$`)`;

$P_{end} \leftarrow Y$;

Perform Algo. 1 to get $RT(P_{init})$ and $RT(P_{end})$;

Note that in the Algo. 4 we begin at sampling the destination mesh M_{end} with a finite set of discrete points with assigned weights $W = 0$, so that geometry of M_{end} can be also represented as $Pow_{W=0}(Y)|M_{end}$ computed by Algo. 2. And then we prescribe the mass for each point y_i as ν_i such that $\mu(M_{init}) = \sum_i v_i$, we set the constant mass for each point as $\nu_i = \mu(M_{init})/k$, and uneven division can be explored in the future work. If $\mu(M_{init}) \neq \mu(M_{end})$, the density across each region $Pow_{W=0}(y_i)|M_{end}$ can be changed according to the prescribed mass ν_i to preserve conservation of the mass. The goal is to get optimal W^* so that $\mu(Pow_{W^*}(y_i)|M_{init}) = \nu_i$. The evolving geometries can be visualized as the gradually morphing regular triangulation with the changing vertex locations and connectivity matrix and the fixed vertex sequence.

The Multi-Parametric Toolbox (MPT) is used in Matlab to construct the polyhedrons of V/H-representations, and do the basic polytope manipulations e.g., intersect polyhedrons, extract extremes, compute volume of polyhedrons. Its optimization engines are based on solvers for a linear complementarity problem (LCP) that represents a superclass for LP and QP. The advantage of representing and solving the optimization problems as LCPs includes that a single solver covers all three scenarios and there is no need for multiple solvers that could potentially return different results [7].

MPT 3.0 provides a C-code implementation of the lexicographic Lemke's algorithm as MEX file to solve LCPs. It has been proved in [8] that the optimal algorithm that intersecting k convex polyhedrons has time complexity of $O(n \cdot \log k)$, where n is the total number of vertices. So for pairwise polyhedron intersection $k = 2$, the time complexity is $O(n)$. But in our case, we first construct V-polytopes from vertices, which are nonstandard representations, and it would make the problem much more difficult since only the vertices are provided at the beginning without any other facial information. So when one intersection of polyhedron executed, "computeHRe" that converts a pair of polyhedrons to H-representation will be performed twice. Then the built-in LCP solver in an external CDD package written in C-code can be called for numerical computation. Given a convex hull, its facet enumeration is $H \in R^{m \times d}$, and its vertex enumeration is $V \in R^{n \times d}$, conversion $H(V)$

takes $O(m \cdot d^2)$ and its dual conversion $V(H)$ takes $O(n \cdot d^2)$ and both of them as forms of the linear programming solved in the above LCP solver. [9]

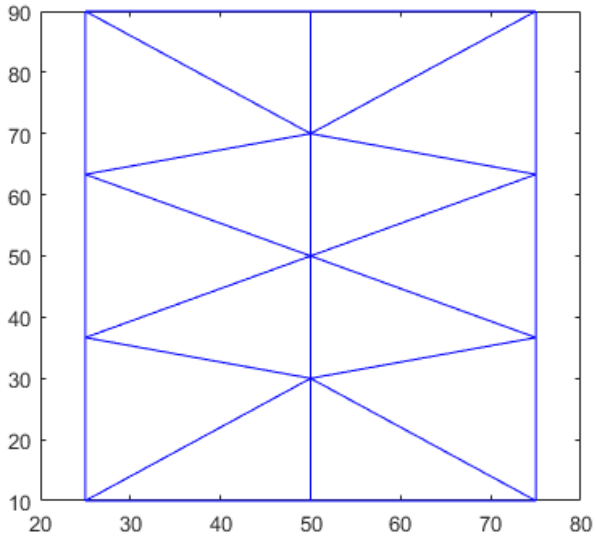
Therefore, we know that the time complexity for Algo.2 is a polynomial function of the number of sampling discrete points k, the number of the target mesh elements N_{init}, the dimension of the geometries d.

