SIAM J. Imaging Sciences
Vol. 11, No. 2, pp. 1134–1178

Density-Equalizing Maps for Simply Connected Open Surfaces*

Gary P. T. Choi† and Chris H. Rycroft‡

Abstract. In this paper, we are concerned with the problem of creating flattening maps of simply connected open surfaces in $\mathbb{R}^3$. Using a natural principle of density diffusion in physics, we propose an effective algorithm for computing density-equalizing maps with any prescribed density distribution. By varying the initial density distribution, a large variety of flattening maps with different properties can be achieved. For instance, area-preserving parameterizations of simply connected open surfaces can be easily computed. Experimental results are presented to demonstrate the effectiveness of our proposed method. Applications to data visualization and surface remeshing are explored.

Key words. density-equalizing map, cartogram, area-preserving parameterization, diffusion, data visualization, surface remeshing

AMS subject classifications. 68U05, 65D18, 76R50

DOI. 10.1137/17M1124796

1. Introduction.

The problem of producing maps has been tackled by scientists and cartographers for centuries. A classical map-making problem is to flatten the globe onto a plane. Numerous methods have been proposed, each aiming to preserve different geometric quantities. For instance, the Mercator projection produces a conformal planar map of the globe: angles and small objects are preserved but the area near the poles is seriously distorted.

One problem in computer graphics closely related to cartogram production is surface parameterization, which refers to the process of mapping a complicated surface to a simpler domain. With the advancement of the computer technology, three-dimensional (3D) graphics have become widespread in recent decades. To create realistic textures on 3D shapes, one common approach is to parameterize the 3D shapes onto $\mathbb{R}^2$. The texture can be designed on $\mathbb{R}^2$ and then be mapped back onto the 3D shapes. Again, different criteria of distortion minimization have led to the invention of a large number of parameterization algorithms.

Gastner and Newman [1] proposed an algorithm for producing density-equalizing cartograms based on the diffusion equation. Specifically, given a map and certain data defined on each part of the map (such as the population at different regions), the algorithm deforms the map such that the density, defined by the population per unit area, becomes a constant

*Received by the editors April 10, 2017; accepted for publication (in revised form) February 12, 2018; published electronically May 1, 2018.
http://www.siam.org/journals/siims/11-2/M112479.html
Funding: The work of the first author was partially supported by a fellowship from the Croucher Foundation. The work of the second author was supported by the Applied Mathematics Program of the U.S. Department of Energy (DOE) Office of Advanced Scientific Computing Research under contract DE-AC02-05CH11231.
†Corresponding author. John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138 (pchoi@g.harvard.edu).
‡John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138, and Mathematics Group, Lawrence Berkeley National Laboratory, Berkeley, CA 94720 (chr@seas.harvard.edu).

all over the deformed map. The diffusion-based cartogram generation approach has been widely used for data visualization. For instance, Dorling [2] applied this approach to visualize sociological data such as global population, income, and age-of-death at different regions. Colizza et al. [3] constructed a geographical representation of disease evolution in the United States for epidemics using this cartogram generation algorithm. Wake and Vredenburg [4] visualized global amphibian species diversity using the method. Other applications include the visualization of the democracies and autocracies of different countries [5], the race/ethnicity distribution of Twitter users in the United States [6], the rate of obesity for individuals in Canada [7], and the world citation network [8].

Inspired by the above approach, we develop an efficient finite-element algorithm for computing density-equalizing flattening maps of simply connected open surfaces in $\mathbb{R}^3$ onto $\mathbb{R}^2$. Given a simply connected open triangulated surface and certain quantities defined on all triangle elements of the surface, we first flatten the surface onto $\mathbb{R}^2$ by a natural flattening map. Then the flattened surface is deformed according to the given quantities using a fast iterative scheme. Furthermore, by altering the input quantities defined on the triangle elements, flattening maps with different properties can be achieved. For instance, area-preserving parameterizations of simply connected open surfaces can be easily obtained.

1.1. Contribution. The contribution of our work for computing density-equalizing flattening maps of simply connected open surfaces is as follows:

(i) Our approach is applicable to a wider class of surfaces when compared to the previous approach by Gastner and Newman [1]. The previous approach [1] works for two-dimensional (2D) domains, while ours works for simply connected open surfaces in $\mathbb{R}^3$.

(ii) We propose a linear formulation for computing a curvature-based flattening map of simply connected open surfaces. The flattening map effectively preserves the curvature of the input surface boundary and serves as a good initialization for the subsequent density-equalizing process.

(iii) We propose a new scheme for constructing an auxiliary region for the density diffusion. When compared to the previous approach [1], which makes use of a regular rectangular grid for constructing the auxiliary region, our approach produces a more adaptive auxiliary region that requires fewer points and hence reduces the computational cost.

(iv) We propose a finite-element iterative scheme for solving the density-diffusion problem without introducing the Fourier space as in the previous approach [1]. The scheme accelerates the computation for density-equalizing maps, with the accuracy well preserved. When compared to the state-of-the-art parameterization approaches, our algorithm also achieves higher efficiency and accuracy.

(v) Our proposed algorithm can be used for a wide range of applications, including the computation of area-preserving parameterizations, data visualization, and surface remeshing.

1.2. Organization of the paper. In section 2, we review the previous works on cartogram generation and surface parameterization. The physical principle of density equalization is outlined in section 3. In section 4, we describe our proposed method for achieving density-equalizing flattening maps of simply connected open surfaces. Experimental results are presented in section 5 for analyzing our proposed algorithm. In section 6, we discuss two

applications of our algorithm. In section 7, we conclude this paper with a discussion on the advantages, limitations, and possible extensions of our current approach.

2. Previous work. The problem of map generation has been studied by cartographers, geographers, and scientists for centuries. Readers are referred to Dorling [9] for a short survey of pre-existing cartogram production methods. Edelsbrunner and Waupotitsch [10] proposed a combinatorial approach to construct homeomorphisms with prescribed area distortion for cartogram generation. Keim, North, and Panse [11] developed the Cartodraw algorithm for producing contiguous cartograms. They also proposed [12] using medial-axis-based transformations for making cartograms.

In this work, cartogram generation is shown to be closely related to surface parameterization. For surface parameterization, a large variety of algorithms have been proposed by different research groups [13, 14, 15]. There are two major classes of surface parameterization algorithms, namely, conformal parameterization and authalic parameterization. Conformal parameterization aims to preserve the angles and hence the infinitesimal shapes of the surfaces while sacrificing the area ratios. By contrast, authalic parameterization aims to preserve the area measure of the surfaces while neglecting angular distortions. For conformal parameterization, established methods include linearization of the Laplace equation [16, 17], least-squares conformal mapping (LSCM) [18], discrete conformal parameterization (DCP) [19], angle-based flattening (ABF) [20, 21, 22], Dirichlet energy minimization [23], homolophic 1-form [24], Yamabe flow [25], circle patterns [26], spectral conformal mapping (SCP) [27], conformal prescription with metric scaling [28], discrete conformal equivalence [29], discrete Ricci flow [30, 31, 32], quasi-conformal composition [33, 34, 35, 36, 37], and boundary first flattening [38]. In contrast to conformal parameterization, only a few works on area-preserving parameterization have been reported. Proposed methods include locally authalic map [19], Lie advection of differential 2-forms [39], and optimal mass transport [40, 41, 42].

It is noteworthy that there are also numerous parameterization approaches trading off between the conformal and authalic distortions, such as area-preserving MIPS [43], as-rigid-as-possible (ARAP) parameterization [44], isometric metric [45], optimized conformal parameterization with controllable area distortions [46], angle-area distortion balancing maps [47], and isometric distortion energy minimization [48]. Other notable related parameterization approaches include landmark-constrained optimized conformal parameterization [49, 50, 51, 52], optimal conformal parameterization for surfaces with arbitrary topology [53, 54], quasi-conformal parameterization [55, 56, 57, 58, 36, 59], bounded distortion parameterization [60, 61, 62], composite majorization [63], approximate Killing vector fields [64], and the simplicial complex augmentation framework (SCAF) [65].

Our density-equalizing algorithm involves handling the density gradient field on triangle meshes. For the interpolation of vector fields, Whitney forms are used in this work. Whitney forms were introduced by Whitney [66] for algebraic topology and subsequently used as finite elements [67]. Readers are referred to the survey [68] for an overview of vector field processing on triangle meshes. A related approach to introducing weights on triangle meshes has also appeared in the construction of orthogonal dual meshes [69].

3. Background. Our work aims to produce flattening maps based on a physical principle of diffusion. The diffusion-based method for producing cartogram proposed by Gastner and

Newman [1] (denoted by *GN* for the rest of the paper) is outlined as follows. Given a planar map and a quantity called the *population* defined on every part of the map, let ρ be the density field defined by the quantity per unit area. The map can be deformed by equalizing the density field ρ using the advection equation

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot \mathbf{j}, \tag{3.1}$$

where the flux is given by Fick's law,

$$\mathbf{j} = -\nabla \rho. \tag{3.2}$$

This yields the diffusion equation

$$\frac{\partial \rho}{\partial t} = \Delta \rho. \tag{3.3}$$

Since time can be rescaled in the subsequent analysis, the diffusion constant in Fick's law is set to 1. Any tracers that are being carried by this density flux will move with velocity

$$\mathbf{v}(\mathbf{r}, t) = \frac{\mathbf{j}}{\rho} = -\frac{\nabla \rho}{\rho}. \tag{3.4}$$

Note that the term $\nabla \rho / \rho$ is independent of the absolute scale of ρ, and hence the update of the velocity field is stable.

If (3.3) is solved to steady state, and the map is deformed according to the velocity field in (3.4), then the final state of the map will have equalized density. To track the deformation of the map, GN introduced tracers $\mathbf{r}(t)$ that follow the velocity field according to

$$\mathbf{r}(t) = \mathbf{r}(0) + \int_0^t \mathbf{v}(\mathbf{r}, \tau) d\tau. \tag{3.5}$$

In other words, taking $t \to \infty$, the above displacement $\mathbf{r}(t)$ produces a map that achieves equalized density per unit area. To avoid infinite expansion of the map, GN proposed to construct a large rectangular auxiliary region, called the *sea*, surrounding the region of interest. Defining the density in the sea to be the average density of the region of interest ensures that the area of the deformed map is as same as that of the initial map. In GN, the above procedures were developed using finite difference grids, and the above equations were solved in Fourier space.

There is room for improvement in the above-mentioned approach in two major aspects. First, the above 2D finite difference approach works for planar domains but not for general simply connected open surfaces in $\mathbb{R}^3$. Second, the large rectangular sea and the large number of grid points may require a long computational time. In this work, an algorithm that further enhances the above-mentioned approach in the two aspects is proposed. The details of our proposed algorithm is described in the following section.

4. Our proposed method. Let S be a simply connected open surface in $\mathbb{R}^3$, and let ρ be a prescribed density distribution. Our goal is to compute a flattening map $f : S \to \mathbb{R}^2$ such that the Jacobian J_f satisfies

$$J_f \propto \rho. \tag{4.1}$$

In other words, the final density per unit area in the flattening map becomes a constant.

Our proposed algorithm primarily consists of three steps, described in sections 4.1, 4.2, and 4.3. We remark that if the input surface is planar, the first step can be skipped. In the following discussions, S is discretized as a triangle mesh $(\mathcal{V}, \mathcal{E}, \mathcal{F})$, where $\mathcal{V}$ is the vertex set, $\mathcal{E}$ is the edge set, and $\mathcal{F}$ is the triangular face set. ρ is discretized as $\rho^{\mathcal{F}}$ on every triangle element $T \subset \mathcal{F}$.

4.1. Initialization: Fast curvature-based flattening map. To compute the density-equalization process, the first step is to flatten S onto $\mathbb{R}^2$. To minimize the discrepancy between the surface and the flattening result, it is desirable that the outline of the flattening result be similar to the surface boundary. We first simplify the problem by considering only the curve flattening problem of the surface boundary. Then we construct a surface flattening map based on the curve flattening result.

4.1.1. Curvature-based flattening of the surface boundary. Let γ be the boundary of the given surface S. Note that γ is a simple closed curve in $\mathbb{R}^3$, and hence we can write it as an arclength parameterized curve $\gamma = \gamma(t) : [0, l_\gamma] \to \mathbb{R}^3$, where l_γ is the total arclength of γ. Our goal is to flatten γ onto $\mathbb{R}^2$ using a map $\varphi : [0, l_\gamma] \to \mathbb{R}^2$ and then obtain the entire flattening map of the surface S. For γ, we can compute two quantities: the curvature κ_γ and the torsion τ_γ. Note that the curvature κ_γ measures the deviation of γ from a straight line, and the torsion τ_γ measures the deviation of γ from a planar curve. By the fundamental theorem of space curves [70], γ is completely determined (up to rigid motion) by κ_γ and τ_γ.

Motivated by the above, we consider mapping γ to $\varphi(\gamma)$ such that $\kappa_\gamma \approx \kappa_{\varphi(\gamma)}$ and $\tau_{\varphi(\gamma)} = 0$. In other words, we project γ onto the space of planar convex curves such that the curvature is preserved as much as possible. By Frenet–Serret formulas [70],

$$\mathbf{T}'(t) = \kappa_\gamma(t)\|\gamma'(t)\|\mathbf{N}(t), \tag{4.2}$$

where $\mathbf{T}$ and $\mathbf{N}$ are, respectively, the unit tangent and unit normal of γ. It follows that

$$\kappa_\gamma(t) = \frac{\|\mathbf{T}'(t)\|}{\|\gamma'(t)\|}. \tag{4.3}$$

After obtaining κ_γ, our goal is to construct a projection of γ onto the space of planar simple convex closed curves. Note that for any simple closed planar curve $\mathcal{C} \subset \mathbb{R}^2$, the total signed curvature of $\mathcal{C}$ is a constant [70]:

$$\int_{\mathcal{C}} k_{\mathcal{C}}(t)dt = 2\pi. \tag{4.4}$$

Now, to construct a closed planar curve φ with total arclength same as γ, we set the target signed curvature k to be

$$k(s) = \frac{2\pi\kappa_\gamma(s)}{\int_\gamma \kappa_\gamma(t)dt} \geq 0. \tag{4.5}$$

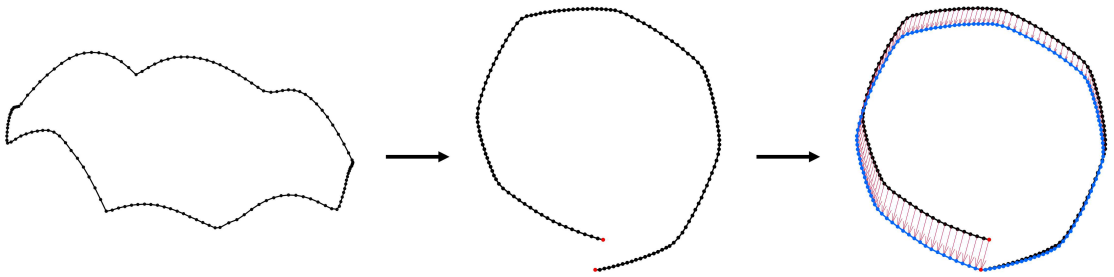

Figure 1. *Our curvature-based curve flattening procedure. Given a closed curve in* $\mathbb{R}^3$ *which represents the surface boundary, a plane curve is first constructed using* (4.6)*. The adjustment* (4.10) *is then used for obtaining a simple closed convex curve.*

Then consider the curve

$$\varphi(s) = \left(\int_0^s \cos\theta(u)du, \int_0^s \sin\theta(u)du \right), \tag{4.6}$$

where

$$\theta(u) = \int_0^u k(t)dt. \tag{4.7}$$

It is easy to check that

$$\varphi'(s) = (\cos\theta(s), \sin\theta(s)), \tag{4.8}$$

and hence φ is an arclength parameterized curve. Moreover, we have

$$k_\varphi(s) = \theta'(s) = k(s) \geq 0. \tag{4.9}$$

However, it should be noted that φ may not be a closed curve. In other words, there may be a small gap between $\varphi(0)$ and $\varphi(l_\gamma)$ with $0 \leq \|\varphi(l_\gamma) - \varphi(0)\| \ll L$, where L is the total arclength of φ. To enforce that φ is closed, we consider updating it to

$$\varphi(s) \leftarrow \varphi(s) - \frac{s}{l_\gamma} \left(\varphi(l_\gamma) - \varphi(0)\right). \tag{4.10}$$

In fact, φ becomes a simple closed convex plane curve under this adjustment. The proof is provided in the appendix (see Appendix A). Figure 1 illustrates our curve flattening procedure. Our algorithm is summarized in Algorithm 1.

We make two remarks about the proposed curve flattening scheme before proceeding to the next step.

(i) Note that the curvature κ_γ of a space curve is always unsigned by its mathematical definition, which leads to the resulting convex plane curve. In case it is desired to flatten the surface boundary as a nonconvex shape, one can introduce a sign on certain $\kappa_\gamma(s)$ before defining the target curvature k. This modification results in a nonconvex plane curve under the curve flattening scheme, although the proof we provide about φ being a simple closed plane curve is no longer applicable.

Algorithm 1: Curvature-based curve flattening.

Input: The boundary γ of a simply connected open surface S in $\mathbb{R}^3$.
Output: A curvature-based flattened curve φ.

1 Let $\gamma = \{v_j\}_{j=1}^{b}$ be the boundary vertices of S in counterclockwise order. Compute the curvature $\kappa = \frac{\|\mathbf{T}'\|}{\|\gamma'\|}$;
2 Rescale κ by $\kappa \leftarrow \frac{2\pi\kappa}{\int_\gamma \kappa(s)ds}$;
3 Obtain the flattened curve $\varphi(s) = \left(\int_0^s \cos\theta(u)du, \int_0^s \sin\theta(u)du\right)$, where $\theta(u) = \int_0^u \kappa(t)dt$;
4 Adjust the map by $\varphi(s) \leftarrow \varphi(s) - \frac{s}{l_\gamma}\left(\varphi(l_\gamma) - \varphi(0)\right)$;

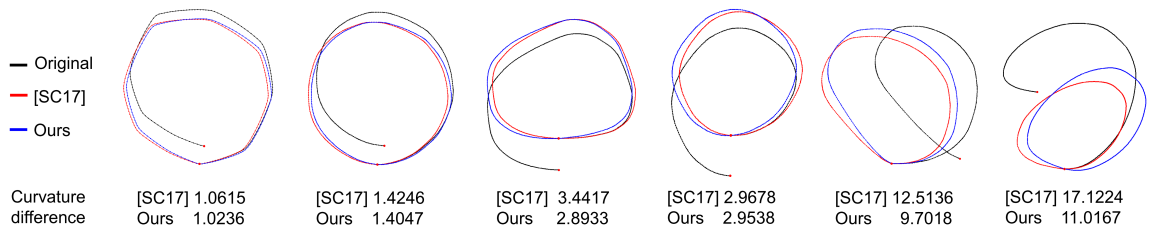

Figure 2. *A comparison between our method and the method by Sawhney and Crane [38] for closing a loop. We first construct six test cases of open curves of length 2π, possibly with a large gap or even self-intersections. We then close the curves by solving the energy minimization problem (equation (18) in [38]) and by our proposed adjustment step (4.10). We quantitatively compare the two methods by evaluating the L_2 norm of the curvature difference $\|\kappa_{closed} - \kappa_{open}\|_2$ between each closed curve and the original curve.*

(ii) Sawhney and Crane [38] proposed another method for constructing a closed plane curve using a similar curve integration approach. There are two major differences between their method and ours:
   (a) Their method constructs the plane curve based on a prescribed target curvature density, while ours constructs the plane curve based on the curvature of the surface boundary.
   (b) Their method enforces the closed loop condition by introducing an energy minimization step that adjusts the length of the tangents before the curve integration step, while ours ensures the condition by the adjustment (4.10).

   A comparison between our approach and their energy minimization approach is provided in Figure 2. Both methods are able to enforce the closed loop condition, and our method achieves a smaller change in curvature. This reflects the advantage of our method in constructing a closed curve that preserves the target curvature as much as possible. For the computation time, both methods require $O(n)$ operations, where n is the number of boundary vertices. However, unlike the method in [38], our method does not involve any matrix multiplication or inverse. A simple operation count suggests that our method is more efficient.

After obtaining the simple closed plane curve φ, we can use it as a boundary constraint and compute a map $\phi : S \to \mathbb{R}^2$ as an initial flattening map of the entire surface S. Two

methods for surface flattening are suggested below.

4.1.2. Curvature-based Tutte flattening map. One way to construct a bijective planar map ϕ is the graph embedding method of Tutte [71], which has been shown to be a good initialization for several existing parameterization approaches. For instance, Gu et al. [23] used the spherical Tutte embedding followed by a nonlinear harmonic energy minimization scheme for achieving a spherical conformal parameterization. Smith and Schaefer [45] also combined the Tutte initialization onto the unit disk with a nonlinear isometric distortion minimization scheme to compute a bijective parameterization. Here, we combine the Tutte mapping method with our curvature-based curve flattening result to obtain an initial surface flattening map.