5.1 L_2 semi-discrete optimal transport in 2D meshes

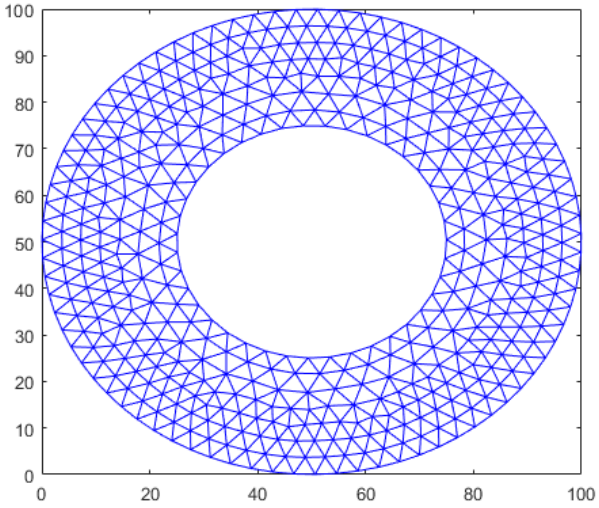
We demonstrate two 2D demos with Algo. 4 using Newton's method.

5.1.1 Experiment 1

We let the "rectangle" evolve to the "ring". First the initial and destination meshes are as follows.

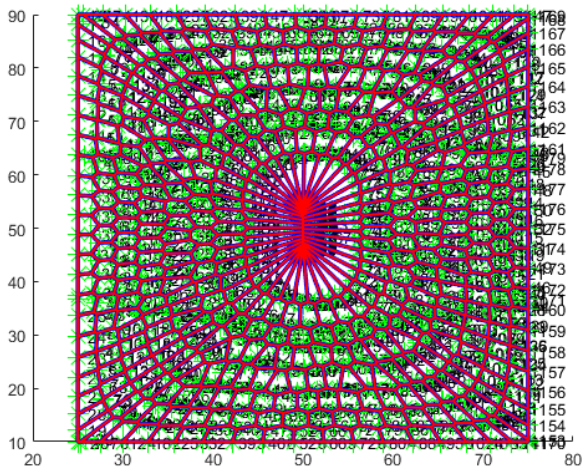


PD constrained by "rectangle"

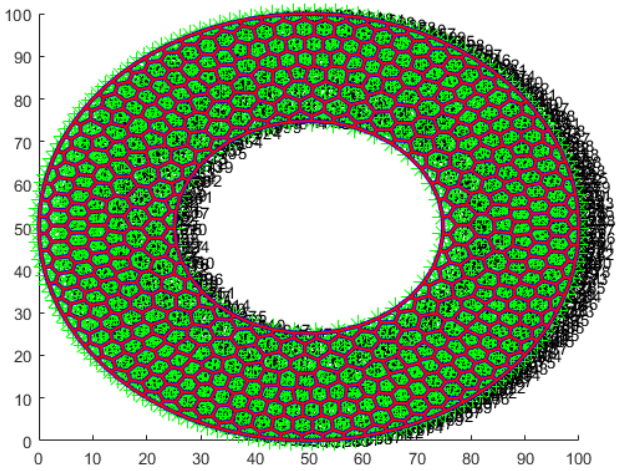


PD constrained by "ring"

where $k = 489$, $N_{init} = 14$. Then we visualize $Pow_{W=0}(Y)|M_{end}$ and $Pow_{W^*}(Y)|M_{init}$ as follows.

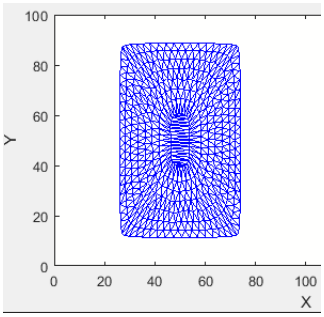


$Pow_{W^*}(Y)|M_{init}$

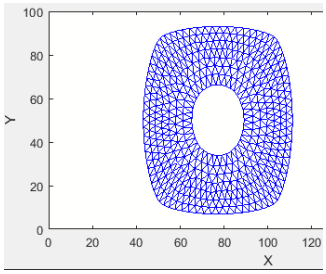


$Pow_{W=0}(Y)|M_{end}$

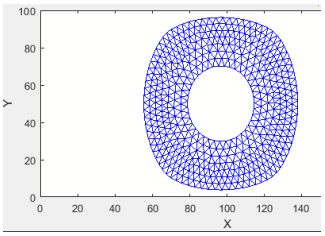
Finally we have the evolving procedure as follows.



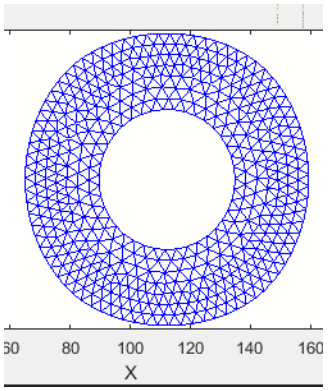
Morphing snapshot 1



Morphing snapshot 2



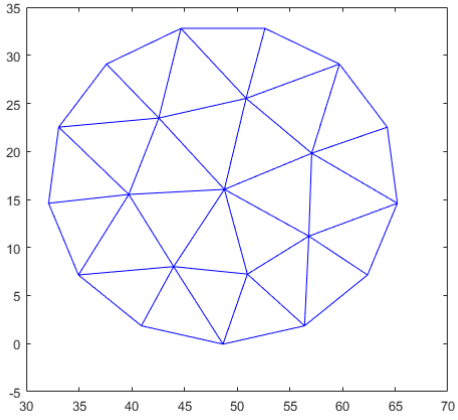
Morphing snapshot 3



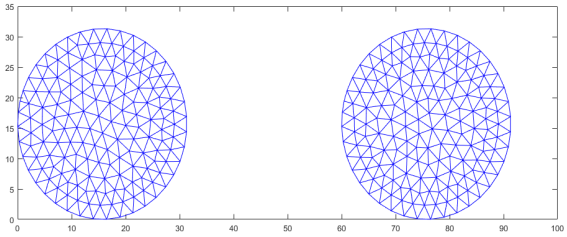
Morphing snapshot 4

5.1.2 Experiment 2

We let the "1circle" evolve to the "2circles". First the initial and destination meshes are as follows.

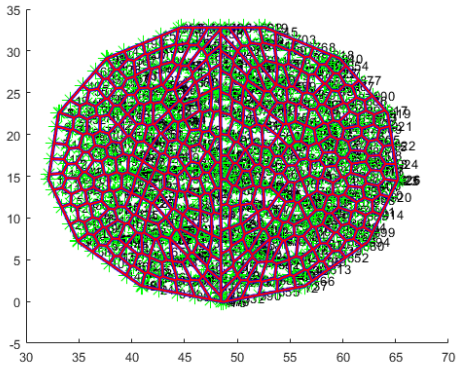


PD constrained by "1circle"

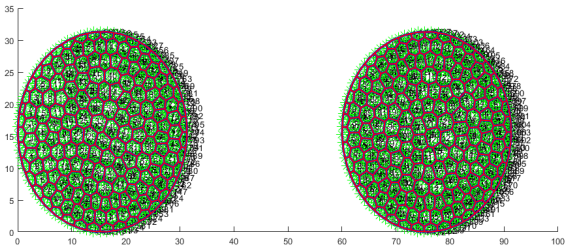


PD constrained by "2circles"

where $k = 594$, $N_{init} = 27$. Then we visualize $Pow_{W=0}(Y)|M_{end}$ and $Pow_{W^*}(Y)|M_{init}$ as follows.

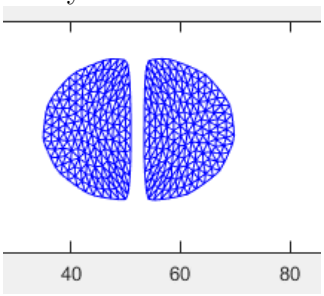


$Pow_{W^*}(Y)|M_{init}$

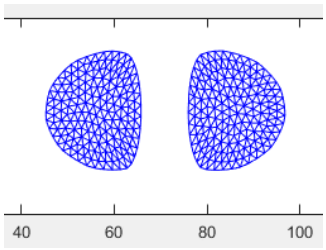


$Pow_{W=0}(Y)|M_{end}$

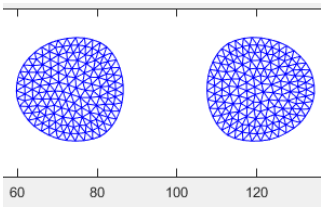
Finally we have the evolving procedure as follows.