To give an overview of the Tutte mapping method, we first introduce the concept of an adjacency matrix. The *adjacency matrix* M is a $|\mathcal{V}| \times |\mathcal{V}|$ matrix defined by

$$M_{ij} = \begin{cases} 1 & \text{if } [i,j] \in \mathcal{E}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.11}$$

In other words, the adjacency matrix only takes the combinatorial information of the input triangle mesh into account and neglects the geometry of it. It was proved by Tutte [71] that there exists a bijective map ϕ between any simply connected open triangulated surface S in $\mathbb{R}^3$ and any convex polygon P on $\mathbb{C}$ with the aid of the adjacency matrix. More explicitly, by representing ϕ as a complex column vector with length $|\mathcal{V}|$, ϕ can be obtained by solving the complex linear system

$$\begin{cases} M^{\text{Tutte}}\phi(v) = 0 & \text{if } v \in S \setminus \partial S, \\ \phi(\partial S) = \partial P, \end{cases} \tag{4.12}$$

where

$$M_{ij}^{\text{Tutte}} = \begin{cases} M_{ij} & \text{if } [x_i, x_j] \in \mathcal{E}, \\ -\sum_{t \neq i} M_{it} & \text{if } j = i, \\ 0 & \text{otherwise.} \end{cases} \tag{4.13}$$

Here the boundary mapping $\phi : \partial S \to \partial P$ can be any bijective map.

The choice of the boundary shape affects the surface flattening result. Some common choices include the unit circle and rectangle, which, respectively, result in a disk parameterization $\phi : S \to \mathbb{D}$ and a rectangular parameterization $\phi : S \to R$. While these choices lead to a simple parameter domain, the distortion of the surface under the flattening map may be large. In particular, the triangle elements near the shape boundary may be seriously squeezed. In contrast to these simple shapes, our curvature-based flattened curve φ resembles the shape of the surface boundary and hence leads to a smaller distortion in the flattened map with the occurrence of such extreme triangle elements reduced. Hence, we propose to use φ as the convex boundary constraint. Let P be the domain enclosed by φ. Then a bijective Tutte flattening map $\phi : S \to P$ can be easily obtained, as described in Algorithm 2.

We remark that if φ was modified to be nonconvex, one has to make a few tweaks to the above approach for guaranteeing the bijectivity of $\phi : S \to P$. One possible way is to compute two disk Tutte maps $\phi_1 : S \to \mathbb{D}$ and $\phi_2 : P \to \mathbb{D}$ and obtain the composition map $\phi = \phi_2^{-1} \circ \phi_1 : S \to P$. As $\mathbb{D}$ is convex, both ϕ_1, ϕ_2 are bijective, and hence ϕ is also.

Algorithm 2: Curvature-based Tutte flattening map.

Input: A simply connected open surface S in $\mathbb{R}^3$.

Output: A curvature-based flattening map $\phi : S \to P$, where P is a planar convex domain.

1 Let $\gamma = \{v_j\}_{j=1}^b$ be the boundary vertices of S. Compute the curvature-based curve flattening $\varphi : \gamma \to \mathbb{C}$;

2 Compute the adjacency matrix M with $M_{ij} = \begin{cases} 1 & \text{if } [i,j] \in \mathcal{E}, \\ 0 & \text{otherwise;} \end{cases}$

3 Solve the linear system $\begin{cases} M^{\text{Tutte}}\phi(v) = 0 & \text{if } v \in S \setminus \partial S, \\ \phi(v_j) = \varphi(v_j) & \text{for all } \{v_j\}_{j=1}^b, \end{cases}$ and obtain the desired map ϕ;

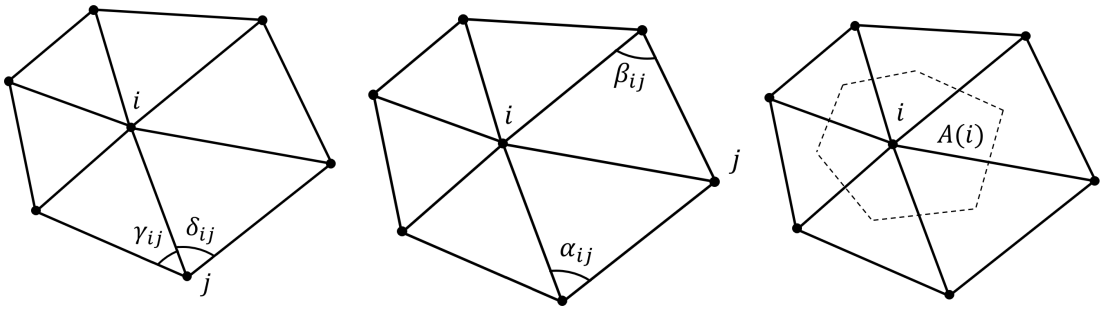

Figure 3. *Left: the angles γ_{ij} and δ_{ij} in the locally authalic Chi energy. Middle: the two angles α_{ij} and β_{ij} opposite the edge $[i,j]$ in the cotangent Laplacian. Right: the vertex area $A(i)$ of a vertex i.*

4.1.3. Curvature-based locally authalic flattening map. By changing the matrix M^{Tutte} in the above method, another way to construct ϕ can be obtained. Desbrun, Meyer, and Alliez [19] proposed a mapping scheme by minimizing the quadratic Chi energy

$$E_\chi(\phi) = \sum_{j \in N(i)} \frac{\cot \gamma_{ij} + \cot \delta_{ij}}{|x_i - x_j|^2} |\phi(x_i) - \phi(x_j)|^2, \tag{4.14}$$

where γ_{ij} and δ_{ij} are the two angles at x_j as illustrated in Figure 3 (left). The minimization of the Chi energy aims to find a locally authalic mapping $\phi : S \to \mathbb{R}^2$ that preserves the local 1-ring area at every vertex as much as possible. The associated authalic matrix of this energy is given by

$$M_{ij}^\chi = \begin{cases} \frac{\cot \gamma_{ij} + \cot \delta_{ij}}{|x_i - x_j|^2} & \text{if } [x_i, x_j] \in \mathcal{E}, \\ -\sum_{t \neq i} M_{it}^\chi & \text{if } j = i, \\ 0 & \text{otherwise.} \end{cases} \tag{4.15}$$

Now consider replacing M^{Tutte} in the Tutte flattening algorithm by M^χ and solve for a new flattening map. It is noteworthy that the minimizer of the Chi energy is not a globally

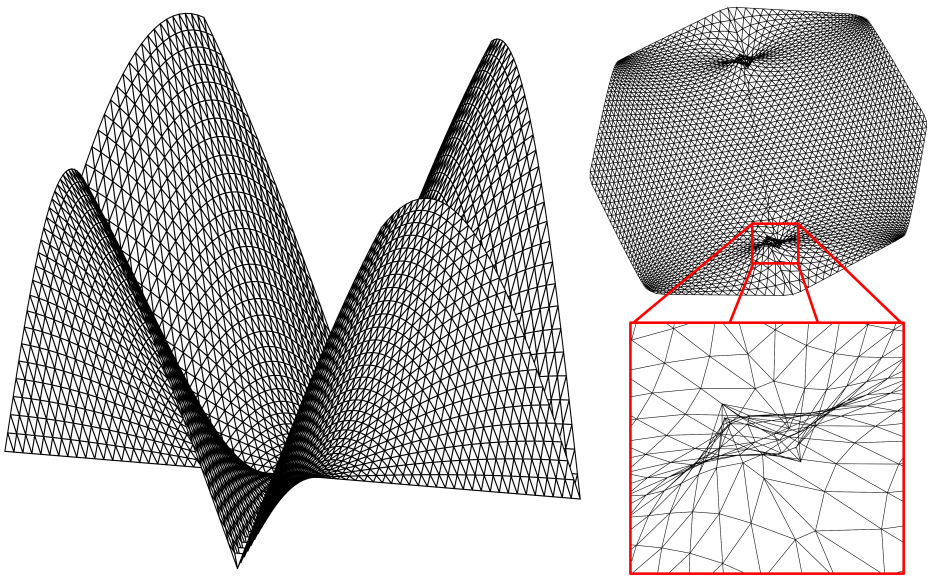

Figure 4. *An example for which the locally authalic map contains overlaps. Left: a mesh that contains many sliver triangles, violating the convex combination mapping property* [13]. *Right: the curvature-based locally authalic flattening map, with a zoom-in of the triangle overlaps.*

optimal area-preserving mapping. Nevertheless, it serves as a reasonably good and simple initialization for our density-equalization problem. More explicitly, using our curvature-based boundary constraint, ϕ can be obtained by solving the following complex linear system:

$$\begin{cases} M^{\chi}\phi(v) = 0 & \text{if } v \in S \setminus \partial S, \\ \phi(\partial S) = \varphi. \end{cases} \tag{4.16}$$

Note that, unlike the Tutte map, the bijectivity of the locally authalic map is only guaranteed when the convex combination mapping property [13] is satisfied, that is, all $\cot\gamma_{ij} + \cot\delta_{ij}$ in M^{χ} are nonnegative. This condition is equivalent to $\gamma_{ij} + \delta_{ij} \le \pi$ for all i, j. Figure 4 shows a mesh violating this condition, with the locally authalic flattening map containing mesh fold-overs. In this case, we simply resort to the Tutte map for ensuring the bijectivity. The curvature-based locally authalic flattening map is summarized in Algorithm 3.

In addition, in case it is desired to balance between the local 1-ring area distortion and the local angle distortion, one can replace M^{χ} in (4.16) by the combined matrix [19] $\lambda M^{\chi} + (1-\lambda)M^{\text{Cotan}}$, where $\lambda \in [0,1]$ is a balancing parameter and M^{Cotan} is the cotangent Laplacian [72] given by

$$M_{ij}^{\text{Cotan}} = \begin{cases} \frac{1}{2}(\cot\alpha_{ij} + \cot\beta_{ij}) & \text{if } [x_i, x_j] \in \mathcal{E}, \\ -\sum_{t\neq i} M_{it}^{\text{Cotan}} & \text{if } j = i, \\ 0 & \text{otherwise,} \end{cases} \tag{4.17}$$

with α_{ij} and β_{ij} being the two angles opposite the edge $[i, j]$, as illustrated in Figure 3 (middle).

We remark that the curvature-based flattening algorithms we introduced above are a good choice of initialization for our problem for the following reasons:

Algorithm 3: Curvature-based locally authalic flattening map.

Input: A simply connected open surface S in $\mathbb{R}^3$.

Output: A curvature-based locally authalic flattening map $\phi : S \to \mathbb{R}^2$.

1 Let $\gamma = \{v_j\}_{j=1}^b$ be the boundary vertices of S. Compute the curvature-based curve flattening $\varphi : \gamma \to \mathbb{C}$;

2 Compute the authalic matrix $M_{ij}^\chi = \begin{cases} \frac{\cot \gamma_{ij} + \cot \delta_{ij}}{|x_i - x_j|^2} & \text{if } [x_i, x_j] \in \mathcal{E}, \\ -\sum_{t \neq i} M_{it}^\chi & \text{if } j = i, \\ 0 & \text{otherwise;} \end{cases}$

3 Solve the linear system $\begin{cases} M^\chi \phi(v) = 0 & \text{if } v \in S \setminus \partial S, \\ \phi(v_j) = \varphi(v_j) & \text{for all } \{v_j\}_{j=1}^b, \end{cases}$ and obtain the desired map ϕ;

4 In case ϕ contains overlaps, resort to the Tutte map using Algorithm 2;

(i) The curve flattening procedure produces a planar curve that resembles the curvature of the given surface boundary. Unlike other conventional parameterizations which map surfaces to a standard planar domain, our curvature-based flattening method results in a more natural flattening map that avoids squeezing particular regions. The diffusion process can then be more accurately executed.

(ii) The computation of the flattening maps is highly efficient. Both algorithms only involve solving one complex linear system without any iterative procedures.

4.2. Construction of sea via reflection. In the diffusion-based approach of GN, one important step is to set up a sea surrounding the area of interest. The presence of the sea allows the area of interest to deform freely, and setting the initial density at the sea as the mean density avoids arbitrary expansion of the region under the diffusion process. In this work, we propose a new method for the construction of such a sea for the diffusion.

If the simply connected open surface S is not planar, then the above-mentioned curvature-based flattening methods give us an initial flattening map $\mathbf{r}_0 = \phi(S)$ in $\mathbb{R}^2$. If S is initially planar, we skip the above step and set $\mathbf{r}_0 = S$. In other words, we treat S itself as the initial flattening map.

Now, we shrink the initial map $\mathbf{r}_0$ and place it inside the unit circle $\mathbb{S}^1 := \{z \in \mathbb{C} : |z| = 1\}$. Note that there will be certain gaps between the shrunk map and the circular boundary. Denote the edge length of the shrunk flattening map by l. We fill up the gaps using uniformly distributed points with distance l. This process results in an even distribution of points all over the unit disk $\mathbb{D} := \{z \in \mathbb{C} : |z| \leq 1\}$. We then triangulate the new points using the Delaunay triangulation. This gives us a triangulation $\mathbb{D}_T$ of the unit disk.

Next, we aim to construct a sea surrounding the unit disk in a natural way. Consider the reflection mapping $g : \mathbb{D} \to \mathbb{C} \setminus \mathbb{D}$ defined by

$$g(z) = \frac{1}{\bar{z}}. \tag{4.18}$$

It is easy to observe that g is bijective. In the discrete case, the above map sends the triangulated unit disk $\mathbb{D}_T$ to a large polygonal region R in $\mathbb{C}$ with the region of $\mathbb{D}$ punctured.

We now glue $\mathbb{D}_T$ and $g(\mathbb{D}_T)$ along the circular boundary $\partial\mathbb{D}_T$. More explicitly, denote the glued mesh by $\widetilde{S} = (\widetilde{\mathcal{V}}, \widetilde{\mathcal{E}}, \widetilde{\mathcal{F}})$. We have

$$\widetilde{\mathcal{V}} = \{z\}_{z\in\mathbb{D}_T} \cup \left\{\frac{1}{\overline{z}}\right\}_{z\in\mathbb{D}_T\setminus\partial(\mathbb{D}_T)}, \tag{4.19}$$

$$\widetilde{\mathcal{F}} = \mathcal{F} \cup \left\{\left[\frac{1}{\overline{z}_i}, \frac{1}{\overline{z}_j}, \frac{1}{\overline{z}_k}\right] : [z_i, z_j, z_k] \in \mathcal{F}\right\}, \tag{4.20}$$

and

$$\widetilde{\mathcal{E}} = \{[z_i, z_j] : [z_i, z_j] \text{ is an edge of a face } T \in \widetilde{\mathcal{F}}\}. \tag{4.21}$$

In practice, the presence of extremely large triangles at the outermost part of the glued mesh due to reflection may result in numerical instability in the subsequent computations. Therefore, we remove those triangles by performing a simple truncation by removing the part far away from the unit disk $\mathbb{D}$. More explicitly, we remove all vertices and faces of $\widetilde{S}$ outside $\{z : |z| > \eta\}$, where η is a thresholding parameter. To set a reasonable η, we note that GN pointed out that having a sea with dimensions a few times the linear extent of the region of interest is sufficient. Therefore, in practice we set $\eta = 5$. Our experiment indicates that such truncation does not affect the accuracy of the density-equalizing map. Finally, we rescale the glued mesh to restore the size of the flattening map. By an abuse of notation, we continue using $\mathbf{r}_0$ to represent the entire region. The above procedures are summarized in Algorithm 4. A graphical illustration of the construction is shown in Figure 5.

Algorithm 4: Construction of sea via reflection.

Input: An initial flattening map $\mathbf{r}_0$, a thresholding parameter η.
Output: An updated map $\mathbf{r}_0$ with a sea surrounding the original domain.

1 Shrink $\mathbf{r}_0$ to sit inside the unit circle $\mathbb{S}^1$;
2 Fill up the gaps between the unit circle and the shrunk map by uniformly distributed points with distance l, where l is the average edge length of $\mathbf{r}_0$;
3 Perform a constrained Delaunay triangulation that triangulates the unit disk with the newly added points. The connectivity of $\mathbf{r}_0$ is kept unchanged;
4 Apply the reflection map $g(z) = \frac{1}{\overline{z}}$ to the triangulated unit disk $\mathbb{D}_T$;
5 Glue $\mathbb{D}_T$ and $g(\mathbb{D}_T)$. Update $\mathbf{r}_0$ by the glued result;
6 Remove all vertices and faces of $\mathbf{r}_0$ outside $\{z : |z| > \eta\}$;
7 Rescale $\mathbf{r}_0$ to restore the size of the flattening map;

One advantage of our construction is that the mesh size of the constructed sea is adaptive. Unlike the approach in GN, which used a uniform finite difference grid for the sea, our construction produces a natural distribution of points at the sea that avoids redundant computation. More specifically, let z_1, z_2 be two points at the interior of the unit disk $\mathbb{D}$. Under the reflection $z \mapsto \frac{1}{\overline{z}}$, we have

$$\left|\frac{1}{\overline{z_1}} - \frac{1}{\overline{z_2}}\right| = \frac{|\overline{z_1} - \overline{z_2}|}{|\overline{z_1 z_2}|} = \frac{|z_1 - z_2|}{|z_1 z_2|}. \tag{4.22}$$

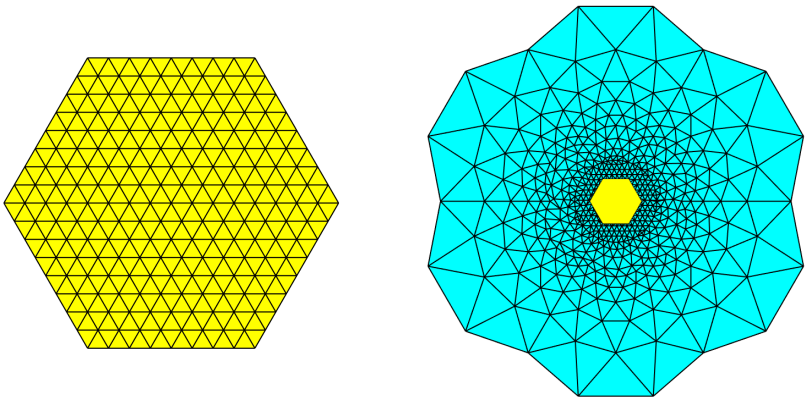

Figure 5. *Illustration of our algorithm for constructing the sea. Left: the initial flattening map. We put it inside the unit circle and fill up the gap with uniformly distributed points, and then reflect the entire region along the circle to construct the sea. Right: the sea constructed (in cyan) and the initial flattening map (in yellow).*

This implies that the edges formed under the reflection are short near the unit disk boundary, and get long further away. Hence, the outermost region of the sea, which stays far away from the region of interest, consists of the coarsest triangulations. By contrast, the innermost region of the sea closest to the unit circle has the densest triangulations. This natural transition of mesh sparsity of the sea helps reduce the number of points needed for the subsequent computation without affecting the accuracy of the result.

Another advantage of our construction is the improvement on the shape of the sea. In GN, a rectangular sea is used for the finite difference framework. The four corner regions are usually unimportant for the subsequent deformation, and hence a large amount of spaces and computational efforts are wasted. By contrast, our reflection-based framework can easily overcome the above drawback. In our construction of the sea, the reflection together with the truncation produces a sea with a more regular shape. This utilizes the use of every point at the sea and prevents any redundant computations.

4.3. Iterative scheme for producing density-equalizing maps. Given any simply connected open triangle mesh, the curvature-based flattening method produces a flattened map in $\mathbb{R}^2$. Suppose we are given a population on each triangle element of the mesh. Define the density $\rho^{\mathcal{F}}$ on each triangle element of the flattened map by $\frac{\text{Population}}{\text{Area of the triangle}}$. Then we interpolate $\rho^{\mathcal{F}}$ on vertices and develop an iterative scheme for deforming the flattened map using density diffusion. Our scheme is based on a finite element formulation.

To solve the diffusion equation on triangle meshes, one important issue is to discretize the Laplacian. Let $u : \widetilde{\mathcal{V}} \to \mathbb{R}$ be a function. To compute the Laplacian of u at every vertex i, we use the discrete finite-element Laplacian [73]

$$\Delta u(i) = -\frac{1}{2A(i)} \sum_{j \in \mathcal{N}(i)} (\cot \alpha_{ij} + \cot \beta_{ij}) \, (u(i) - u(j)), \tag{4.23}$$

where $\mathcal{N}(i)$ is the 1-ring vertex neighborhood of i, α_{ij} and β_{ij} are the two angles opposite to

the edge $[i, j]$, and $A(i)$ is the vertex area of the vertex i. More specifically,

$$A(i) = \frac{1}{3} \sum_{T \in \mathcal{N}^{\widetilde{\mathcal{F}}}(i)} \text{Area}(T), \tag{4.24}$$

where $\mathcal{N}^{\widetilde{\mathcal{F}}}(i)$ is the 1-ring face neighborhood of i. It is easy to observe that

$$\sum_{i \in \widetilde{\mathcal{V}}} A(i) = \sum_{T \in \widetilde{\mathcal{F}}} \text{Area}(T). \tag{4.25}$$

This shows that the vertex area is a good discretization of the total surface area at the vertex set. The graphical illustrations of α_{ij}, β_{ij} and $A(i)$ are given in Figure 3 (middle and right). It is noteworthy that the sea we constructed has an additional purpose of complementing the use of the FEM Laplacian (4.23), which assumes the natural boundary condition $\nabla u \cdot \mathbf{n} = 0$ in its derivation. The use of the sea ensures that the region of interest is not restricted by this natural boundary condition and hence can deform freely. An outline of the derivation of (4.23) is provided in the appendix (see Appendix B).