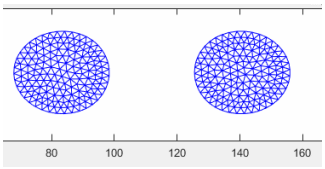
Morphing snapshot 1



Morphing snapshot 2



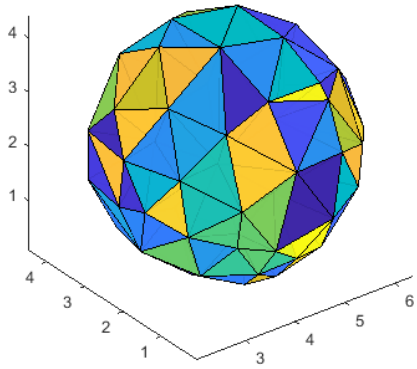
Morphing snapshot 3



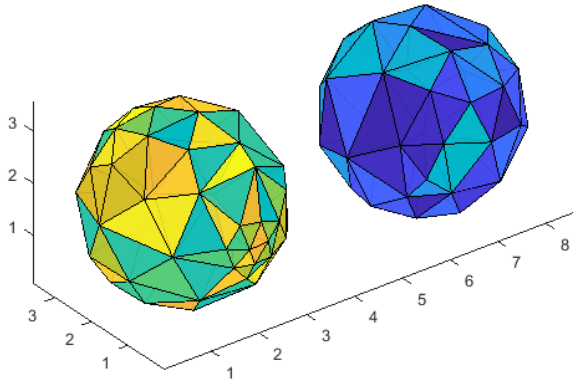
Morphing snapshot 4

5.2 L_2 semi-discrete optimal transport in 3D meshes

We demonstrate one 3D demos with Algo. 4 using Newton's method. We let the "1sphere" evolve to the "2spheres". First the initial and destination meshes are as follows.

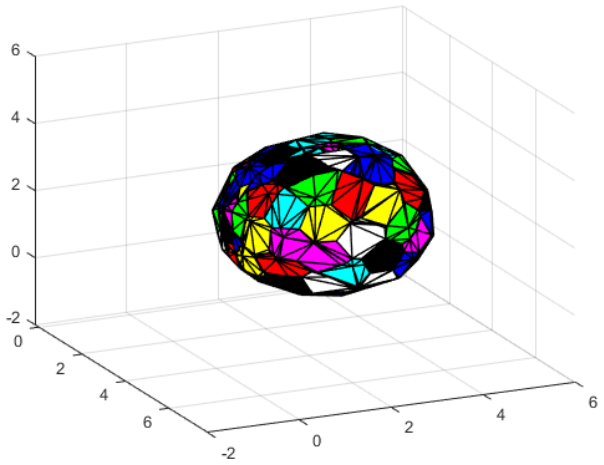


Initial mesh M_{init}

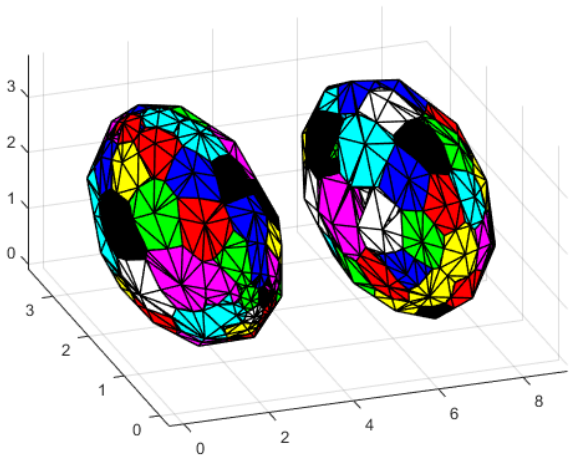


Destination mesh M_{end}

where $k = 130$, $N_{init} = 203$. Then we visualize $Pow_{W=0}(Y)|M_{end}$ and $Pow_{W^*}(Y)|M_{init}$ as follows.

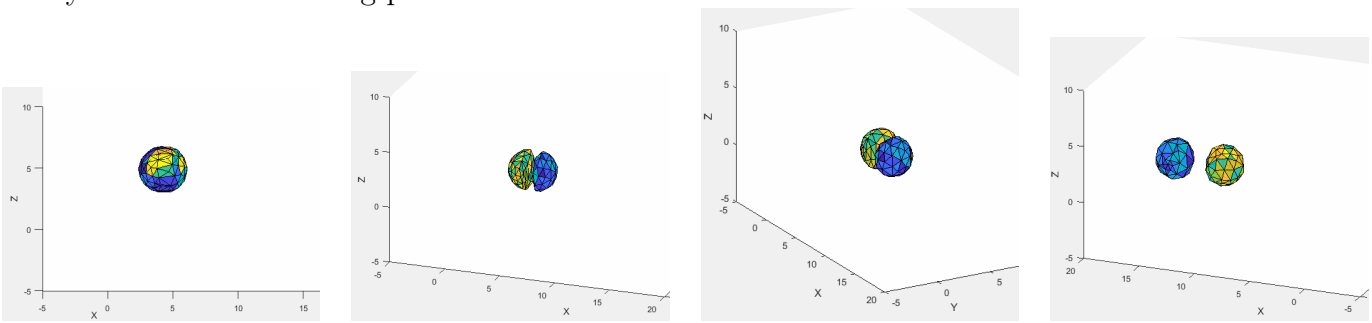


$Pow_{W^*}(Y)|M_{init}$



$Pow_{W=0}(Y)|M_{end}$

Finally we have the evolving procedure as follows.



Morphing snapshot 1 Morphing snapshot 2 Morphing snapshot 3 Morphing snapshot 4

6 Discussions & improvements

It can be noticed that the time it takes to compute the power diagrams constrained by the target mesh is proportional to the multiplication of the number of supporting points and the number of triangles/tetrahedra in mesh. We have used the profiler to measure the execution time of Algo. 2 for 6 optimization iterations in 3D experiment of Section. 5.2. We can observe that it took more than 70 percentage of the total execution time to compute $\mu(Pow_W(y_i)|M)$.

Profile Summary (Total time: 4111.824 s)

Flame Graph

Flame graph is not available because the number of function calls exceeds the current profiler history size of 500000. Increase the value for the 'historysize' profiling option and rerun the Profiler. For more information, see profile.

Generated 30-Dec-2021 03:37:54 using performance time.

Function Name	Calls	Total Time (s)	Self Time* (s)	Total Time Plot (dark band = self time)
test_3d_pd1	1	4109.314	0.326	
compute_restricted_polytopes_3d	6	4094.581	88.962	
Polyhedron.intersect	161994	3350.475	20.955	
Polyhedron.minHRep	331047	3325.338	273.147	
mpt_solve	6340197	2601.685	350.470	
mpt_solve>sub_callsolver	6340197	2196.961	198.167	
mpt_call_lcp	6495945	1998.794	1272.759	
lcp (MEX-file)	6495945	606.017	606.017	
Polyhedron.computeHRep	323988	495.445	126.808	
Polyhedron.Polyhedron>Polyhedron.Polyhedron	811972	465.929	329.938	
Polyhedron.minVRep	331250	263.656	25.691	
Polyhedron.interiorPoint	1472470	249.915	80.400	
Polyhedron.isEmptySet	845874	247.905	40.147	
Polyhedron.Polyhedron>Polyhedron.get.V	1311881	212.523	5.663	
convhulln	667866	209.280	138.609	
Polyhedron.computeVRep	161994	206.859	6.042	
Polyhedron.chebyCenter	176111	148.254	23.107	
rank	6820136	128.718	128.718	
validate_polyhedron	486983	120.968	120.968	
unique	1694611	104.749	81.630	
Polyhedron.isFullDim	669559	104.140	20.202	
polyfun\private\mergeDuplicatePoints	667865	62.353	7.947	
mpt_solve>sub_fix_nan	6340197	54.255	54.255	

Figure 2: Profiler results of Algo. 2 of 6 iterations in serial mode

As the resolution of the target mesh and the number of supporting points of PD that usually sampled from source mesh increase, the execution time will extend dramatically. It can be seen that in serial computing mode, the average time for one polyhedron intersection operation involved in 20 points approximately is about $20.7ms$.