Note that the density ρ is originally defined on the triangle faces, while the above Laplacian works on vertices. To handle this discrepancy, we consider interpolating $\rho^{\mathcal{F}}$ at vertices. Note that for every vertex $v \in \mathcal{V}$, $\rho^{\mathcal{V}}(v)$ should only depend on the value of $\rho^{\mathcal{F}}$ within its 1-ring face neighborhood. This property is related to the Whitney 2-forms [68], which are piecewise-constant functions supported on triangle elements:

$$\phi_T^W(x) = \begin{cases} \frac{1}{\text{Area}(T)} & \text{if } x \text{ lies on } T, \\ 0 & \text{otherwise.} \end{cases} \tag{4.26}$$

Note that the function value at a vertex should depend on the values within its 1-ring face neighborhood. Any face-valued function $f : \mathcal{F} \to \mathbb{R}$ can be interpolated at all vertices $v \in \mathcal{V}$ by

$$f(v) = \frac{\sum_{T \in \mathcal{F}} \int_T f(T) \phi_T^W(v) dA}{\sum_{T \in \mathcal{F}} \int_T \phi_T^W(v) dA} = \frac{\sum_{T \in \mathcal{N}^{\mathcal{F}}(v)} \frac{f(T)}{\text{Area}(T)} \text{Area}(T)}{\sum_{T \in \mathcal{N}^{\mathcal{F}}(v)} \frac{1}{\text{Area}(T)} \text{Area}(T)} = \frac{\sum_{T \in \mathcal{N}^{\mathcal{F}}(v)} f(T)}{|\mathcal{N}^{\mathcal{F}}(v)|}. \tag{4.27}$$

The above interpolation result only depends on the mesh structure but not the geometry. In our case of interpolating ρ, as ρ is related to the deformation of face area, it is desirable to emphasize the weight of different faces in the interpolation. We consider the modified version of Whitney 2-forms,

$$\widetilde{\phi_T^W}(x) = \begin{cases} 1 & \text{if } x \text{ lies on } T, \\ 0 & \text{otherwise,} \end{cases} \tag{4.28}$$

which gives the desired interpolation

$$\rho^{\mathcal{V}}(v) = \frac{\sum_{T \in \mathcal{F}} \int_T \rho^{\mathcal{F}}(T) \widetilde{\phi_T^W}(v) dA}{\sum_{T \in \mathcal{F}} \int_T \widetilde{\phi_T^W}(v) dA} = \frac{\sum_{T \in \mathcal{N}^{\mathcal{F}}(v)} \rho^{\mathcal{F}}(T) \text{Area}(T)}{\sum_{T \in \mathcal{N}^{\mathcal{F}}(v)} \text{Area}(T)}. \tag{4.29}$$

If we regard $\rho^{\mathcal{V}}$ as a $|\mathcal{V}| \times 1$ matrix and $\rho^{\mathcal{F}}$ as a $|\mathcal{F}| \times 1$ matrix, the above formula can be represented as a matrix multiplication

$$\rho^{\mathcal{V}} := W^{\mathcal{F}\mathcal{V}} \rho^{\mathcal{F}}, \tag{4.30}$$

where $W^{\mathcal{F}\mathcal{V}}$ is a $|\mathcal{V}| \times |\mathcal{F}|$ sparse matrix

$$W^{\mathcal{F}\mathcal{V}} := \begin{pmatrix} W_{1,:}/\|W_{1,:}\|_1 \\ W_{2,:}/\|W_{2,:}\|_1 \\ \vdots \\ W_{|\mathcal{V}|,:}/\|W_{|\mathcal{V}|,:}\|_1 \end{pmatrix}, \tag{4.31}$$

with

$$W_{ij} = \begin{cases} \text{Area}(T_j) & \text{if the } j\text{th triangle } T_j \text{ contains the } i\text{th vertex,} \\ 0 & \text{otherwise.} \end{cases}$$

After computing the density $\rho^{\mathcal{V}}$ at the initial flattening map, we can extend $\rho^{\mathcal{V}}$ to the adaptive sea surrounding the map. As suggested in GN, the density at the sea should equal the mean density at the initial map. Therefore, for every vertex v' at the sea, we set

$$\rho^{\mathcal{V}}(v') = \text{mean}_v \rho^{\mathcal{V}}(v). \tag{4.32}$$

It is noteworthy that the density at the boundary of the original flattening map may be blurred by the sea if we perform the above-mentioned interpolation with the sea included. Therefore, it is more desirable to perform the interpolation and obtain $\rho^{\mathcal{V}}$ at the initial flattening map first, and then set the density at the sea nodes. In the following discussions, by an abuse of notation, we continue using the notation $\mathcal{V}$, $\mathcal{F}$ without tilde whenever referring to the discrete mesh structure including the sea.

With the density interpolated at all vertices, we use the following semidiscrete backward Euler method for solving the diffusion equation (3.3):

$$\frac{\rho_n^{\mathcal{V}} - \rho_{n-1}^{\mathcal{V}}}{\delta t} = \Delta_{n-1} \rho_n^{\mathcal{V}} \Longleftrightarrow \rho_n^{\mathcal{V}} = (I - \delta t \Delta_{n-1})^{-1} \rho_{n-1}^{\mathcal{V}}. \tag{4.33}$$

Here $\rho_n^{\mathcal{V}}$ is the value of $\rho^{\mathcal{V}}$ at the nth iteration, Δ_n is the FEM Laplacian of the deformed map $\mathbf{r}_n$, and δt is the time step for the iterations. In the discrete case, note that Δ_n can be represented as $\Delta_n = -A_n^{-1} L_n$, where A_n is a $|\mathcal{V}| \times |\mathcal{V}|$ diagonal matrix containing the vertex area of each vertex, and L_n is a $|\mathcal{V}| \times |\mathcal{V}|$ symmetric sparse matrix representing the cotangent components in (4.23). Hence, the above equation is equivalent to

$$\rho_n^{\mathcal{V}} = (A_{n-1} + \delta t L_{n-1})^{-1} \left(A_{n-1} \rho_{n-1}^{\mathcal{V}} \right). \tag{4.34}$$

The above semidiscrete backward Euler method is unconditionally stable. The matrix $A_{n-1} + \delta t L_{n-1}$ is a symmetric sparse matrix, and hence (4.34) can be efficiently solved by numerical solvers.

After discretizing the diffusion equation, we consider the production of the induced vector field. We first need to discretize the gradient operator ∇. Let $(\nabla \rho)_n^{\mathcal{F}}(T)$ be the face-based

discretization defined on every triangle element $T = [i, j, k]$ at the nth iteration. Denote the three directed edges of T in the form of vectors by $e_{ij} = [i, j], e_{jk} = [j, k], e_{ki} = [k, i]$, and denote the unit normal vector of T by N. A formula of $(\nabla\rho)_n^{\mathcal{F}}(T)$ can be derived using Whitney 0-forms [68], which are hat functions on the vertices:

$$\phi_i^W(p) = \begin{cases} 1 & \text{if } p = i, \\ 0 & \text{if } p \text{ is outside } N^{\mathcal{F}}(i), \\ \text{affine} & \text{if } p \text{ lies on } N^{\mathcal{F}}(i). \end{cases} \tag{4.35}$$

Give any function f defined on vertices, f can be interpolated at any point x lying on the triangle face $T = [i, j, k]$ by

$$f(x) = f_i\phi_i^W(x) + f_j\phi_j^W(x) + f_k\phi_k^W(x), \tag{4.36}$$

where f_i, f_j, f_k are the values of f at the vertices i, j, k. Using the property that $\nabla\phi_i^W = \frac{N\times e_{jk}}{2\text{Area}(T)}$ [68], we obtain

$$\begin{aligned} (\nabla\rho)_n^{\mathcal{F}}(T) &= \nabla\left(\rho_n^{\mathcal{V}}(i)\phi_i^W + \rho_n^{\mathcal{V}}(j)\phi_j^W + \rho_n^{\mathcal{V}}(k)\phi_k^W\right) \\ &= \rho_n^{\mathcal{V}}(i)\nabla\phi_i^W + \rho_n^{\mathcal{V}}(j)\nabla\phi_j^W + \rho_n^{\mathcal{V}}(k)\nabla\phi_k^W \\ &= \frac{1}{2\text{Area}(T)}N \times \left(\rho_n^{\mathcal{V}}(i)e_{jk} + \rho_n^{\mathcal{V}}(j)e_{ki} + \rho_n^{\mathcal{V}}(k)e_{ij}\right). \end{aligned} \tag{4.37}$$

This gives us an accurate approximation of the gradient $(\nabla\rho)_n^{\mathcal{F}}$ on triangulated surfaces. Then we can again use Whitney 2-forms (4.29) to obtain $(\nabla\rho)_n^{\mathcal{V}}$ on the vertices.

With all differential operators discretized, we are now ready to introduce our iterative scheme for computing density-equalizing maps. In each iteration, we update the density by solving (4.34) and compute the induced gradient $(\nabla\rho)_n^{\mathcal{V}}$ based on the above-mentioned procedures. Then we deform the map by

$$\mathbf{r}_n = \mathbf{r}_{n-1} - \delta t(\nabla\rho)_n^{\mathcal{V}}/\rho_n^{\mathcal{V}}. \tag{4.38}$$

For the stopping criterion, we consider the quantity $\text{sd}(\rho_n^{\mathcal{V}})/\text{mean}(\rho_n^{\mathcal{V}})$. Note that the standard deviation $\text{sd}(\rho_n^{\mathcal{V}})$ measures the dispersion of the updated density $\rho_n^{\mathcal{V}}$, and we normalize it using $\text{mean}(\rho_n^{\mathcal{V}})$ to remove the effect of arbitrary scaling of $\rho_n^{\mathcal{V}}$. Also, it is easy to note that $\text{sd}(\rho_n^{\mathcal{V}})/\text{mean}(\rho_n^{\mathcal{V}}) = 0$ if and only if the density is completely equalized. Hence, this normalized quantity can be used for determining the convergence of the iterative algorithm. Finally, we rescale the mapping result so that the total area of S is preserved under our density-equalizing mapping algorithm.

We remark that the step size δt affects the convergence rate of the algorithm. By dimensional analysis of the diffusion equation (3.3), an appropriate dimension of δt would be L^2. Also, note that δt should be independent of the magnitude of ρ. Therefore, a reasonable choice of δt is

$$\delta t = \min\left\{\frac{\min(\rho_0^{\mathcal{V}})}{\text{mean}(\rho_0^{\mathcal{V}})}, \frac{\text{mean}(\rho_0^{\mathcal{V}})}{\max(\rho_0^{\mathcal{V}})}\right\} \times \text{Area}(S). \tag{4.39}$$

The first term is a dimensionless quantity that takes extreme relative density ratios into account, and the second term is a natural quantity with dimension L^2. One can also rescale

Algorithm 5: Density-equalizing map (DEM) for simply connected open surfaces.

Input: A simply connected open triangulated surface S, a population on each triangle, and a stopping parameter ϵ.

Output: A density-equalizing flattening map $f : S \to \mathbb{R}^2$.

1 **if** *S is planar* **then**
2 Set $\mathbf{r}_0 = S$; ;
3 **else**
4 Compute a curvature-based flattening map $\phi : S \to \mathbb{C}$ using Algorithm 2 or Algorithm 3. Denote $\mathbf{r}_0 = \phi(S)$;
5 Define the density $\rho_0^{\mathcal{F}} = \frac{\text{Given population}}{\text{Area of the triangle}}$ on each triangle of $\mathbf{r}_0$;
6 Compute $\rho_0^{\mathcal{V}} = W^{\mathcal{FV}} \rho_0^{\mathcal{F}}$;
7 Update $\mathbf{r}_0$ with an adaptive sea constructed using Algorithm 4;
8 Extend $\rho_0^{\mathcal{V}}$ to the whole domain by setting $\rho_0^{\mathcal{V}}$ at the sea to be the mean of the original $\rho_0^{\mathcal{V}}$;
9 Set $\delta t = \min\left\{ \frac{\min(\rho_0^{\mathcal{V}})}{\text{mean}(\rho_0^{\mathcal{V}})}, \frac{\text{mean}(\rho_0^{\mathcal{V}})}{\max(\rho_0^{\mathcal{V}})} \right\} \times \text{Area}(S)$;
10 Set $n = 0$;
11 **repeat**
12 Update $n = n + 1$;
13 Solve $\rho_n^{\mathcal{V}} = (A_{n-1} + \delta t L_{n-1})^{-1} \left(A_{n-1} \rho_{n-1}^{\mathcal{V}} \right)$;
14 Compute the face-based discrete gradient $(\nabla\rho)_n^{\mathcal{F}}(T) = \frac{1}{2\text{Area}(T)} N \times \left(\rho_n^{\mathcal{V}}(i) e_{jk} + \rho_n^{\mathcal{V}}(j) e_{ki} + \rho_n^{\mathcal{V}}(k) e_{ij} \right)$;
15 Compute $(\nabla\rho)_n^{\mathcal{V}} = W^{\mathcal{FV}} (\nabla\rho)_n^{\mathcal{F}}$;
16 Update $\mathbf{r}_n = \mathbf{r}_{n-1} - \delta t (\nabla\rho)_n^{\mathcal{V}} / \rho_n^{\mathcal{V}}$;
17 **until** $\frac{sd(\rho_n^{\mathcal{V}})}{mean(\rho_n^{\mathcal{V}})} < \epsilon$;
18 Obtain $f(S) = \mathbf{r}_n \times \frac{\text{Area}(\mathbf{r}_0)}{\text{Area}(\mathbf{r}_n)}$;

δt by a constant multiple. Algorithm 5 summarizes our proposed method for producing density-equalizing maps (DEMs) of simply connected open surfaces.

While our DEM algorithm naturally deforms the entire shape to achieve a density-equalizing map, in some situations it may be desirable to have a flattening map with a prescribed simple boundary shape such as rectangle or disk. Our density-equalizing mapping algorithm can be easily modified to handle such situations by the following approach. First, we replace the initialization (lines 1–4) in Algorithm 5 by an initial map with the desired boundary shape, such as a disk Tutte map or a rectangular Tutte map. Then, as it is desired that the boundary shape remain unchanged and the region outside the mesh is not of our interest, we can skip the construction of the sea (lines 7–8).

Theoretically, the deformation of the map during the density-equalizing process is guided by the density gradient field $\nabla\rho$. To ensure that the overall boundary shape remains unchanged,

at each iteration we can enforce a Neumann boundary condition

$$(\nabla\rho)\cdot\mathbf{n}=0. \tag{4.40}$$

In fact, this condition has been implicitly enforced in the derivation of (4.23) (see Appendix B). Therefore, in theory we can obtain a density-equalizing map with the desired shape without making any change in the iterative scheme.

However, in the discrete case, there is often a numerical error in approximating the density gradient, and hence $\nabla\rho$ at the boundary may have a small nonzero normal component, causing the boundary to undergo a small change in the overall shape. To correct that, we can simply add a step of projecting all boundary vertices onto the prescribed shape at the end of each iteration. This ensures that the boundary vertices stay on the prescribed boundary shape throughout the density-equalizing process, while having the freedom to slide along it to achieve density equalization. With the implicit assumption of $(\nabla\rho)\cdot\mathbf{n}=0$ in the cotangent Laplacian formulation, this extra projection step will not lead to a notable discrepancy between the density field and the shape deformation. Algorithm 6 summarizes this modified version of Algorithm 5 for computing shape-prescribed density-equalizing maps.

Algorithm 6: Shape-prescribed density-equalizing map for simply connected open surfaces.

Input: A simply connected open triangulated surface S, a prescribed boundary shape, a population on each triangle, and a stopping parameter ϵ.

Output: A shape-prescribed density-equalizing flattening map $f: S\to\mathbb{R}^2$.

Compute an initial map $\phi: S\to\mathbb{C}$ with the prescribed boundary shape. Denote $\mathbf{r}_0=\phi(S)$;

Define the density $\rho_0^{\mathcal{F}}=\frac{\text{Given population}}{\text{Area of the triangle}}$ on each triangle of $\mathbf{r}_0$;

Compute $\rho_0^{\mathcal{V}}=W^{\mathcal{FV}}\rho_0^{\mathcal{F}}$;

Set $\delta t=\min\left\{\frac{\min(\rho_0^{\mathcal{V}})}{\text{mean}(\rho_0^{\mathcal{V}})},\frac{\text{mean}(\rho_0^{\mathcal{V}})}{\max(\rho_0^{\mathcal{V}})}\right\}\times\text{Area}(S)$;

Set $n=0$;

repeat

  Update $n=n+1$;

  Solve $\rho_n^{\mathcal{V}}=(A_{n-1}+\delta tL_{n-1})^{-1}\left(A_{n-1}\rho_{n-1}^{\mathcal{V}}\right)$;

  Compute the face-based discrete gradient $(\nabla\rho)_n^{\mathcal{F}}(T)=\frac{1}{2\text{Area}(T)}N\times\left(\rho_n^{\mathcal{V}}(i)e_{jk}+\rho_n^{\mathcal{V}}(j)e_{ki}+\rho_n^{\mathcal{V}}(k)e_{ij}\right)$;

  Compute $(\nabla\rho)_n^{\mathcal{V}}=W^{\mathcal{FV}}(\nabla\rho)_n^{\mathcal{F}}$;

  Update $\mathbf{r}_n=\mathbf{r}_{n-1}-\delta t(\nabla\rho)_n^{\mathcal{V}}/\rho_n^{\mathcal{V}}$;

  Project all boundary vertices onto the prescribed boundary shape;

until $\frac{sd(\rho_n^{\mathcal{V}})}{mean(\rho_n^{\mathcal{V}})}<\epsilon$;

Obtain $f(S)=\mathbf{r}_n\times\frac{\text{Area}(\mathbf{r}_0)}{\text{Area}(\mathbf{r}_n)}$;

4.4. The choice of population and its effects. Before ending this section, we discuss the choice of the initial population and its effect on the final result obtained by our algorithm.

Some choices and the corresponding effects are listed below:

(i) If we set a relatively high population at a certain region of the input surface, the population will cause an expansion during the density equalization. The region will be magnified in the final density-equalizing mapping result.

(ii) Similarly, if we set a relatively low population at a certain region of the input surface, the region will shrink in the final density-equalizing mapping result.

(iii) If we set the population to be the area of every triangle element of the input surface, the resulting density-equalizing map will be an area-preserving planar parameterization of the input surface, as we have

$$\frac{\text{Initial area}}{\text{Final area}} = \frac{\text{Given population}}{\text{Final area}} = \text{Density} = \text{Constant}. \tag{4.41}$$

Examples are given in section 5 to illustrate the effect of different input populations.

5. Experimental results. In this section, we demonstrate the effectiveness of our proposed algorithm using various experiments. Our algorithms are implemented in MATLAB (see M112479_01.zip [local/web 209KB]). The linear systems in our algorithm are solved using the backslash operator in MATLAB. The Fast InPolygon detection MEX custom MATLAB function [74] is used in the step of constructing the sea. All experiments are performed on a PC with an Intel i7-6700K CPU and 16GB RAM. All surfaces are discretized in the form of triangle meshes. In all experiments, the stopping parameter ϵ of our algorithms is set to be 10^{-3}. Some of the surface meshes are adapted from the AIM@SHAPE Shape Repository [75].

5.1. Examples of density-equalizing maps produced by our DEM algorithm. We begin with two synthetic examples of regular polygons on $\mathbb{R}^2$. Figures 6 and 7 show, respectively, a square and a hexagon with a given population on every triangle element, along with the density-equalizing results obtained by our proposed DEM algorithm. In both examples the final densities $\frac{\text{Given population}}{\text{Final area}}$ highly concentrate at 1, meaning that the densities are well equalized. Also, the plots of the quantity $\frac{\text{sd}(\rho_n^{\mathcal{V}})}{\text{mean}(\rho_n^{\mathcal{V}})}$ versus the number of iterations show that the iterative scheme converges rapidly.

We then consider a synthetic example of a surface in $\mathbb{R}^3$ with Gaussian shape. The domain of the shape is $[0,1] \times [0,1]$ and the population is set to be $2.2 - |x| - |y|$, where (x, y) are the x- and y-coordinates of the centroid of each triangle element. Algorithm 2 is used for the initialization of the density-equalization algorithm. Figure 8 shows the initial surface and the mapping result obtained by our DEM algorithm. The plots indicate that the density is well equalized by our algorithm.

It is noteworthy that the curvature-based initial flattening map and the final density-equalizing map can be significantly different in shape, as illustrated in Figure 9. In particular, the convex initializations can be deformed to nonconvex shapes under our DEM algorithm.