6.1 Parallel computing acceleration

Since the intersection operations of $Pow_W(y_i)$ and the target mesh is independent from each other that means one loop iteration cannot depend on a previous iteration and the iterations are executed in a non-deterministic order. In the parallel computing toolbox, "for-loop" is replaced with "parfor-loop" so that the for-loop iterations can be executed in parallel on workers in a parallel pool. We connect Matlab with "helvetios" computing cluster of EPFL and configure the settings as follows.

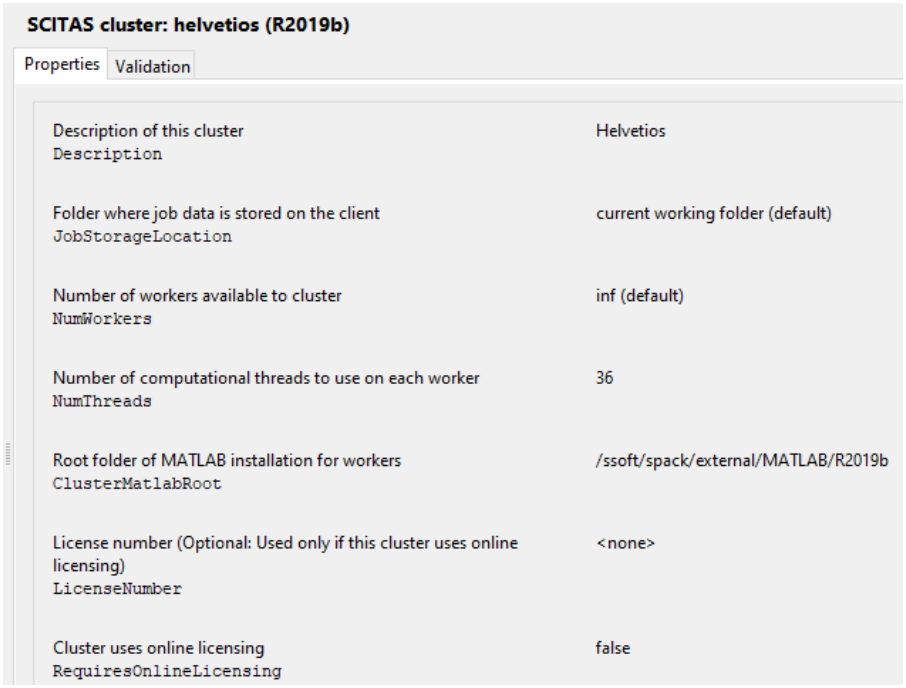
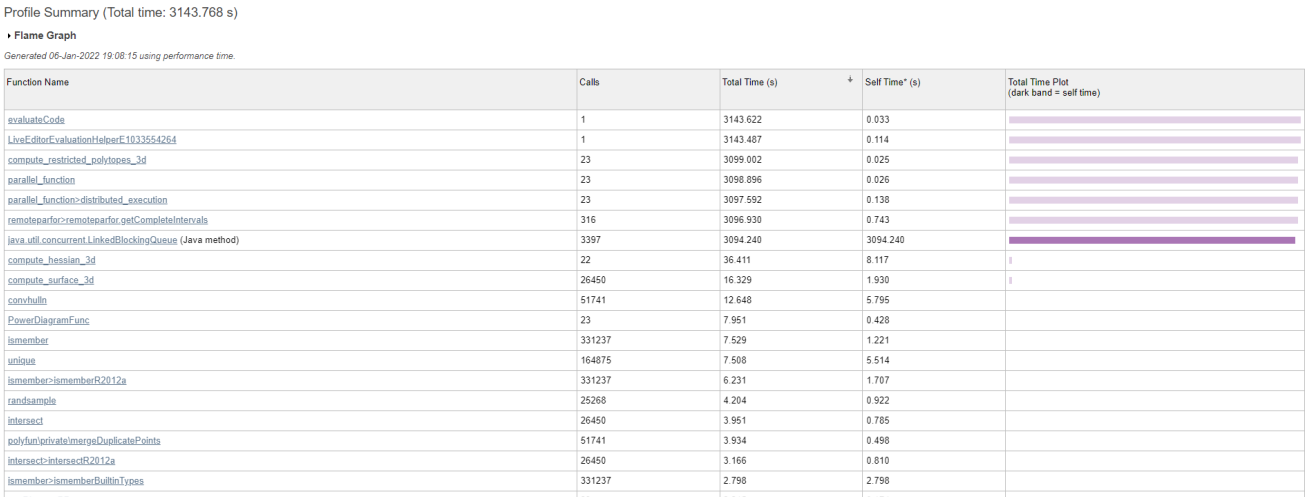


Figure 3: Configuration of parallel computing using "helvetios" cluster

Note that we can activate parallel computing locally as well. The number of workers to start on the local machine is equal to the number of cores at default. We iterate the subset $Pow_W(y_i)$ in parallel with 4 workers on the local laptop with CPU Intel i7-1065G7 1.30GHz. We can observe that this time the utilization rate of CPU is 100%, which means the computation resources have been fully exploited. And the average time to execute one polyhedron intersection operation in parallel mode is $4.9ms$, which is at least 4 times less than that of the original serial one. Part of the profiler result for the same program in Fig. (2) is as follows


Profile Summary (Total time: 3143.768 s)

Flame Graph

Generated 06-Jan-2022 19:08:15 using performance time.

Function Name	Calls	Total Time (s)	Self Time* (s)	Total Time Plot (dark band = self time)
evaluateCode	1	3143.622	0.033	
LiveEditorEvaluationHelperE1033554264	1	3143.487	0.114	
compute_restricted_polytopes_3d	23	3099.002	0.025	
parallel_function	23	3098.896	0.026	
parallel_function>distributed_execution	23	3097.592	0.138	
remoteparfor>remoteparfor.getCompleteIntervals	316	3096.930	0.743	
java.util.concurrent.LinkedBlockingQueue (Java method)	3397	3094.240	3094.240	
compute_hessian_3d	22	36.411	8.117	
compute_surface_3d	26450	16.329	1.930	
convhulln	51741	12.648	5.795	
PowerDiagramFunc	23	7.951	0.428	
ismember	331237	7.529	1.221	
unique	164875	7.508	5.514	
ismember>ismemberR2012a	331237	6.231	1.707	
randsample	25268	4.204	0.922	
intersect	26450	3.951	0.785	
polyfun\private\mergeDuplicatePoints	51741	3.934	0.498	
intersect>intersectR2012a	26450	3.166	0.810	
ismember>ismemberBuiltinTypes	331237	2.798	2.798	

Figure 4: Profiler results of Algo. 2 of 23 iterations in parallel mode

6.2 V-H representations conversion optimization

From the previous profiler results, it can be known that within one intersection operation if the input convex hulls are of vertex representation the two V-H conversion operations will take more than 90% of the time, in which the half-space representation is computed by *computeHRep* method. So in this section, we compute the H-representation of the convex hulls before the intersection operations such that the total time taken by *computeHRep* method will increase linearly with the number of mesh cells and discrete points rather than polynomially. Part of the profiler result for the same program in Fig. (2) is as follows, the average time to execute one polyhedron intersection operation with H-Rep convex hulls as intermediate inputs in parallel mode is 0.52ms, which is at least 10 times less than that of the original parallel one.

Profile Summary (Total time: 1318.651 s)

▸ Flame Graph

Generated 28-Jan-2022 10:26:19 using performance time.

Function Name	Calls	Total Time (s)	Self Time* (s)	Total Time Plot (dark band = self time)
evaluateCode	1	1318.512	0.021	
LiveEditorEvaluationHelperE1585049703	1	1318.436	0.207	
compute_restricted_polytopes_3d	75	1105.655	0.355	
parallel_function	225	1104.848	0.129	
parallel_function>distributed_execution	225	1098.787	1.255	
remoteparfor>remoteparfor.getCompleteIntervals	3215	1089.965	2.882	
java.util.concurrent.LinkedBlockingQueue (Java method)	6670	1061.723	1061.723	
compute_hessian_3d	74	169.563	37.928	
compute_surface_3d	98090	71.942	7.367	
convhulln	191985	63.567	30.594	
PowerDiagramFunc	75	42.996	2.258	
ismember	1116537	41.371	6.597	
unique	596427	37.641	27.726	
ismember>ismemberR2012a	1116537	34.328	8.740	
remoteparfor>remoteparfor.deserialize	5903	24.184	0.488	
deserialize	5903	23.696	1.041	
distcompdeserialize (MEX-file)	5903	22.372	6.800	
intersect	98090	18.637	4.107	