We consider another synthetic example of a surface with multiple peaks in $\mathbb{R}^3$. This time we set the population as the area of each triangle element on the initial surface. In other words, our proposed DEM algorithm should result in an area-preserving flattening map. Again, Algorithm 2 is used for the initialization of the density-equalization algorithm. Figure 10 shows the initial surface and the mapping result obtained by our density-equalizing mapping algorithm. The flattening map effectively preserves the area ratios.

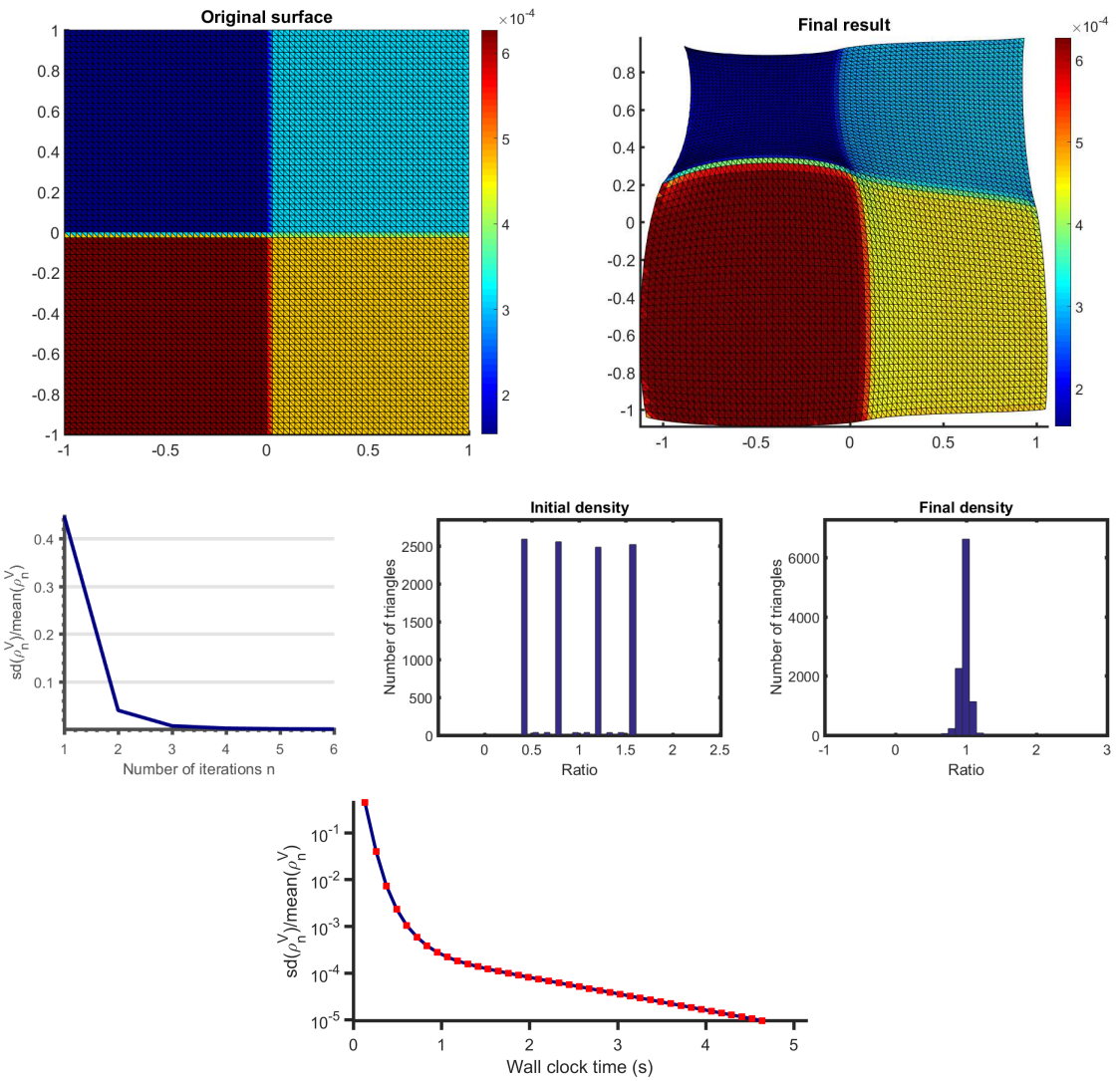

Figure 6. *Density equalization on a square. Top row (left to right): the initial shape colored with a given population distribution, and the density-equalizing map colored with the final area of each triangle element. Middle row (left to right): the values of* $\frac{sd(\rho_n^{\mathcal{V}})}{mean(\rho_n^{\mathcal{V}})}$, *the histogram of the initial density* $\frac{Given\ population}{Initial\ area}$ *on each triangle element, and the histogram of the final density* $\frac{Given\ population}{Final\ area}$ *on each triangle element. See Table 1 for statistics. Bottom: an additional semilog plot of* $\frac{sd(\rho_n^{\mathcal{V}})}{mean(\rho_n^{\mathcal{V}})}$ *versus time with a stronger threshold of* $\epsilon = 10^{-5}$, *which shows rapid convergence.*

Now consider computing the area-preserving mapping for a real surface mesh of a lion face in $\mathbb{R}^3$ using our DEM algorithm. Again, we set the population as the area of each triangle element on the initial surface for achieving an area-preserving parameterization. Algorithm 3 is used for the initialization step of our density-equalizing mapping algorithm. Figure 11 shows the initial surface and the mapping result obtained by our density-equalizing mapping

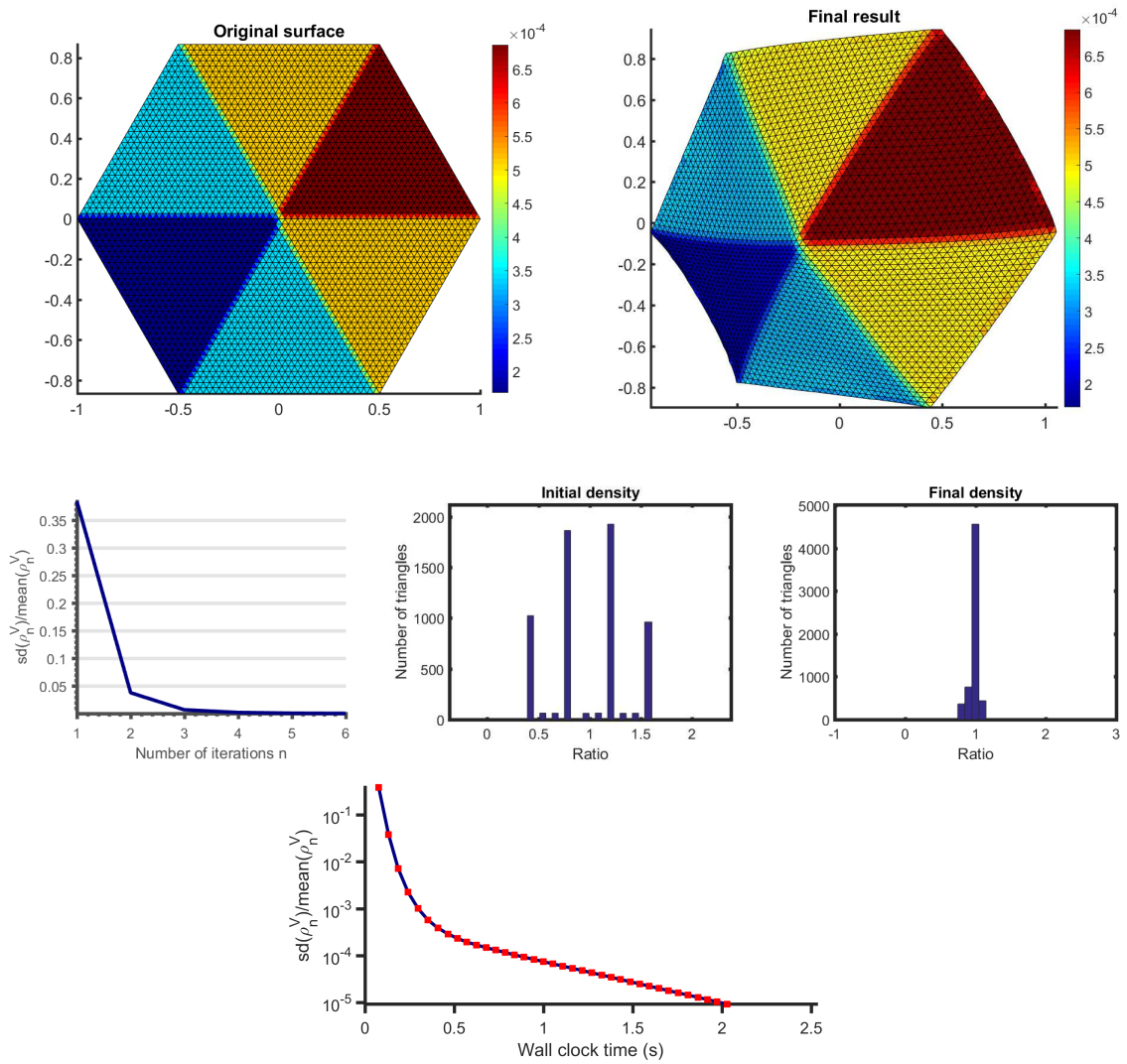

Figure 7. *Density equalization on a hexagon. Top row (left to right): the initial shape colored with a given population distribution, and the density-equalizing map colored with the final area of each triangle element. Middle row (left to right): the values of* $\frac{sd(\rho_n^{\mathcal{V}})}{mean(\rho_n^{\mathcal{V}})}$*, the histogram of the initial density* $\frac{Given\ population}{Initial\ area}$ *on each triangle element, and the histogram of the final density* $\frac{Given\ population}{Final\ area}$ *on each triangle element. See Table 1 for statistics. Bottom: an additional semilog plot of* $\frac{sd(\rho_n^{\mathcal{V}})}{mean(\rho_n^{\mathcal{V}})}$ *versus time with a stronger threshold of* $\epsilon = 10^{-5}$*, which shows rapid convergence.*

algorithm. For better visualization, we color the meshes with the mean curvature of the input lion face. The locally authalic initialization does not preserve the global area ratio; in particular, the nose of the lion is shrunk. By contrast, the final density-equalizing flattening map effectively preserves the area ratios.

In addition, our algorithm can produce density-equalizing flattening maps with different

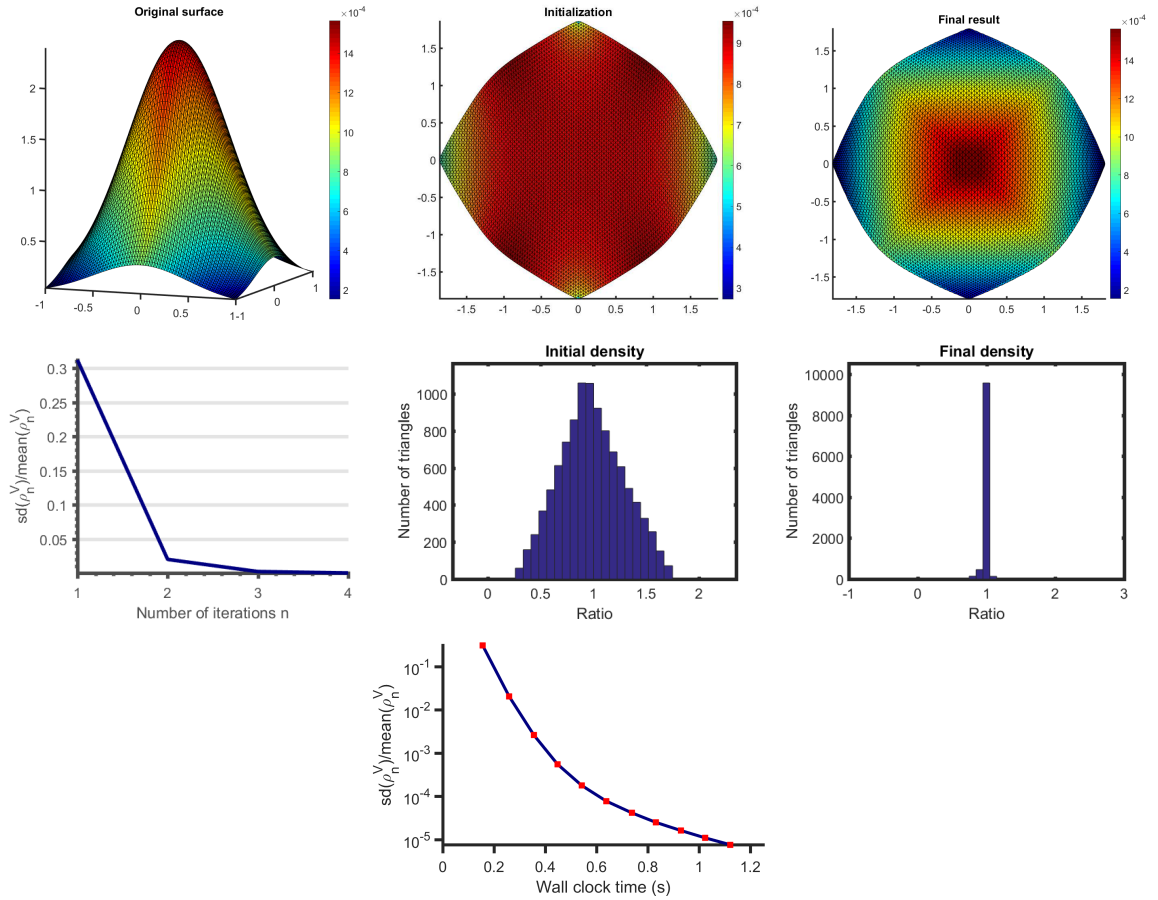

Figure 8. *Density-equalizing map for a surface in* $\mathbb{R}^3$ *with Gaussian shape. Top row (left to right): the initial shape colored with a given population distribution, the curvature-based Tutte flattening initialization colored with the area of each flattened triangle element, and the final density-equalizing map colored with the final area of each triangle element. Middle row (left to right): the values of* $\frac{sd(\rho_n^{\mathcal{V}})}{mean(\rho_n^{\mathcal{V}})}$*, the histogram of the density* $\frac{Given\ population}{Initial\ flattened\ area}$ *on each flattened triangle element after the Tutte flattening initialization, and the histogram of the density* $\frac{Given\ population}{Final\ area}$ *on each triangle element of the final result. See Table 1 for statistics. Bottom: an additional semilog plot of* $\frac{sd(\rho_n^{\mathcal{V}})}{mean(\rho_n^{\mathcal{V}})}$ *versus time with a stronger threshold of* $\epsilon = 10^{-5}$*, which shows rapid convergence.*

effects by changing the input population. Figure 12 shows two examples with different input populations. For the Niccolò da Uzzano model, we set the input population to be the area of each triangle element on the mesh except the eyes, and the population at the eyes to be 2 times the area of the triangles there. For the Max Planck model, we set the input population to be the area of each triangle element on the mesh except the nose, and the population at the nose to be 1.5 times the area of the triangles there. The resulting density-equalizing maps have the eyes and nose magnified, respectively.

As discussed in section 4.3, our DEM algorithm can be modified as shape-prescribed DEM to achieve a prescribed target shape. Four simple examples are shown in Figure 13. First, we

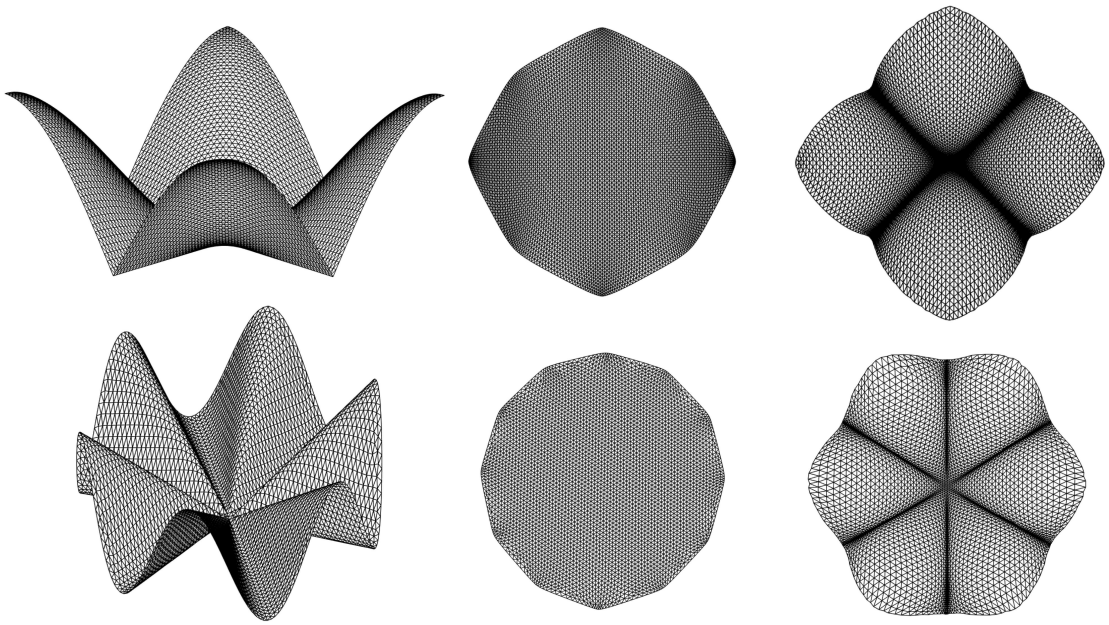

Figure 9. *Two more examples of density-equalizing maps. In these examples, we set the population according to the height of different parts of the surfaces, aiming to achieve an expansion at the peaks. Left: the original surfaces. Middle: the curvature-based initial flattening maps. Right: the density-equalizing maps. Note that the convex boundaries in the initializations can become nonconvex under DEM.*

consider the square example in Figure 6 again and compute two density-equalizing maps of it to a square and a rectangle with aspect ratio $2:3$, respectively. For the initial maps, the Tutte method is used. The boundary vertices have moved along the shape boundary in the resulting maps. We also consider the hexagon example in Figure 7 again and compute two density-equalizing maps of it to, respectively, a circle and an ellipse with aspect ratio $2:1$. For the initial map, the disk/ellipse Tutte method with arclength parameterized boundary constraint is used. Again, the boundary vertices move along the curved boundary. This demonstrates the effectiveness of our shape-prescribed DEM. In addition, similarly to the ordinary DEM, area-preserving disk/rectangular parameterization can be easily achieved using our shape-prescribed DEM, as illustrated in Figure 14. We can also achieve the effects in Figure 12 with boundary shape prescribed, as shown in Figure 15.

We remark that for a surface with a highly convoluted boundary, directly computing the curvature-based flattening map and the density-equalization may be difficult. Nevertheless, we can extend our sea approach for handling this situation. By prescribing a sea with a simpler shape around such a surface, we can easily flatten it and carry out the usual density-equalization procedure, achieving different desired effects. Figure 16 shows a simply connected open surface mimicking a space-filling curve on a torus. We define a population on the mesh with $\frac{\max(\text{population})}{\min(\text{population})} \approx 25$, aiming to produce a large deformation under density equalization. By prescribing a rectangular sea around the mesh on the torus, we can flatten it as a rectangle on the plane and compute the density-equalizing map. The mapping result shows that a

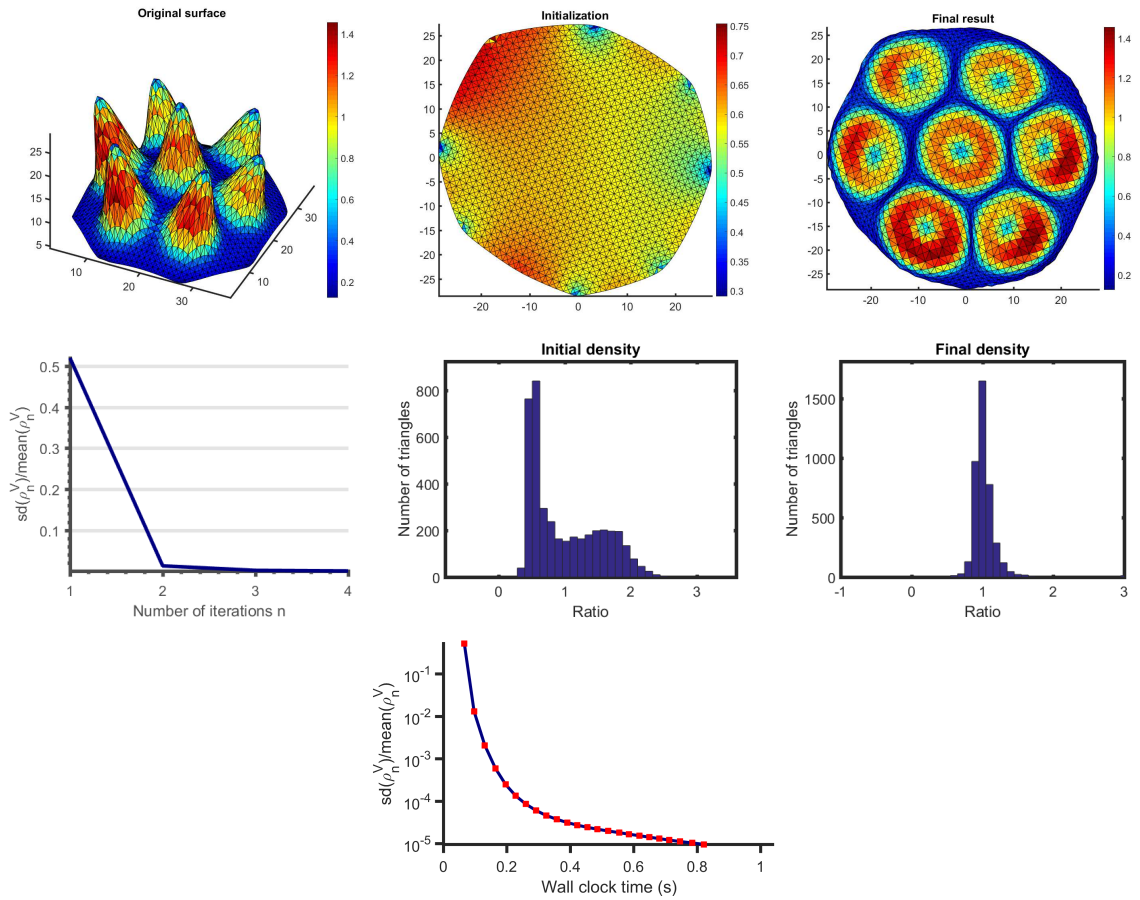

Figure 10. *Area-preserving parameterization of a surface with multiple peaks in* $\mathbb{R}^3$. *Top row (left to right): the initial shape colored with the initial area of each triangle element, the curvature-based Tutte flattening initialization colored with the area of each flattened triangle element, and the final density-equalizing map colored with the final area of each triangle element. Middle row (left to right): the values of* $\frac{sd(\rho_n^{\mathcal{V}})}{mean(\rho_n^{\mathcal{V}})}$, *the histogram of the density* $\frac{Initial\ area}{Initial\ flattened\ area}$ *on each flattened triangle element after the Tutte flattening initialization, and the histogram of the density* $\frac{Initial\ area}{Final\ area}$ *on each triangle element of the final result. See Table 1 for statistics. Bottom: an additional semilog plot of* $\frac{sd(\rho_n^{\mathcal{V}})}{mean(\rho_n^{\mathcal{V}})}$ *versus time with a stronger threshold of* $\epsilon = 10^{-5}$, *which shows rapid convergence.*

desired effect is successfully achieved. It is also noteworthy that the presence of the sea at the gaps helps regularize the deformation and avoid global overlaps. The idea is that the density information at two geometrically close but topologically distant regions on the original mesh can be transmitted between each other via the sea directly without going through the complicated domain. Therefore, even if both regions are required to expand, they can coordinate with each other and find a nonoverlapping direction for the expansion.

5.2. Quantitative results of our algorithm. For a quantitative analysis, Table 1 lists the detailed statistics of the performance of our DEM algorithm on a number of simply connected

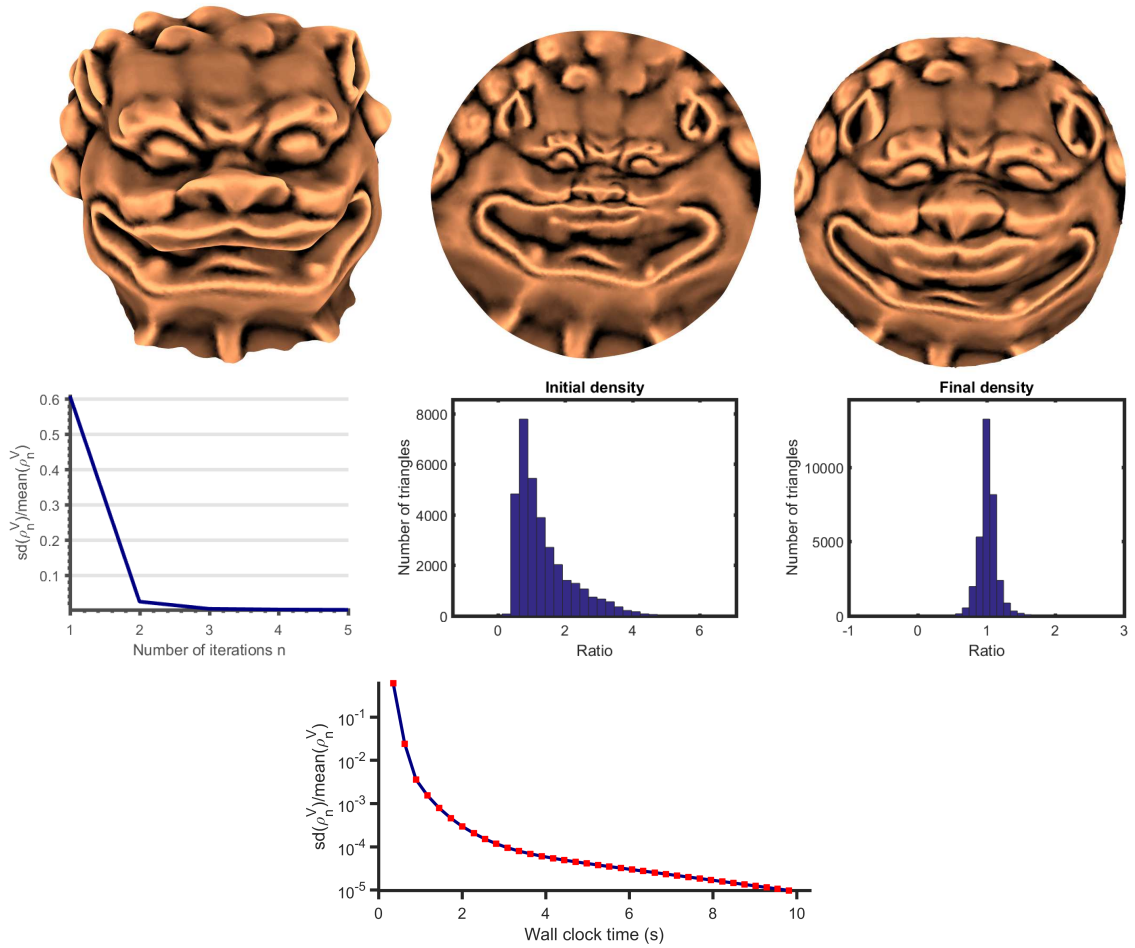

Figure 11. *Area-preserving parameterization of a lion face in* $\mathbb{R}^3$. *Top row (left to right): the initial shape, the curvature-based locally authalic flattening initialization, and the final density-equalizing map. Middle row (left to right): the values of* $\frac{sd(\rho_n^{\mathcal{V}})}{mean(\rho_n^{\mathcal{V}})}$, *the histogram of the density* $\frac{Initial\ area}{Initial\ flattened\ area}$ *on each flattened triangle element after the flattening initialization, and the histogram of the density* $\frac{Initial\ area}{Final\ area}$ *on each triangle element of the final result. See Table 1 for statistics. Bottom: an additional semilog plot of* $\frac{sd(\rho_n^{\mathcal{V}})}{mean(\rho_n^{\mathcal{V}})}$ *versus time with a stronger threshold of* $\epsilon = 10^{-5}$, *which shows rapid convergence.*

open meshes. From the time spent and the number of iterations needed, the convergence of our proposed algorithm is fast. Also, the median and the interquartile range of the density show that the density is well equalized under our algorithm. The experiments reflect the efficiency and accuracy of our proposed DEM algorithm.

Table 2 shows the performance of the shape-prescribed DEM algorithm. The shape-based prescribed DEM is reasonably accurate but not as accurate as DEM because of the extra shape constraints. As the shape-prescribed DEM does not include the sea, the computation of it is faster than DEM.

We then further analyze DEM in terms of the execution time. We use the Run and

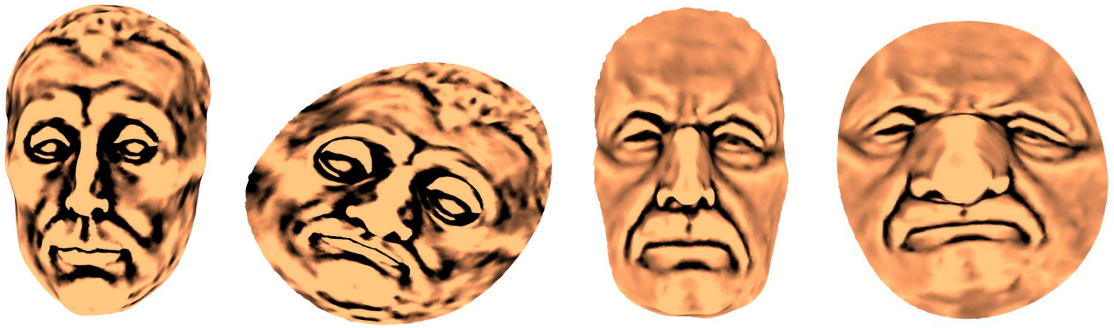

Figure 12. *Density-equalizing flattening maps with different effects obtained by our proposed DEM algorithm. Left: the Niccolò da Uzzano model and the density-equalizing flattening map with the eyes magnified. Right: the Max Planck model and the density-equalizing flattening map with the nose magnified. See Table 1 for statistics.*

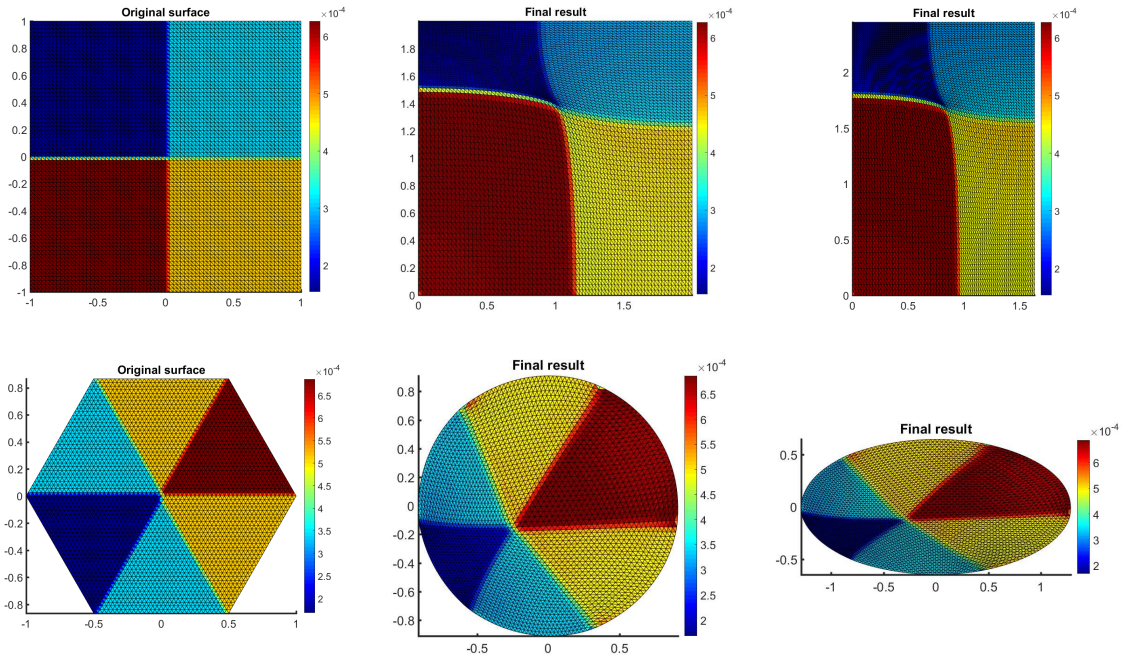

Figure 13. *Computing density-equalizing maps using our shape-prescribed DEM. Top row: the example in Figure 6 and two density-equalizing maps onto a square and a rectangle with aspect ratio 2 : 3. Bottom row: the example in Figure 7 and two density-equalizing maps onto a disk and an ellipse with aspect ratio 2 : 1. See Table 2 for statistics.*

Time built-in tool in MATLAB to measure the execution time of different parts of DEM. Table 3 shows the record for three examples. Both the curvature-based flattening map and the construction of sea are highly efficient and only account for around 10% of the total computation time. Also, although the Laplacian needs to be updated at every step of the density-equalizing process, each computation costs only around 0.1 seconds. Each iteration of the density-equalizing process typically requires less than 0.3 seconds for triangle meshes with 30k face elements.

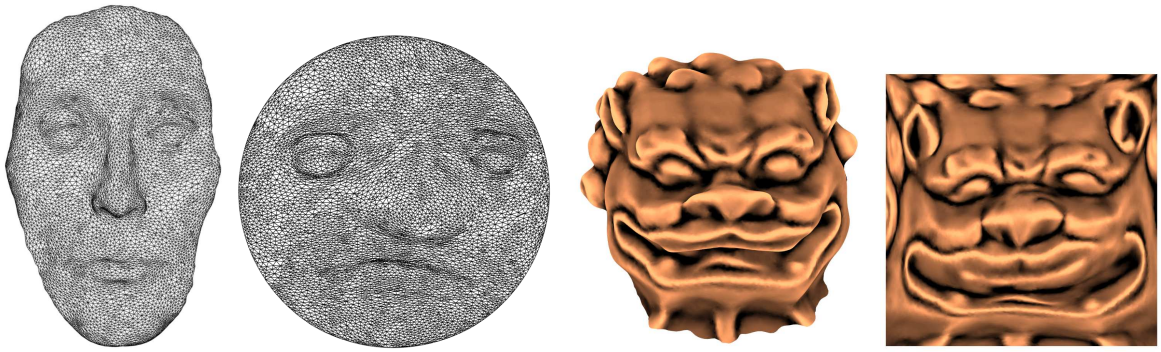

Figure 14. *Area-preserving parameterization via shape-prescribed DEM. Top left: the human face model and its area-preserving parameterization onto a disk. Top right: the lion model and its area-preserving parameterization onto a square. See Table 2 for statistics.*

Figure 15. *Achieving shape-prescribed density-equalizing flattening maps with the effects in Figure 12. Left: the Niccolò da Uzzano model and the disk density-equalizing map with the eyes magnified. Right: the Max Planck model and the rectangular (with aspect ratio 2 : 3) density-equalizing map with the nose magnified. See Table 2 for statistics.*

We are also interested in analyzing the difference in the performance of our algorithm and GN with the implementation available online [76]. Recall that GN works on finite difference grids. Therefore, for a fair comparison, we deploy the two methods on a 100×100 square grid $\{(x, y) \in \mathbb{Z}^2 : 0 \leq x, y \leq 99\}$ and compare the results. Each square is divided into two right-angled triangles in running our algorithm. Following the suggestion by GN, the dimension of the sea is set to be two times the linear extent of the square grid in running GN. Various initial populations are tested for the computation of density-equalizing maps. Figure 17 shows several density-equalizing mapping results produced by the two methods. The statistics of the experiments are recorded in Table 4. With the accuracy well preserved, our method demonstrates an improvement on the computational time by over 80% when compared to GN.

5.3. Comparison with other parameterization methods. We first visually compare our DEM algorithm with the various existing parameterization algorithms [19, 40, 48, 37]. Figure 18 shows the parameterization results, whereby the peaks are substantially shrunk for conformal

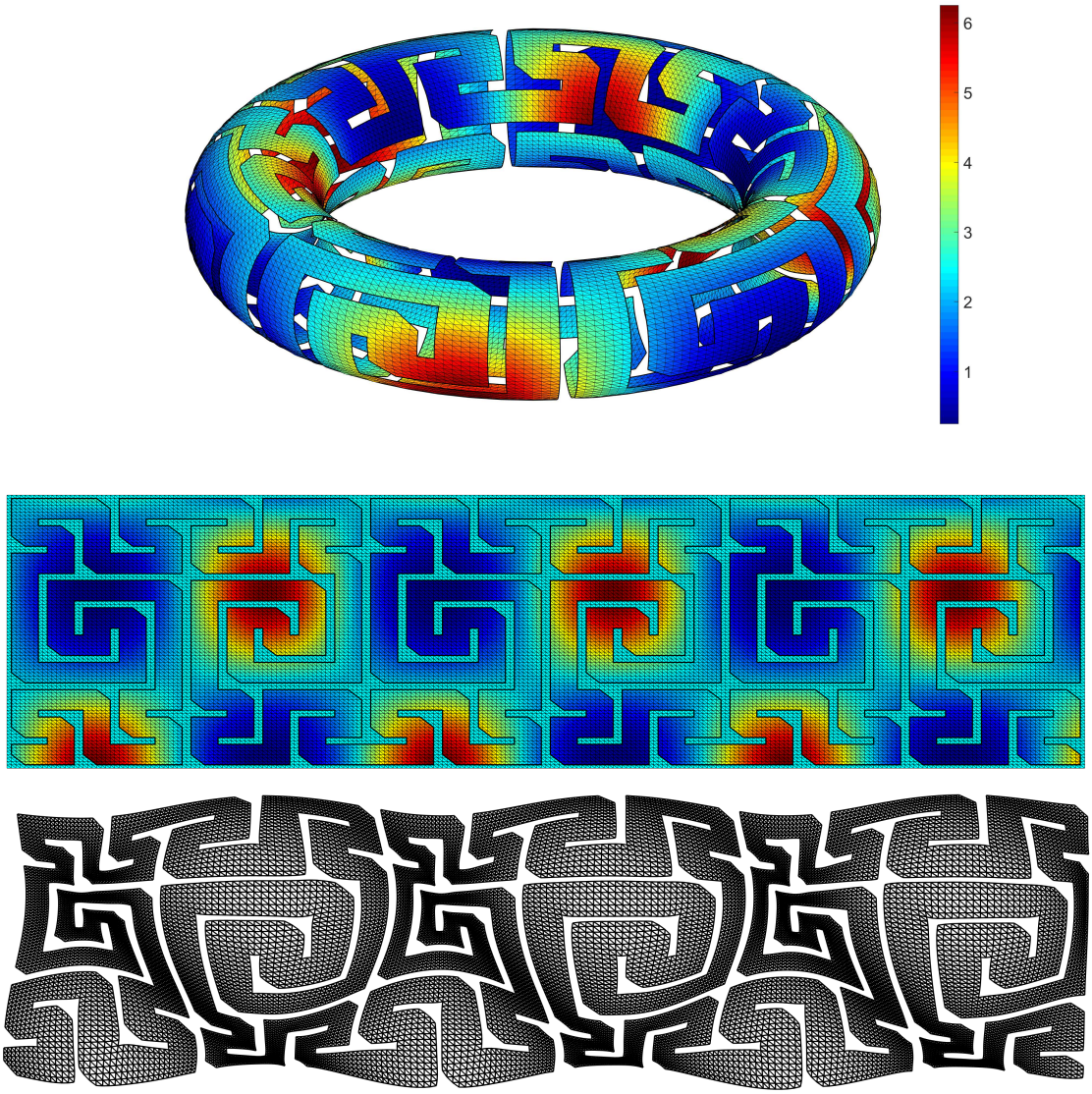

Figure 16. *Density-equalizing map for a simply connected open surface with a space-filling curve pattern on torus. Top: the mesh color coded with the input population. Middle: by prescribing a sea with a simpler shape around the surface, the entire surface can be flattened for the subsequent density-equalization step. Bottom: the density-equalizing mapping result.*

parameterizations, and the boundary of the free-boundary conformal parameterization is significantly different from that of the original surface. By contrast, the peaks are flattened without being shrunk under our proposed DEM algorithm, the optimal mass transport (OMT) map [40], and the scalable locally injective map (SLIM) [48]. Figure 19 shows more comparisons of our DEM with OMT and SLIM. DEM and OMT are more capable than SLIM in producing flattening maps that avoid squeezed regions (such as the peak of the Gaussian mesh and the

Table 1

The performance of our DEM algorithm. For each surface, we record the number of triangle elements, the time taken (in seconds) for the entire density-equalization algorithm (including the computation of initial map and the construction of sea), the number of iterations taken in the iterative scheme, and the median and interquartile range of the density defined on each triangle element by $\frac{\text{Given population}}{\text{Final area}}$.

Surface	No. of triangles	Time (s)	No. of iterations	Median of density	IQR of density
Square	10368	0.6983	6	1.0120	0.0705
Hexagon	6144	0.3123	6	1.0232	0.0549
Gaussian	10368	0.4088	4	1.0040	0.0309
Peaks	4108	0.1493	4	1.0018	0.1322
Lion	33369	1.4443	5	1.0169	0.1266
Niccolò da Uzzano	25900	2.0740	8	1.0247	0.0801
Max Planck	26452	2.4307	11	1.0203	0.0631
Human face	6912	0.7933	6	1.0069	0.0573
US Map (Romney)	46587	3.7946	3	1.0018	0.0145
US Map (Obama)	46587	3.7801	3	1.0004	0.0145
US Map (Trump)	46587	5.2797	4	1.0017	0.0176
US Map (Clinton)	46587	3.7643	3	1.0001	0.0244

Table 2

The performance of our shape-prescribed DEM algorithm. The density again refers to $\frac{\text{Given population}}{\text{Final area}}$.