Figure 5: Profiler results of Algo. 2 of 75 iterations in parallel mode with H-Rep as inputs

6.3 Discussion & outlooks

We accomplish computation of the constrained power diagrams and 2D/3D geometries morphing in Matlab and have the following outlooks:

- Use "SCITAS" computing cluster to speed up further. Modify the computational graphics predicate functions e.g., *isColinear*, *isCoplanar* and *isUnique* to let them more robust and use different kinds of precision and arithmetics.
- Explore other patterns to visualize the morphing process apart from in the form of regular triangulation. Use other types of elements in Fig. (1) to comprise meshes and more generic density distribution μ for meshes.
- Optimize and change to other data structures to represent the subsets of the power diagram and polyhedrons such as the winged-edge data structure since it uses edges to keep track almost everything and maintain the edge table in which each entry contains information i.e., edge name, start vertex and end vertex, left face and right face, the predecessor and successor edges when traversing its left face, and the predecessor and successor edges when traversing

its right face. And the clockwise ordering (viewing from outside of the polyhedron) is used for traverse. [10]

- Optimal the parallel computing configuration with a specific workers. Scale up "parfor-loops" to the computing cluster. We can first start on the local multi-core desktop and measure the time required to run a calculation as a function of increasing numbers of workers. The test is called a strong scaling test. Generally the measure of time required for the calculation would decrease if more workers added so that we can estimate the parallel speed-up scalability of the code. Then we can decide whether it is useful to increase the number of workers in the parallel pool and to further scale up to cluster computing.
- Explore the applications of the constrained power diagram and semi-discrete optimal transport in other fields such as swarm path planning and bio-process in nature like cell division.

Acknowledges

Many thanks to Prof. Daniel Kuhn's insightful comments and guidance and sincere help and advice from Bahar Taskesen.

References

[1] Lévy, B. (2015). A numerical algorithm for L2 semi-discrete optimal transport in 3D. ESAIM: Mathematical Modelling and Numerical Analysis, 49(6), 1693-1715.

[2] Peyré, G., & Cuturi, M. (2019). Computational optimal transport: With applications to data science. Foundations and Trends in Machine Learning, 11(5-6), 355-607.

[3] Michal Zemek. (2009). Regular Triangulation in 3D and Its Applications. The State of the Art and Concept of Ph.D. Thesis.

[4] Herbert Edelsbrunner and Nimish R. Shah. Incremental topological flipping works for regular triangulations. In SCG 92: Proceedings of the eighth annual symposium on Computational geometry, pages 43–52, New York, NY, USA, 1992. ACM.

[5] J. Barzilai and J. Borwein. Two-point step size gradient method. IMA J. Numerical Analysis 8, 141–148, 1988

[6] Lévy, B., & Schwindt, E. L. (2018). Notions of optimal transport theory and how to implement them on a computer. Computers Graphics, 72, 135-148.

[7] M. Herceg, M. Kvasnica, C. N. Jones and M. Morari, "Multi-Parametric Toolbox 3.0," 2013 European Control Conference (ECC), 2013, pp. 502-510

[8] Chazelle, B. (1992). An optimal algorithm for intersecting three-dimensional convex polyhedra. SIAM Journal on Computing, 21(4), 671-696.

[9] D. Bremner, K. Fukuda, and A. Marzetta. Primal-dual methods for vertex and facet enumeration. In Proc. 13th Annu. ACM Sympos. Comput. Geom., pages 49–56, 1997.

[10] Baumgart, B. G. (1975, May). A polyhedron representation for computer vision. In Proceedings of the May 19-22, 1975, national computer conference and exposition (pp. 589-596).