Surface	No. of triangles	Target shape	Time (s)	No. of iterations	Median of density	IQR of density
Square	10368	Square	0.1891	5	1.0044	0.1097
Square	10368	Rectangle	0.2260	6	1.0042	0.1367
Hexagon	6144	Circle	0.1560	6	1.0011	0.0538
Hexagon	6144	Ellipse	0.2431	9	1.0053	0.0549
Peaks	4108	Circle	0.0615	4	0.9979	0.1362
Human face	6912	Circle	0.2642	5	1.0068	0.1125
Lion	33369	Square	0.8491	6	1.0109	0.1302
Niccolò da Uzzano	25900	Circle	1.3215	14	1.0054	0.0751
Max Planck	26452	Rectangle	1.2847	13	1.0164	0.0804

nose of the human face). Also, DEM and SLIM are more flexible in shape than OMT.

We then numerically compare our DEM algorithm with OMT and SLIM, in terms of the efficiency and accuracy, for computing area-preserving parameterizations. The implementations of both OMT and SLIM are provided by the authors [77, 78]. For a fair evaluation of the area-preserving property, we first rescale all parameterization results computed by the three algorithms, so that the total area of every parameterization result is the same as the original surface. Then we evaluate the area-preserving property of the three algorithms by considering

Table 3

Analyzing the execution time of DEM for three examples using the Run and Time tool in MATLAB. Note that the tool requires additional computation time, and so the total time for each example is slightly longer than that shown in Table 1.

Human face (No. of triangles = 6912)			
Function	**Call**	**Total time (s)**	**Percentage**
Curvature-based curve flattening	1	0.03	3.3%
Initial flattening map	1	0.05	5.6%
Interpolate $\rho^{\mathcal{V}}$	1	0.01	1.1%
Construction of sea	1	0.09	10.0%
Compute cotangent Laplacian L_n	5	0.22	24.4%
Solve diffusion equation	5	0.26	28.9%
Compute $(\nabla\rho)_n^{\mathcal{F}}$	5	0.04	4.4%
Interpolate $(\nabla\rho)_n^{\mathcal{V}}$	5	0.13	14.4%
Other		0.07	7.9%
Total		0.90	100%
Niccolò da Uzzano (No. of triangles = 25900)			
Function	**Call**	**Total time (s)**	**Percentage**
Curvature-based curve flattening	1	0.03	1.4%
Initial flattening map	1	0.10	4.6%
Interpolate $\rho^{\mathcal{V}}$	1	0.02	0.9%
Construction of sea	1	0.13	5.9%
Compute cotangent Laplacian L_n	7	0.56	25.6%
Solve diffusion equation	7	0.74	33.8%
Compute $(\nabla\rho)_n^{\mathcal{F}}$	7	0.15	6.8%
Interpolate $(\nabla\rho)_n^{\mathcal{V}}$	7	0.40	18.3%
Other		0.06	2.7%
Total		2.19	100%
Lion (No. of triangles = 33369)			
Function	**Call**	**Total time (s)**	**Percentage**
Curvature-based curve flattening	1	0.03	1.9%
Initial flattening map	1	0.13	8.2%
Interpolate $\rho^{\mathcal{V}}$	1	0.03	1.9%
Construction of sea	1	0.15	9.5%
Compute cotangent Laplacian L_n	4	0.36	22.8%
Solve diffusion equation	4	0.46	29.1%
Compute $(\nabla\rho)_n^{\mathcal{F}}$	4	0.09	5.7%
Interpolate $(\nabla\rho)_n^{\mathcal{V}}$	4	0.25	15.8%
Other		0.08	5.1%
Total		1.58	100%

the absolute relative error in triangle area, defined by

$$E_A(T) = \left| \frac{\text{Area of } T \text{ on the parameterization}}{\text{Area of } T \text{ on the original surface}} - 1 \right| \tag{5.1}$$

for every triangle T. Ideally, all E_A should be equal to 0 for a perfectly area-preserving parameterization. Table 5 lists the performance of the three algorithms. Our DEM algorithm achieves faster computation and higher accuracy for computing area-preserving parameterizations.

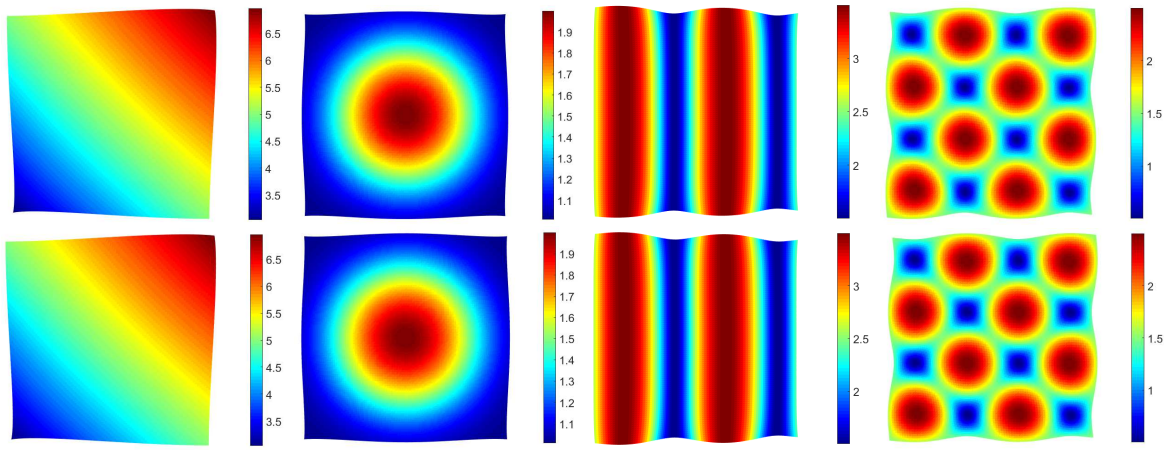

Figure 17. *The density-equalizing maps produced by our proposed DEM algorithm and GN with various input population functions. Each column shows a set of experimental results color-coded by the input population function as described in Table 4. Top row: the results by GN. Bottom row: the results by our method. Our method produces results as accurate as those by GN.*

Table 4

Comparing the performance of our DEM algorithm and GN deployed on a 100×100 *square mesh with various input population functions. Here, the map difference is given by mean*$\left(\frac{|z_{prev}-z_{ours}|}{\text{side length of square}}\right)$*, where* z_{prev} *and* z_{ours} *are, respectively, the complex coordinates of the density-equalizing mapping results by GN and our method.* $\bar{x}$ *and* $\bar{y}$ *are the mean of the* x*-coordinates and the* y*-coordinates of the square.*

Input population	Time by GN (s)	Time by our DEM method (s)	Map difference
$5+\frac{(x-\bar{x})+(y-\bar{y})}{50}$	4.843	0.639	0.0016
$1+e^{-\frac{(x-\bar{x})^2+(y-\bar{y})^2}{1000}}$	4.452	0.848	0.0008
$2.5+\sin\frac{\pi(x-\bar{x})}{25}$	4.959	0.855	0.0012
$1.5+\sin\frac{\pi(x-\bar{x})}{25}\sin\frac{\pi(y-\bar{y})}{25}$	4.592	0.597	0.0026

5.4. On the use and construction of the sea. Note that, unlike our proposed DEM algorithm, most of the existing parameterization approaches do not involve the construction of a sea outside the region of interest. In fact, the sea in our proposed method does not only contribute to the density-equalizing computation in our algorithm but also serves as an important feature for analyzing the physical phenomenon of density propagation around the region of interest.

From a physical point of view, consider the deformation of the sea under the density-equalizing process. Let r be the displacement of a tracer at the sea from the origin before the deformation, and let $\Delta r = r_{\text{final}} - r$ be the change in displacement of it under the density-equalizing process. We can analyze the density propagation under the DEM algorithm by recording Δr for all vertices at the sea. Figure 20 shows several log–log plots of Δr against r outside the unit circle. Our experimental results show that Δr and r are related by the relationship $\Delta r \propto r^{-2}$ at the outer part of the sea. In other words, the influence of density

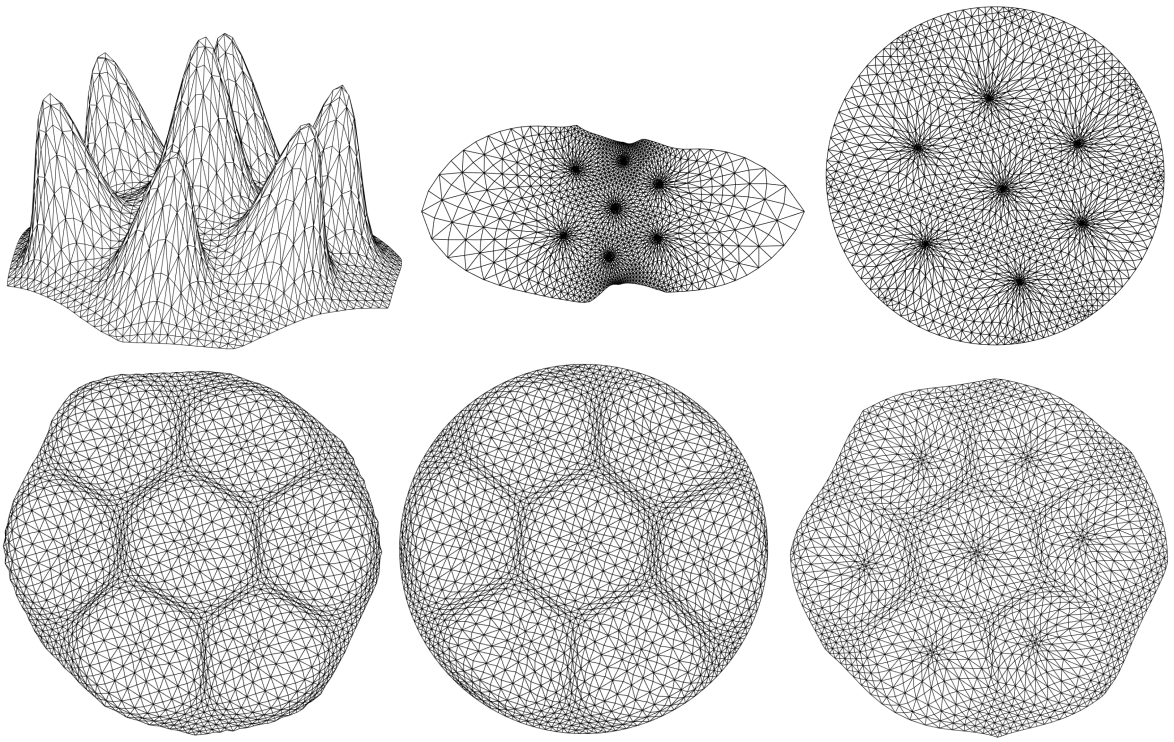

Figure 18. *Comparison of different parameterization schemes for a surface with multiple peaks in* $\mathbb{R}^3$ *shown in Figure 10. Top left: the peaks surface. Top middle: the free-boundary conformal parameterization by Desbrun, Meyer, and Alliez [19]. Top right: the disk conformal parameterization by Choi and Lui [37]. Bottom left: the area-preserving parameterization by our DEM algorithm. Bottom middle: the OMT map [40]. Bottom right: the scalable locally injective map [48].*

diffusion on the displacement of nodes at the sea decays quadratically. From an algorithmic point of view, this also suggests that setting a coarser sea at the outermost part by our reflection-based method does not affect the accuracy of the density-equalizing map.

One may ask why our reflection-based construction of sea with adaptive mesh size is advantageous when compared to other standard constructions. We consider replacing our adaptive sea by several seas consisting of uniformly spaced nodes with various choices of spacing and analyze the performance of the density-equalizing algorithm.

Table 6 shows the performance of DEM using our adaptive sea and uniform seas with three choices of node spacing. Here, a natural choice for a uniform "dense" sea is with node spacing equaling the average vertex spacing (denoted by a) at the initial flattening maps. We also consider two other uniform seas with average spacing $3a$ and $5a$, which are regarded as "moderate" and "coarse" seas, respectively. DEM with the adaptive sea can achieve accuracy at the surface boundary comparable to that with a uniform dense sea while costing less than 10% of computation time when compared to the dense one. When compared to the coarse and moderate seas, the adaptive sea achieves significantly better accuracy at the surface boundary with comparable computation time. This shows that our reflection-based construction of sea can save computational resources without sacrificing the accuracy.

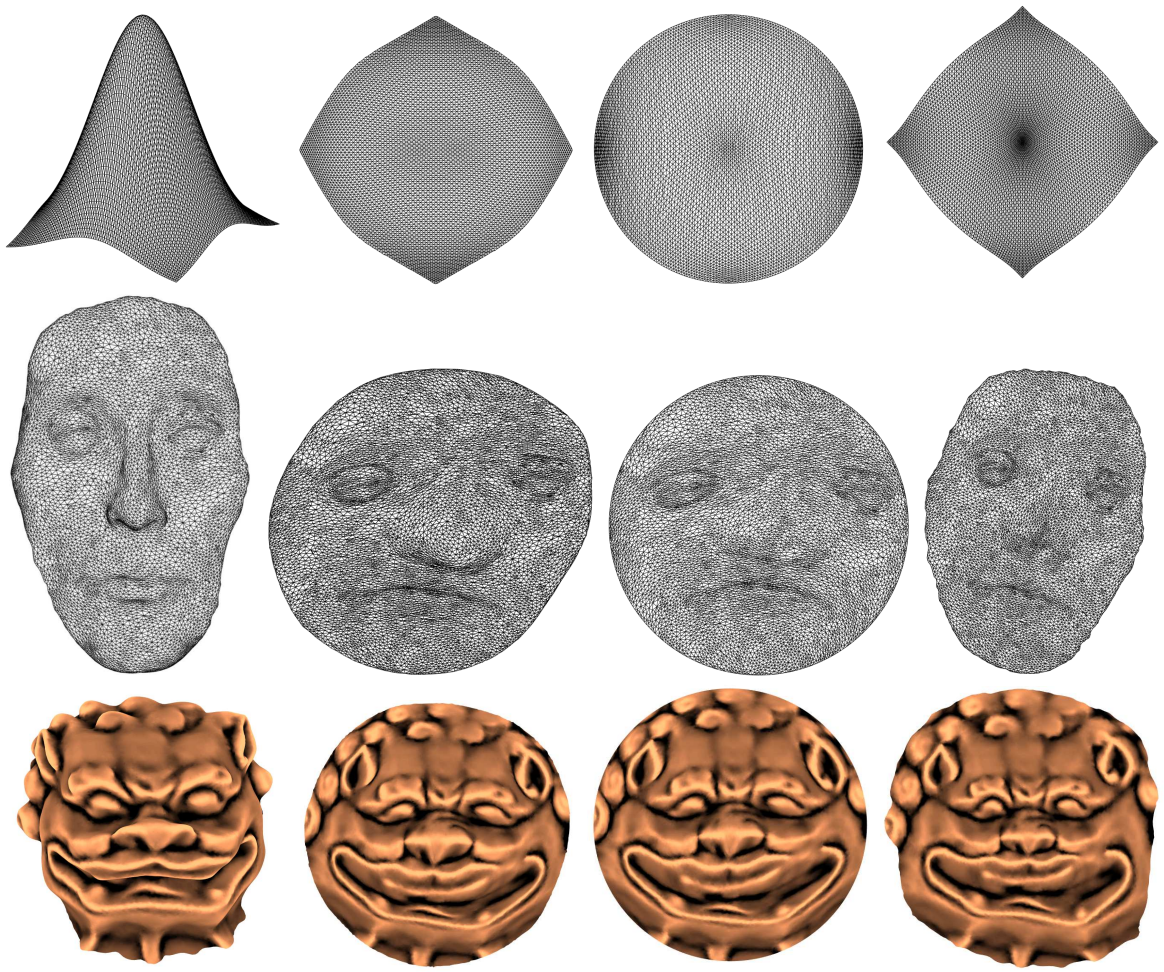

Figure 19. *More comparisons with the state-of-the-art surface parameterization schemes. Left to right: the input surface, DEM, OMT* [40]*, SLIM* [48]*.*

6. Applications.

Our proposed density-equalizing mapping algorithm is useful for various applications. In this section, we discuss two applications of our algorithm.

6.1. Data visualization.

Similarly to GN, our DEM algorithm can be used for data visualization. We consider visualizing the percentage of popular vote for the Republican Party and the Democratic Party in each state in the 2012 and 2016 US presidential elections. To visualize the data, we set the population on each state on a triangulated US map as the percentage of popular vote obtained by the two parties and run our proposed algorithm. Figure 21 shows the density-equalizing results. For the Republican Party, the East Coast and West Coast are significantly shrunk. This reflects the relatively low percentage of popular vote obtained at those regions. By contrast, for the Democratic Party, the East Coast and West Coast are significantly enlarged under the density equalization, which reflects the relative high percentage of popular vote there. Some differences between the 2012 and the 2016 results

Table 5

The performance of our DEM algorithm compared with the state-of-the-art nonlinear parameterization algorithms for computing area-preserving parameterizations. Both OMT [40] and SLIM [48] are run in the default setting in their implementation: The error threshold for OMT is 0.0001*, and the number of iterations for SLIM is* 20*. Note that the choice* 20 *was tested and shown to be a good convergence criterion by the authors of SLIM [48]. We have also verified this by comparing the results with different number of iterations. The input population in DEM is set to be the triangle area for computing area-preserving parameterizations. The distortion measure* $E_A(T) = \left| \frac{\textit{Area of } T \textit{ on the parameterization}}{\textit{Area of } T \textit{ on the original surface}} - 1 \right|$ *is the absolute relative error of the area of each triangle face under the parameterization.*

Surface	No. of triangles	Measure	OMT [40]	SLIM [48]	DEM
Gaussian	10368	Time (s)	0.956	3.413	0.333
		# iterations	22	20	3
		mean(E_A)	0.1014	0.2678	0.0164
		std(E_A)	0.0791	0.2041	0.0233
		median(E_A)	0.1065	0.2104	0.0101
		IQR(E_A)	0.1157	0.2839	0.0154
Peaks	4108	Time (s)	0.466	0.998	0.149
		# iterations	24	20	4
		mean(E_A)	0.1108	0.3214	0.0928
		std(E_A)	0.1015	0.1791	0.1027
		median(E_A)	0.0863	0.3107	0.0649
		IQR(E_A)	0.1106	0.1892	0.0934
Lion	33369	Time (s)	3.278	14.791	1.444
		# iterations	22	20	5
		mean(E_A)	0.1285	0.1857	0.0938
		std(E_A)	0.1062	0.1298	0.0981
		median(E_A)	0.1050	0.1612	0.0640
		IQR(E_A)	0.1299	0.1934	0.0982
Niccolò da Uzzano	25900	Time (s)	2.469	9.211	2.020
		# iterations	22	20	8
		mean(E_A)	0.1282	0.1400	0.0737
		std(E_A)	0.1101	0.0970	0.1461
		median(E_A)	0.1037	0.1266	0.0369
		IQR(E_A)	0.1293	0.1250	0.0589
Max Planck	26452	Time (s)	3.035	9.762	2.021
		# iterations	26	20	9
		mean(E_A)	0.1223	0.1075	0.0754
		std(E_A)	0.1015	0.1078	0.1839
		median(E_A)	0.0991	0.0786	0.0333
		IQR(E_A)	0.1243	0.1174	0.0540
Human face	33369	Time (s)	1.700	3.844	0.793
		# iterations	30	20	6
		mean(E_A)	0.1511	0.0830	0.0612
		std(E_A)	0.1386	0.0824	0.1438
		median(E_A)	0.1174	0.0613	0.0291
		IQR(E_A)	0.1507	0.0830	0.0481

can also be observed. For instance, the area of California becomes more extreme on the density-equalizing maps in 2016 when compared to those in 2012. For Trump, California has further shrunk on the map, while for Clinton, it has further expanded on the map. Another

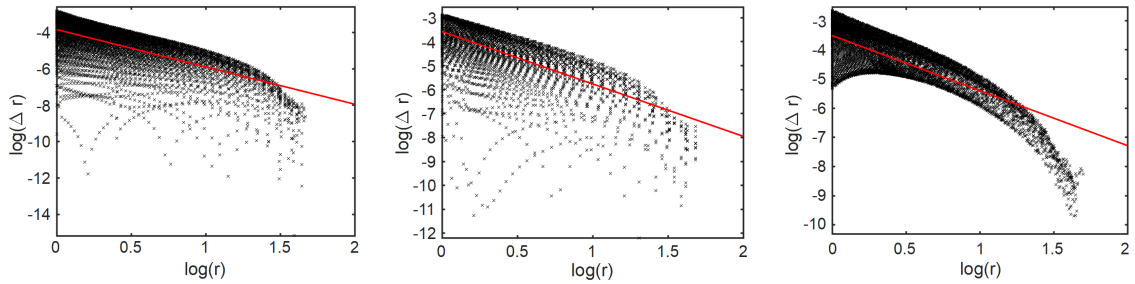

Figure 20. *The log–log plot of the displacement of the sea under our density-equalizing algorithm. To study the effect at the outer sea, only the region outside the unit circle is considered. The x-axis represents the logarithm of the displacement of every point at the sea from the origin. The y-axis represents the logarithm of the change in the displacement under the density-equalizing map. Each cross represents a point at the sea, and the red line is the least-squares line. Left: the square example. Middle: the hexagon example. Right: the human face example.*

Table 6

The performance of the density-equalizing maps with our adaptive sea and three uniform seas. Here the point spacing at the coarse, moderate, and dense seas are, respectively, $5a$, $3a$, and a, where a is the average spacing of the vertices of the initial flattening maps. For a fair comparison, the size of all seas is set to 5, which is consistent with our choice of $\eta = 5$ in our sea construction algorithm. Here d_{bdy} is defined by $\frac{\text{Given population}}{\text{Final area}}$ at the boundary elements of the surfaces under density-equalizing maps.

Surface	Measure	Adaptive sea	Uniform sea (coarse)	Uniform sea (moderate)	Uniform sea (dense)
Square	# points at sea	23882	16148	44463	389182
	DEM time (s)	0.6983	0.5838	1.2037	10.4504
	median(d_{bdy})	1.0057	1.0148	1.0413	1.0137
	IQR(d_{bdy})	0.0707	0.6781	0.1351	0.0689
Hexagon	# points at sea	10022	6437	17511	150677
	DEM time (s)	0.3123	0.2615	0.5022	3.8555
	median(d_{bdy})	1.0290	1.0630	1.0239	1.0163
	IQR(d_{bdy})	0.0695	0.3504	0.1754	0.0676
Gaussian	# points at sea	18676	15187	43688	364657
	DEM time (s)	0.4088	0.3820	0.7674	6.1897
	median(d_{bdy})	0.9684	1.5254	1.1205	0.9646
	IQR(d_{bdy})	0.1442	1.7124	0.6226	0.1478
Max Planck	# points at sea	31444	15187	41714	364737
	DEM time (s)	2.4307	1.7412	2.9786	19.5934
	median(d_{bdy})	1.1282	1.3877	1.2603	1.1387
	IQR(d_{bdy})	0.2898	1.1896	0.6747	0.2510

example is West Virginia. The area of it has decreased in the map for Clinton when compared to that for Obama, while the area has increased in the map for Trump when compared to that for Romney. This example of US presidential elections shows the usefulness of our DEM algorithm in data visualization.

An interesting feature of US presidential elections is the electoral college system. As the number of electoral votes is different for different states, it is interesting to ask how powerful

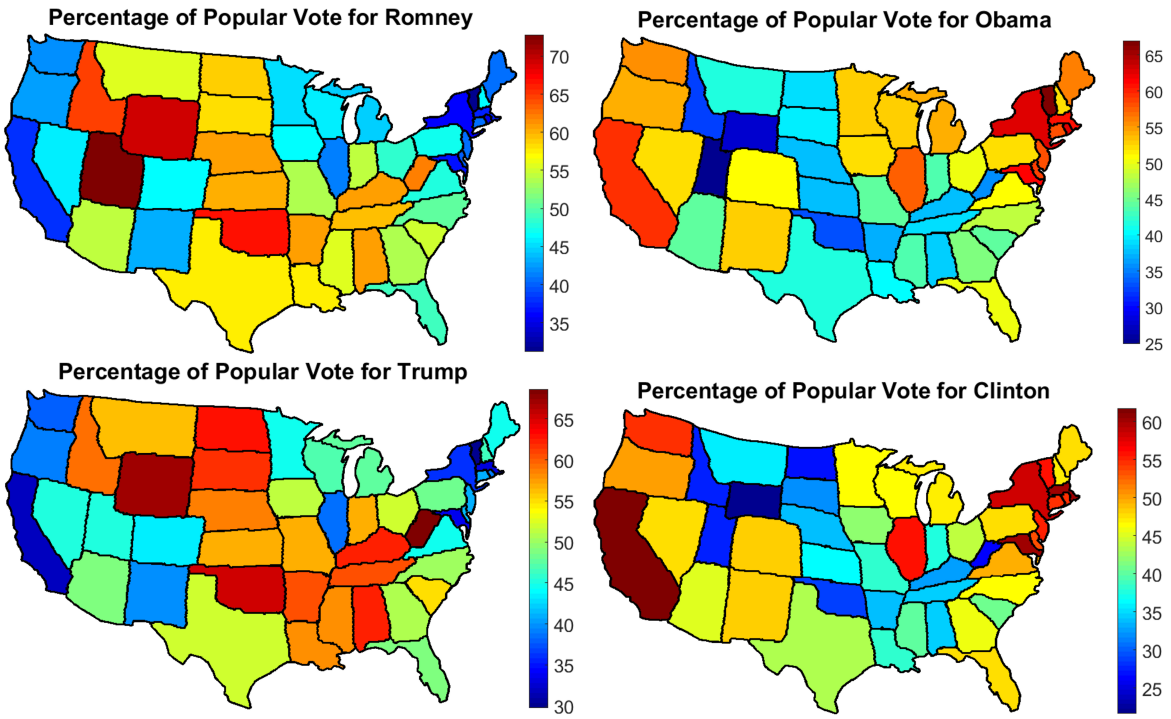

Figure 21. *Percentage of popular vote in each state visualized on density-equalizing US maps (only including the contiguous 48 states). The triangulations are set to be transparent for enhancing the visual quality.*

each popular vote is in each state. We define the vote power in each state by

$$\text{Vote power} = \frac{\frac{\text{Number of electoral votes}}{\text{Number of popular votes}}}{\text{mean}\left(\frac{\text{Number of electoral votes}}{\text{Number of popular votes}}\right)}. \tag{6.1}$$

We would like to visualize it on an area cartogram using DEM. Note that the vote power is not related to the area of each state. To remove the effect of the original area of each state such that the area cartogram solely reflects the vote power, we run our DEM algorithm with input $\frac{\text{Vote power}}{\text{Area of the state}}$. Figure 22 shows the resulting cartogram. In contrast to the original US map, the DEM cartogram gives us a more intuitive view of the vote power. For instance, Wyoming is much bigger than Texas in the cartogram, as the vote power in Wyoming is significantly greater than that in Texas. This can give us a better understanding of the electoral system.

6.2. Adaptive surface remeshing. Note that the input population affects the size of different regions in the resulting density-equalizing map. Specifically, a higher population leads to a magnification, and a lower population leads to a shrinkage. Using this property of the density-equalizing map, we can perform adaptive surface remeshing easily.

Let S be a surface to be remeshed. Given a population, we first compute the density-equalizing map $f : S \to \mathbb{C}$. Now consider a set of uniformly distributed points $\mathcal{P}$ on the density-equalizing map. We triangulate the set of points and denote the triangulation by $\mathcal{T}$.

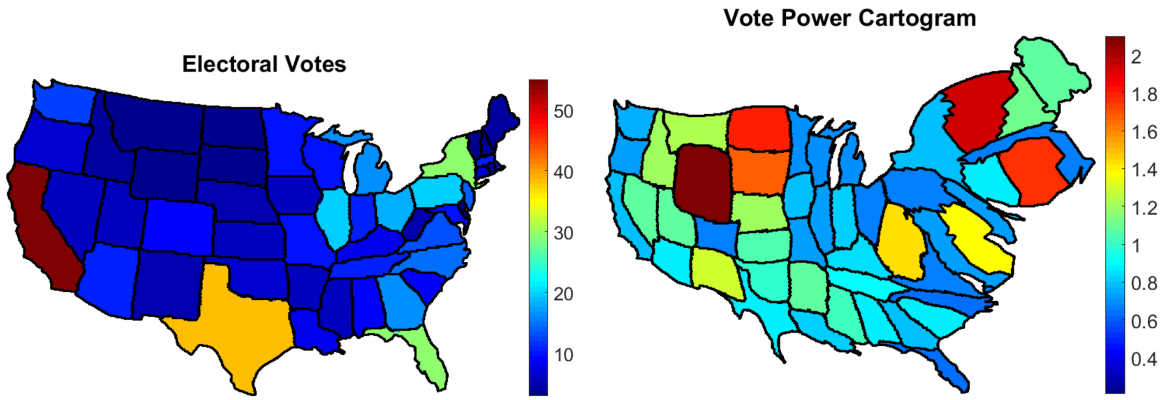

Figure 22. *Visualizing the vote power of different states (only including the contiguous* 48 *states). Left: the normal US map color coded by the number of electoral votes. Right: the area cartogram generated by DEM that accurately reflects the vote power of all states.*

Then, using the inverse mapping f^{-1}, we can interpolate $\mathcal{P}$ onto S. The mesh $(f^{-1}(\mathcal{P}), \mathcal{T})$ gives a remeshed representation of the surface S.

Now, to increase the level of detail at a region of S, we can set a larger population there in running our density-equalizing mapping algorithm. Since the region is enlarged in the mapping result and $\mathcal{P}$ is uniformly distributed, more points will lie on that part, and hence the inverse mapping will map more points back onto that particular region of S. This completes our adaptive surface remeshing scheme.

Figure 23 shows an example of remeshing a triangulated human face using the above-mentioned scheme. We set the population to be the triangle area of the original mesh in running our algorithm. To highlight the advantage of the use of our density-equalizing map, we compare the remeshing result with that obtained via a conventional free-boundary conformal parameterization method [19]. The eyes and the nose of the human face are enlarged in our density-equalizing mapping result, while such features are shrunk in the conformal parameterization because of the preservation of conformality. This difference causes significantly different remeshing results. Also, note that the representation of the nose is poor in the remeshing result via conformal parameterization. By contrast, the remeshing result via our density-equalizing mapping algorithm is with a more balanced distribution of points. This example demonstrates the strength of our algorithm in surface remeshing.

7. Discussion.

We conclude the paper with a discussion of the advantages, limitations, and possible extensions of our work.

7.1. Advantages.

In this work, we have proposed an efficient algorithm for computing density-equalizing flattening maps of simply connected open surfaces in $\mathbb{R}^3$. When compared to GN, our method is particularly well suited to planar domains with complex geometry because of the use of triangle meshes. With this advantage, our method can possibly lead to a wider range of applications of density-equalizing maps in data visualization. Our method is also well suited for handling disk-like surfaces in $\mathbb{R}^3$ such as human faces. This suggests a new approach for adaptive surface remeshing via density-equalizing maps. When compared to the existing

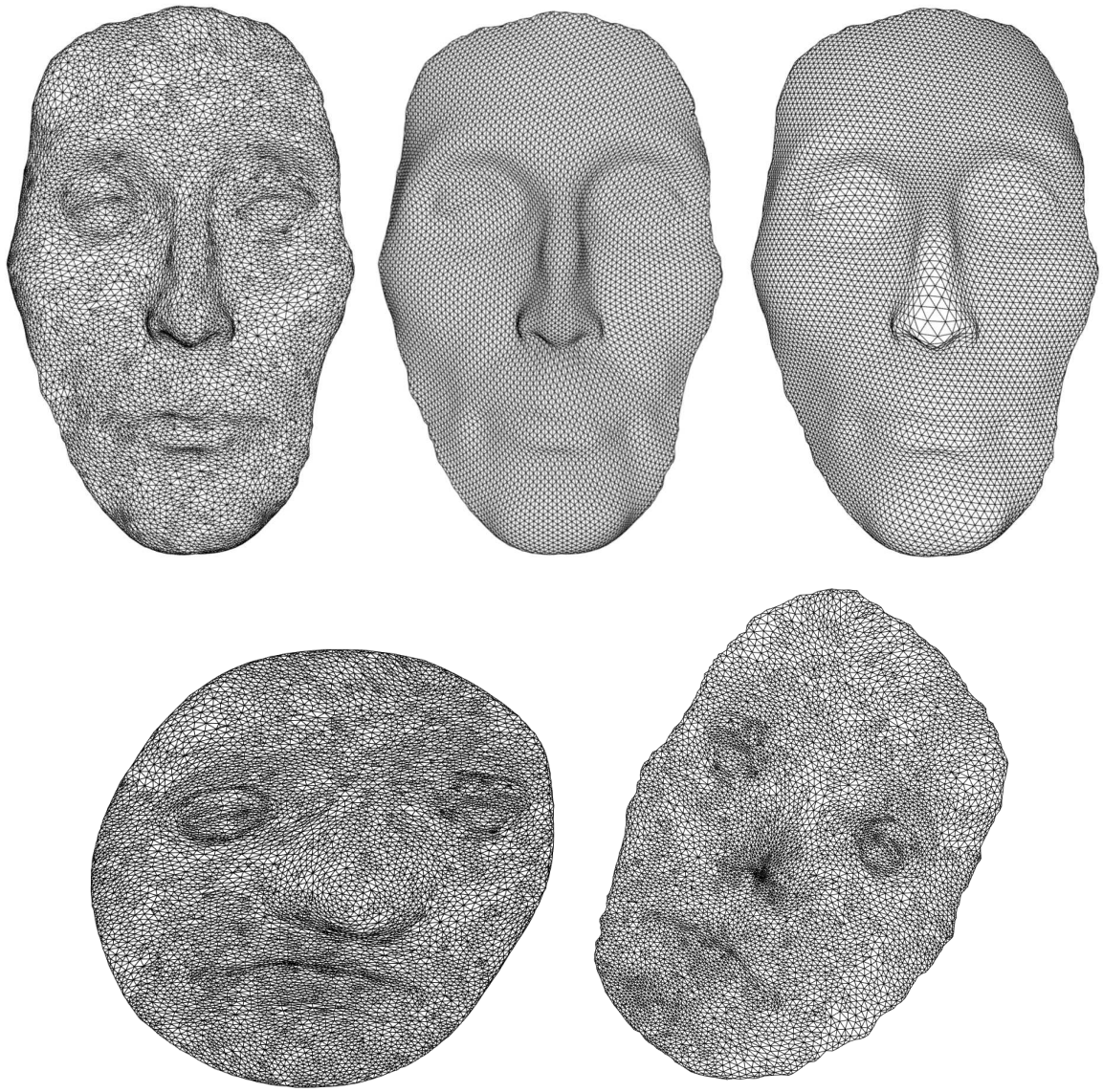

Figure 23. *Remeshing a human face. Top left: the original human face. Top middle: the remeshing result via our parameterization. Top right: the remeshing result via the free-boundary conformal parameterization by Desbrun, Meyer, and Alliez* [19]. *Bottom left: the density-equalizing parameterization by our algorithm. Bottom right: the free-boundary conformal parameterization by Desbrun, Meyer, and Alliez* [19].

parameterization-based remeshing approaches, our method can easily control the remeshing quality at different regions of the surfaces by changing the population at those regions.

7.2. Limitations. Since the density-diffusion process is solved on a triangle mesh, the triangle quality affects the accuracy of the discretization and hence the final density-equalization result. If the triangle mesh consists of highly irregular triangle elements, the ultimate density distribution may not be optimal even after the algorithm converges. Also, since the

discretization is based on the triangles for every step in our algorithm, if the input population is too extreme or highly discontinuous, the triangles may become highly irregular at a certain step and affect the accuracy of the subsequent results. In other words, triangle meshes with moderate triangle quality and input population are desired. In addition, for surfaces in $\mathbb{R}^3$ with a highly tubular shape and with the boundary lying at one end, the flattening step may cause extremely squeezed regions on the planar domain. In this case, the accuracy of the subsequent computations for density equalization may be affected.

7.3. Extensions. It is well known that the discretization accuracy of the cotangent Laplacian is affected by the mesh regularity. For applications that require higher discretization accuracy of the Laplacian, one can introduce an extra step of recomputing a Delaunay triangulation at every iteration in Algorithm 5. This can improve the accuracy of the Laplacian. Note that the change in triangulation will not cause any ambiguity in the density-equalizing process, as all density values are stored at vertices.

While mesh fold-overs are fairly rare in our experiments, currently there is no theoretical guarantee about the bijectivity of the density-equalizing maps computed. Nevertheless, as the deformation process is done at the step $\mathbf{r}_n = \mathbf{r}_{n-1} - \delta t(\nabla\rho)_n^{\mathcal{V}}/\rho_n^{\mathcal{V}}$, one possible future direction to ensure the bijectivity may be to let the time step δt change adaptively throughout the iterations to check and prevent any occurrence of fold-over.

Although our current work only focuses on simply-connected surfaces in $\mathbb{R}^3$, it can be naturally extended to general surfaces. For instance, density-equalizing maps of multiply-connected surfaces can be computed by filling up the holes and treating them as the sea in our proposed algorithm. Similarly, density-equalizing maps of multiple disconnected surfaces can be handled with the aid of a large sea.

Appendix A. We prove that the curvature-based curve flattening step in section 4.1 produces a simple closed convex curve.

Proposition A.1. *Let $\varphi : [0, l_\gamma] \to \mathbb{R}^2$ be the arclength parameterized curve defined as in section 4.1. Consider the new curve $\Phi : [0, l_\gamma] \to \mathbb{R}^2$ defined by*

$$\Phi(s) = \varphi(s) - \frac{s}{l_\gamma}\left(\varphi(l_\gamma) - \varphi(0)\right). \tag{A.1}$$

Φ is a simple closed convex curve.

Proof. It is easy to note that $\Phi(0) = \varphi(0) = \Phi(l_\gamma)$ and hence Φ is closed. Since φ is an arclength parameterized curve, for any $0 \le a < b \le l_\gamma$ we have

$$\|\varphi(b) - \varphi(a)\| \le \int_a^b \|\varphi'(s)\| ds = b - a, \tag{A.2}$$

where the equality holds if and only if $\varphi([a, b])$ is a straight line. In particular, since γ is the boundary of the original simply connected open surface, by our construction of φ, we have $\|\varphi(l_\gamma) - \varphi(0)\| \ll l_\gamma$.

We now prove that the signed curvature of Φ, denoted by k_Φ, is nonnegative for all $s \in [0, l_\gamma]$.

Denote $\varphi(s) = (x(s), y(s))$ and $\Phi(s) = (X(s), Y(s))$. We have

$$\begin{aligned}\Phi' &= (X', Y') \\ &= \left(x'(s) - \frac{1}{l_\gamma}\left(x(l_\gamma) - x(0)\right), y'(s) - \frac{1}{l_\gamma}\left(y(l_\gamma) - y(0)\right)\right) \\ &= \left(\cos\theta(s) - \frac{1}{l_\gamma}\left(x(l_\gamma) - x(0)\right), \sin\theta(s) - \frac{1}{l_\gamma}\left(y(l_\gamma) - y(0)\right)\right)\end{aligned} \tag{A.3}$$

and

$$\Phi'' = (X'', Y'') = (x'', y'') = (-k_\varphi(s)\sin\theta(s), k_\varphi(s)\cos\theta(s)). \tag{A.4}$$

Hence, we have

$$X'Y'' - X''Y' = k_\varphi(s)\left(1 - \frac{(x(l_\gamma) - x(0))\cos\theta(s) - (y(l_\gamma) - y(0))\sin\theta(s)}{l_\gamma}\right). \tag{A.5}$$

Now recall that by (4.5), $k_\varphi(s) \geq 0$ for all s. Also, we have

$$\begin{aligned}&(x(l_\gamma) - x(0))\cos\theta(s) - (y(l_\gamma) - y(0))\sin\theta(s) \\ &\leq \sqrt{(x(l_\gamma) - x(0))^2 + (y(l_\gamma) - y(0))^2}\sqrt{\cos^2\theta(s) + \sin^2\theta(s)} \\ &= \sqrt{(x(l_\gamma) - x(0))^2 + (y(l_\gamma) - y(0))^2} \\ &= \|\varphi(l_\gamma) - \varphi(0)\| \\ &\leq l_\gamma.\end{aligned} \tag{A.6}$$

Here, the first inequality follows from the Cauchy–Schwarz inequality, and the second inequality follows from (A.2). Therefore, we have

$$1 - \frac{(x(l_\gamma) - x(0))\cos\theta(s) - (y(l_\gamma) - y(0))\sin\theta(s)}{l_\gamma} \geq 0 \text{ for all } s \in [0, l_\gamma], \tag{A.7}$$

and it follows that

$$k_\Phi(s) = \frac{X'Y'' - X''Y'}{(X'^2 + Y'^2)^{3/2}} \geq 0 \text{ for all } s \in [0, l_\gamma]. \tag{A.8}$$

We proceed to show that Φ is simple. Note that since Φ is a closed plane curve, the total curvature of Φ should satisfy

$$\int_0^{l_\varphi} k_\Phi(s)ds = 2\pi n_\Phi, \tag{A.9}$$

where n_Φ is the turning number of Φ. From the above results, we have

$$\begin{aligned} 2\pi n_\Phi &= \int_0^{l_\varphi} \frac{k_\varphi(s)\left(1-\frac{(x(l_\gamma)-x(0))\cos\theta(s)-(y(l_\gamma)-y(0))\sin\theta(s)}{l_\gamma}\right)}{(X'^2+Y'^2)^{3/2}}ds \\ &\le \left(\int_0^{l_\varphi} k_\varphi(s)ds\right) \max_{s\in[0,l_\gamma]} \left| \frac{\left(1-\frac{(x(l_\gamma)-x(0))\cos\theta(s)-(y(l_\gamma)-y(0))\sin\theta(s)}{l_\gamma}\right)}{(X'^2+Y'^2)^{3/2}} \right| \\ &= 2\pi \max_{s\in[0,l_\gamma]} \left(\frac{1-A\cos\theta(s)+B\sin\theta(s)}{(1+A^2+B^2-2A\cos\theta(s)-2B\sin\theta(s))^{3/2}} \right), \end{aligned} \tag{A.10}$$

where $A=\frac{x(l_\gamma)-x(0)}{l_\gamma}$ and $B=\frac{y(l_\gamma)-y(0)}{l_\gamma}$. The above equation can be further simplified to

$$2\pi \max_{s\in[0,l_\gamma]} \left(\frac{1-C\cos(\theta(s)+\eta)}{(1+C^2-2C\cos(\theta(s)-\eta))^{3/2}} \right), \tag{A.11}$$

where $C=\sqrt{A^2+B^2}=\frac{\|\varphi(l_\gamma)-\varphi(0)\|}{l_\gamma}\ll 1$ and $\eta=\tan^{-1}\frac{B}{A}=\tan^{-1}\frac{y(l_\gamma)-y(0)}{x(l_\gamma)-x(0)}$. Hence, it is easy to see that

$$\max_{s\in[0,l_\gamma]} \left(\frac{1-C\cos(\theta(s)+\eta)}{(1+C^2-2C\cos(\theta(s)-\eta))^{3/2}} \right) \le \frac{1+C}{(1-C)^3} < 2. \tag{A.12}$$

This implies that

$$2\pi n_\Phi < 4\pi \Rightarrow n_\Phi < 2. \tag{A.13}$$

It follows that $n_\Phi = 1$ and hence Φ is simple. Finally, note that a simple closed curve is convex if and only if its signed curvature does not change sign [79]. From (A.8), we conclude that Φ is a simple closed convex curve. ∎

Appendix B. We review the finite element formulation for the discrete Laplacian. Readers are referred to the paper by Reuter et al. [73] for a more detailed discussion. Here we aim to highlight the assumption of the natural boundary condition in the derivation in the FEM Laplacian formula (4.23), which provides us with a theoretical support for our shape-prescribed density-equalizing maps (Algorithm 6).

To solve the equation $\Delta u = f$ on a domain Ω, we let $u=\sum x_i\phi_i$ and $f=\sum f_i\phi_i$, where ϕ_i is the hat function on vertex i, $i=1,2,\ldots,|\mathcal{V}|$. We have

$$\int_\Omega \Delta u\phi_j = \int_\Omega f\phi_j \tag{B.1}$$

for all j. By Green's first identity, we have

$$\int_\Omega \Delta u\phi_j = \int_{\partial\Omega} \phi_j(\nabla u\cdot\mathbf{n}) - \int_\Omega \nabla u\cdot\nabla\phi_j. \tag{B.2}$$

Suppose that Ω does not have boundary or the natural boundary condition $\nabla u \cdot \mathbf{n} = 0$ holds. Then we have

$$\int_\Omega \Delta u \phi_j = -\int_\Omega \nabla u \cdot \nabla \phi_j = -\int_\Omega \nabla \left(\sum x_i \phi_i \right) \cdot \nabla \phi_j = -\sum x_i \int_\Omega \nabla \phi_i \cdot \nabla \phi_j. \tag{B.3}$$

Also, it is easy to see that

$$\int_\Omega f \phi_j = \sum f_i \int_\Omega \phi_i \cdot \phi_j. \tag{B.4}$$

Hence it suffices to solve

$$-L\mathbf{x} = A\mathbf{f}, \tag{B.5}$$

where L is a $|\mathcal{V}| \times |\mathcal{V}|$ matrix with $L_{ij} = \int_\Omega \nabla \phi_i \cdot \nabla \phi_j$, and A (called the mass matrix) is a $|\mathcal{V}| \times |\mathcal{V}|$ matrix with $A_{ij} = \int_\Omega \phi_i \cdot \phi_j$. Using trigonometry, one can show that L is exactly the cotangent Laplacian [72]

$$L_{ij} = \begin{cases} -\frac{1}{2}(\cot \alpha_{ij} + \cot \beta_{ij}) & \text{if } [x_i, x_j] \in \mathcal{E}, \\ -\sum_{k \neq i} L_{ik} & \text{if } j = i, \\ 0 & \text{otherwise,} \end{cases} \tag{B.6}$$

with α_{ij} and β_{ij} being the two angles opposite to the edge $[i, j]$. Similarly,

$$A_{ij} = \begin{cases} \frac{1}{12}(\text{Area}(T_{ij}^1) + \text{Area}(T_{ij}^2)) & \text{if } [x_i, x_j] \in \mathcal{E}, \\ \frac{1}{6} \sum_{T \in \mathcal{N}^{\mathcal{F}}(i)} \text{Area}(T) & \text{if } j = i, \\ 0 & \text{otherwise,} \end{cases} \tag{B.7}$$

where T_{ij}^1 and T_{ij}^2 are the two triangles incident to both i, j. To simplify computation, one common approach is to lump the mass matrix A, i.e., to add all entries in each row to the diagonal entry. This gives us an approximation of A by a diagonal matrix $\widetilde{A}$ with $\widetilde{A}_{ii} = \frac{1}{3} \sum_{T \in \mathcal{N}^{\mathcal{F}}(i)} \text{Area}(T)$. Hence, the Laplacian Δ can be discretized as

$$\Delta = -\widetilde{A}^{-1} L, \tag{B.8}$$

which is equivalent to (4.23).

Acknowledgment. We would like to thank the anonymous reviewers for their valuable comments and suggestions.

REFERENCES

[1] M. T. Gastner and M. E. J. Newman, *Diffusion-based method for producing density-equalizing maps*, Proc. Natl. Acad. Sci. USA, 101 (2004), pp. 7499–7504.

[2] D. Dorling, M. Newman, and A. Barford, *The Atlas of the Real World: Mapping the Way We Live*, Thames & Hudson, 2010.

[3] V. Colizza, A. Barrat, M. Barthélemy, and A. Vespignani, *The role of the airline transportation network in the prediction and predictability of global epidemics*, Proc. Natl. Acad. Sci. USA, 103 (2006), pp. 2015–2020.

[4] D. B. Wake and V. T. Vredenburg, *Are we in the midst of the sixth mass extinction? A view from the world of amphibians*, Proc. Natl. Acad. Sci. USA, 105 (2008), pp. 11466–11473.
[5] K. S. Gleditsch and M. D. Ward, *Diffusion and the international context of democratization*, Internat. Org., 60 (2006), pp. 911–933.
[6] A. Mislove, S. Lehmann, Y. Y. Ahn, J. P. Onnela, and J. N. Rosenquist, *Understanding the demographics of Twitter users*, in Proceedings of the 5th International AAAI Conference on Weblogs and Social Media (CWSM), 2011, pp. 554–557.
[7] A. Vanasse, M. Demers, A. Hemiari, and J. Courteau, *Obesity in Canada: Where and how many?*, Internat. J. Obesity, 30 (2006), pp. 677–683.
[8] R. K. Pan, K. Kaski, and S. Fortunato, *World citation and collaboration networks: Uncovering the role of geography in science*, Sci. Reports, 2 (2012), 902.
[9] D. Dorling, *Area Cartograms: Their Use and Creation*, Concepts and Techniques in Modern Geography 59, Environmental Publications, University of East Anglia, 1996.
[10] H. Edelsbrunner and R. Waupotitsch, *A combinatorial approach to cartograms*, Comput. Geom., 7 (1997), pp. 343–360.
[11] D. A. Keim, S. C. North, and C. Panse, *Cartodraw: A fast algorithm for generating contiguous cartograms*, IEEE Trans. Visualization Comput. Graphics, 10 (2004), pp. 95–110.
[12] D. A. Keim, C. Panse, and S. C. North, *Medial-axis-based cartograms*, IEEE Comput. Graphics Appl., 25 (2005), pp. 60–68.
[13] M. Floater and K. Hormann, *Surface parameterization: A tutorial and survey*, in Advances in Multiresolution for Geometric Modelling, Springer, 2005, pp. 157–186.
[14] A. Sheffer, E. Praun, and K. Rose, *Mesh parameterization methods and their applications*, Found. Trends Comput. Graphics Vision, 2 (2006), pp. 105–171.
[15] K. Hormann, B. Lévy, and A. Sheffer, *Mesh parameterization: Theory and practice*, in Proceedings of ACM SIGGRAPH 2007, ACM, 2007, pp. 1–122, article 1.
[16] S. Angenent, S. Haker, A. Tannenbaum, and R. Kikinis, *Conformal geometry and brain flattening*, in Medical Image Computing and Computer-Assisted Intervention (MICCAI), 1999, pp. 271–278.
[17] S. Haker, S. Angenent, A. Tannenbaum, R. Kikinis, G. Sapiro, and M. Halle, *Conformal surface parameterization for texture mapping*, IEEE Trans. Visualization Comput. Graphics, 6 (2000), pp. 181–189.
[18] B. Lévy, S. Petitjean, N. Ray, and J. Maillot, *Least squares conformal maps for automatic texture atlas generation*, ACM Trans. Graphics (Proceedings of ACM SIGGRAPH 2002), 21 (2002), pp. 362–371.
[19] M. Desbrun, M. Meyer, and P. Alliez, *Intrinsic parameterizations of surface meshes*, Computer Graphics Forum, 21 (2002), pp. 209–218.
[20] A. Sheffer and E. de Sturler, *Parameterization of faceted surfaces for meshing using angle-based flattening*, Engineering with Computers, 17 (2001), pp. 326–337.
[21] A. Sheffer, B. Lévy, M. Mogilnitsky, and A. Bogomyakov, *ABF++: Fast and robust angle based flattening*, ACM Trans. Graphics, 24 (2005), pp. 311–330.
[22] R. Zayer, B. Lévy, and H.-P. Seidel, *Linear angle based parameterization*, in Eurographics Symposium on Geometry Processing, 2007, pp. 135–141.
[23] X. Gu, Y. Wang, T. F. Chan, P. M. Thompson, and S. T. Yau, *Genus zero surface conformal mapping and its application to brain surface mapping*, IEEE Trans. Med. Imaging, 23 (2004), pp. 949–958.
[24] M. Jin, Y. Wang, S. T. Yau, and X. Gu, *Optimal global conformal surface parameterization*, in Proceedings of the Conference on Visualization'04, 2004, pp. 267–274.
[25] F. Luo, *Combinatorial Yamabe flow on surfaces*, Commun. Contemp. Math., 6 (2004), pp. 765–780.
[26] L. Kharevych, B. Springborn, and P. Schröder, *Discrete conformal mappings via circle patterns*, ACM Trans. Graphics, 25 (2006), pp. 412–438.
[27] P. Mullen, Y. Tong, P. Alliez, and M. Desbrun, *Spectral conformal parameterization*, Computer Graphics Forum, 27 (2008), pp. 1487–1494.
[28] M. Ben-Chen, C. Gotsman, and G. Bunin, *Conformal flattening by curvature prescription and metric scalings*, Computer Graphics Forum, 27 (2008), pp. 449–458.
[29] B. Springborn, P. Schröder, and U. Pinkall, *Conformal equivalence of triangle meshes*, ACM Trans.

Graphics, 27 (2008).
[30] M. Jin, J. Kim, F. Luo, and X. Gu, *Discrete surface Ricci flow*, IEEE Trans. Visualization Comput. Graphics, 14 (2008), pp. 1030–1043.
[31] Y. L. Yang, R. Guo, F. Luo, S. M. Hu, and X. F. Gu, *Generalized discrete Ricci flow*, Computer Graphics Forum, 28 (2009), pp. 2005–2014.
[32] M. Zhang, R. Guo, W. Zeng, F. Luo, S.-T. Yau, and X. Gu, *The unified discrete surface Ricci flow*, Graphical Models, 76 (2014), pp. 321–339.
[33] P. T. Choi, K. C. Lam, and L. M. Lui, *FLASH: Fast landmark aligned spherical harmonic parameterization for genus-0 closed brain surfaces*, SIAM J. Imaging Sci., 8 (2015), pp. 67–94, https://doi.org/10.1137/130950008.
[34] P. T. Choi and L. M. Lui, *Fast disk conformal parameterization of simply-connected open surfaces*, J. Sci. Comput., 65 (2015), pp. 1065–1090.
[35] G. P.-T. Choi, K. T. Ho, and L. M. Lui, *Spherical conformal parameterization of genus-0 point clouds for meshing*, SIAM J. Imaging Sci., 9 (2016), pp. 1582–1618, https://doi.org/10.1137/15M1037561.
[36] T. W. Meng, G. P.-T. Choi, and L. M. Lui, *TEMPO: Feature-endowed Teichmüller extremal mappings of point clouds*, SIAM J. Imaging Sci., 9 (2016), pp. 1922–1962, https://doi.org/10.1137/15M1049117.
[37] G. P.-T. Choi and L. M. Lui, *A linear formulation for disk conformal parameterization of simply-connected open surfaces*, Adv. Comput. Math., 44 (2018), pp. 87–114.
[38] R. Sawhney and K. Crane, *Boundary First Flattening*, preprint, https://arxiv.org/abs/1704.06873, 2017.
[39] G. Zou, J. Hu, X. Gu, and J. Hua, *Authalic parameterization of general surfaces using Lie advection*, IEEE Trans. Visualization Comput. Graphics, 17 (2011), pp. 2005–2014.
[40] X. Zhao, Z. Su, X. D. Gu, A. Kaufman, J. Sun, J. Gao, and F. Luo, *Area-preservation mapping using optimal mass transport*, IEEE Trans. Visualization Comput. Graphics, 19 (2013), pp. 2838–2847.
[41] J.-D. Benamou, B. D. Froese, and A. M. Oberman, *Numerical solution of the optimal transportation problem using the Monge-Ampère equation*, J. Comput. Phys., 260 (2014), pp. 107–126.
[42] K. Su, L. Cui, K. Qian, N. Lei, J. Zhang, M. Zhang, and X. D. Gu, *Area-preserving mesh parameterization for poly-annulus surfaces based on optimal mass transportation*, Comput Aided Geom. Design, 46 (2016), pp. 76–91.
[43] P. Degener, J. Meseth, and R. Klein, *An adaptable surface parameterization method*, in Proceedings of the 12th International Meshing Roundtable, 2003, pp. 227–237.
[44] L. Liu, L. Zhang, Y. Xu, C. Gotsman, and S. J. Gortler, *A local/global approach to mesh parameterization*, Computer Graphics Forum, 27 (2008), pp. 1495–1504.
[45] J. Smith and S. Schaefer, *Bijective parameterization with free boundaries*, ACM Trans. Graphics, 34 (2015), 70.
[46] K. C. Lam and L. M. Lui, *Optimized quasiconformal parameterization with user-defined area distortions*, Commun. Math. Sci., 15 (2017), pp. 2027–2054.
[47] S. Nadeem, Z. Su, W. Zeng, A. Kaufman, and X. Gu, *Spherical parameterization balancing angle and area distortions*, IEEE Trans. Visualization Comput. Graphics, 23 (2017), pp. 1663–1676.
[48] M. Rabinovich, R. Poranne, D. Panozzo, and O. Sorkine-Hornung, *Scalable locally injective mappings*, ACM Trans. Graphics, 36 (2017), 16.
[49] Y. Wang, L. M. Lui, T. F. Chan, and P. M. Thompson, *Optimization of brain conformal mapping with landmarks*, in International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI), 2005, pp. 675–683.
[50] L. M. Lui, Y. Wang, T. F. Chan, and P. Thompson, *Landmark constrained genus zero surface conformal mapping and its application to brain mapping research*, Appl. Numer. Math., 57 (2007), pp. 847–858.
[51] Y. Wang, L. M. Lui, X. Gu, K. M. Hayashi, T. F. Chan, A. W. Toga, P. M. Thompson, and S. T. Yau, *Brain surface conformal parameterization using Riemann surface structure*, IEEE Trans. Med. Imaging, 26 (2007), pp. 853–865.
[52] L. M. Lui, S. Thiruvenkadam, Y. Wang, P. M. Thompson, and T. F. Chan, *Optimized conformal surface registration with shape-based landmark matching*, SIAM J. Imaging Sci., 3 (2010), pp. 52–78, https://doi.org/10.1137/080738386.
[53] X. Li, Y. Bao, X. Guo, M. Jin, X. Gu, and H. Qin, *Globally optimal surface mapping for surfaces*

with arbitrary topology, IEEE Trans. Visualization Comput. Graphics, 14 (2008), pp. 805–819.
[54] X. Li, X. Gu, and H. Qin, *Surface mapping using consistent pants decomposition*, IEEE Trans. Visualization Comput. Graphics, 15 (2009), pp. 558–571.
[55] O. Weber, A. Myles, and D. Zorin, *Computing extremal quasiconformal maps*, Computer Graphics Forum, 31 (2012), pp. 1679–1689.
[56] W. Zeng, L. M. Lui, F. Luo, T. F. Chan, S. T. Yau, and X. Gu, *Computing quasiconformal maps using an auxiliary metric and discrete curvature flow*, Numer. Math., 121 (2012), pp. 671–703.
[57] L. M. Lui, K. C. Lam, T. W. Wong, and X. Gu, *Texture map and video compression using Beltrami representation*, SIAM J. Imaging Sci., 6 (2013), pp. 1880–1902, https://doi.org/10.1137/120866129.
[58] L. M. Lui, K. C. Lam, S. T. Yau, and X. Gu, *Teichmüller mapping (T-map) and its applications to landmark matching registration*, SIAM J. Imaging Sci., 7 (2014), pp. 391–426, https://doi.org/10.1137/120900186.
[59] G. P.-T. Choi, M. H. Y. Man, and L. M. Lui, *Fast spherical quasiconformal parameterization of genus-0 closed surfaces with application to adaptive remeshing*, Geom. Imaging Comput., to appear.
[60] Y. Lipman, *Bounded distortion mapping spaces for triangular meshes*, ACM Trans. Graphics, 31 (2012), 108.
[61] N. Aigerman, R. Poranne, and Y. Lipman, *Lifted bijections for low distortion surface mappings*, ACM Trans. Graphics, 33 (2014), 69.
[62] E. Chien, Z. Levi, and O. Weber, *Bounded distortion parametrization in the space of metrics*, ACM Trans. Graphics, 35 (2016), 215.
[63] A. Shtengel, R. Poranne, O. Sorkine-Hornung, S. Z. Kovalsky, and Y. Lipman, *Geometric optimization via composite majorization*, ACM Trans. Graphics, 36 (2017), 38.
[64] S. Claici, M. Bessmeltsev, S. Schaefer, and J. Solomon, *Isometry-aware preconditioning for mesh parameterization*, Computer Graphics Forum, 36 (2017), pp. 37–47.
[65] Z. Jiang, S. Schaefer, and D. Panozzo, *Simplicial complex augmentation framework for bijective maps*, ACM Trans. Graphics (SIGGRAPH Asia 2017), 36 (2017), 186.
[66] H. Whitney, *Geometric Integration Theory*, Princeton University Press, 1957.
[67] A. Bossavit, *Whitney forms: A class of finite elements for three-dimensional computations in electromagnetism*, IEE Proc. A Phys. Sci. Measurement Instrum. Manage. Educ. Rev., 135 (1988), pp. 493–500.
[68] F. de Goes, M. Desbrun, and Y. Tong, *Vector field processing on triangle meshes*, in Proceedings of ACM SIGGRAPH 2016, ACM, 2016, 27.
[69] F. de Goes, P. Memari, P. Mullen, and M. Desbrun, *Weighted triangulations for geometry processing*, ACM Trans. Graphics, 33 (2014), 28.
[70] M. P. do Carmo, *Differential Geometry of Curves and Surfaces*, Prentice-Hall, 1976.
[71] W. T. Tutte, *How to draw a graph*, Proc. London Math. Soc. (3), 13 (1963), pp. 743–767.
[72] U. Pinkall and K. Polthier, *Computing discrete minimal surfaces and their conjugates*, Exp. Math., 2 (1993), pp. 15–36.
[73] M. Reuter, S. Biasotti, D. Giorgi, G. Patanè, and M. Spagnuolo, *Discrete Laplace–Beltrami operators for shape analysis and segmentation*, Comput. Graph., 33 (2009), pp. 381–390.
[74] G. Jacquenot, *Fast InPolygon Detection MEX*, https://www.mathworks.com/matlabcentral/fileexchange/20754-fast-inpolygon-detection-mex.
[75] *AIM@SHAPE Shape Repository*, http://visionair.ge.imati.cnr.it/ontologies/shapes/.
[76] *Cart: Computer Software for Making Cartograms*, http://www-personal.umich.edu/~mejn/cart/.
[77] X. Gu, *Optimal Mass Transportation Map*, http://www3.cs.stonybrook.edu/~gu/software/omt/index.html.
[78] M. Rabinovich, *GitHub Repository for Scalable Locally Injective Mappings*, http://github.com/MichaelRabinovich/Scalable-Locally-Injective-Mappings.
[79] A. Gray, *Modern Differential Geometry of Curves and Surfaces with Mathematica*, CRC Press, 1998, pp. 163–165.