Lawrence Berkeley National Laboratory
LBL Publications

Title
Density-Equalizing Maps for Simply Connected Open Surfaces

Permalink
https://escholarship.org/uc/item/4km3k0ss

Journal
SIAM Journal on Imaging Sciences, 11(2)

ISSN
1936-4954

Authors
Choi, Gary PT
Rycroft, Chris H

Publication Date
2018

DOI
10.1137/17m1124796

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California

arXiv:1704.02525v1 [cs.CG] 8 Apr 2017

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 flattening maps with any prescribed density distribution. By varying the initial density distribution, a large variety of mappings 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

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 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 the global population, the income and the age-of-death at different regions. Colizza et al. [3] constructed a geographical representation of disease evolution in the United States for an 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 $\mathbb{R}^2$

*Paulson School of Engineering and Applied Sciences, Harvard University, MA 02138.
†Department of Mathematics, Lawrence Berkeley Laboratory, Berkeley, CA 94720.

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 boundary-aware convex flattening map of simply-connected open surfaces. The flattening map effectively preserves the curvature of the input surface boundary and serves as a good initial mapping 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.

(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 limitation of our current approach and possible future works.

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 et al. [11] developed the Cartodraw algorithm for producing contiguous cartograms. Gastner and Newman [1] proposed a cartogram production algorithm based on density diffusion. Keim et al. [12] proposed to use 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. 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 the existing parameterization algorithms, we refer the readers to the comprehensive surveys [13, 14, 15]. We highlight the works on surface parameterization in the recent decade. For conformal parameterization, recent advances include the discrete Ricci flow method [16, 17, 18] and the quasi-conformal composition [19, 20, 21, 22, 23]. For area-preserving parameterization, the state-of-the-art approaches are mainly based on Lie advection of differential 2-forms [24] and optimal mass transport [25, 26]. Recently, Nadeem et al. [27] proposed an algorithm for achieving spherical parameterization with controllable area distortion.

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] is outlined as follows. For the rest of the paper, we denote the method by Gastner and Newman [1] as *GN*. 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

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

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, Gastner and Newman introduce 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. By defining the density at the sea to be the average density of the region of interest, it can be ensured 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 of the abovementioned 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 cause long computational time. In this work, an algorithm that further enhances the abovementioned approach in the two aspects is proposed. The details of our proposed algorithm in described in the following section.

4. Our proposed method. We now describe our method for computing density-equalizing maps of simply-connected open surfaces in $\mathbb{R}^3$. Let S be a simply-connected open surface in $\mathbb{R}^3$ and 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 Secs. 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 triangular 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 is 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 arc-length parameterized curve $\gamma = \gamma(t) : [0, l_\gamma] \to \mathbb{R}^3$, where l_γ is the total arc length 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. We have the following important theorem.

THEOREM 4.1 (Fundamental theorem of space curves [28]). *The curve γ is completely determined (up to rigid motion) by its curvature κ_γ and its torsion τ_γ.*

Motivated by the above theorem, 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 [28],

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

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

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

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 [28]:

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

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

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

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 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 by

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

In fact, φ becomes a simple closed convex plane curve under this adjustment. The proof is provided in the appendix.

The algorithm is summarized in Algorithm 1. After obtaining the simple closed convex plane curve φ, we can use it as a convex 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 [29]. To give an overview of the method, we first introduce the concept of 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 triangular mesh into account and neglects the geometry of it. It was proved by Tutte [29] 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

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 anti-clockwise 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 $\varphi(s) \leftarrow \varphi(s) - \frac{s}{l_\gamma}\left(\varphi(l_\gamma) - \varphi(0)\right)$;

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. Using our curvature-based flattened curve φ as the convex boundary constraint, a bijective Tutte flattening map $\phi : S \to P$ can be easily obtained as described in Algorithm 2.

Algorithm 2: Curvature-based Tutte flattening map

Input: A simply-connected open surface S in $\mathbb{R}^3$.
Output: A curvature-based flattening map $\phi : S \to \mathbb{R}^2$.

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 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;

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 et al. [30] 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 4.1. The minimization of the Chi energy aims to find a locally authalic mapping $\phi : S \to \mathbb{R}^2$ that preserves

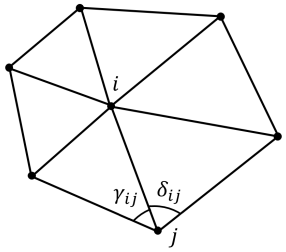

FIG. 4.1. *The angles γ_{ij} and δ_{ij} in the locally authalic Chi energy.*

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 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

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

The curvature-based locally authalic flattening map is summarized in Algorithm 3.

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;

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

(i) The boundary is flattened as a convex closed planar curve. The convex boundary constraint leads to a bijective flattening map of the surface using our proposed algorithm.

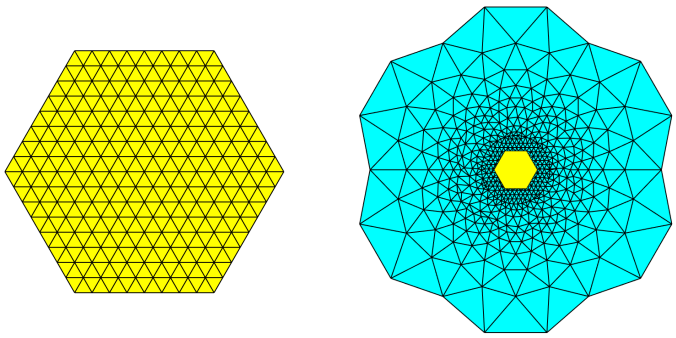

FIG. 4.2. *Illustration of our algorithm for constructing the sea. Left: the initial flattening map. We put it inside the unit circle and fill up the gap with uniformly distributed points, and then reflect the entire region along the circle to construct the sea. Right: the sea constructed (in cyan) and the initial flattening map (in yellow).*

(ii) The computation of the flattening maps is highly efficient. Both algorithms only involve solving one complex linear system without any iterative procedures.
(iii) Unlike other conventional parameterizations such as conformal parameterizations, our curvature-based flattening maps result in a relatively uniform distribution of vertices on $\mathbb{C}$ and avoid shrinking particular regions. The diffusion process can then be more accurately executed.

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. Setting the initial density at the sea as the mean density ensures a proper deformation of the area of interest, and 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 abovementioned 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

$$g(z) = \frac{1}{\bar{z}}. \tag{4.17}$$

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

$$\widetilde{\mathcal{V}} = \{z\}_{z\in\mathbb{D}_T} \cup \left\{\frac{1}{\bar{z}}\right\}_{z\in\mathbb{D}_T\setminus\partial(\mathbb{D}_T)}, \tag{4.18}$$

$$\widetilde{\mathcal{F}} = \mathcal{F} \cup \left\{\left[\frac{1}{\bar{z}_i}, \frac{1}{\bar{z}_j}, \frac{1}{\bar{z}_k}\right] : [z_i, z_j, z_k] \in \mathcal{F}\right\}, \tag{4.19}$$

and

$$\widetilde{\mathcal{E}} = \{[z_i, z_j] : [z_i, z_j] \text{ is an edge of a face } T \in \widetilde{\mathcal{F}}\}. \tag{4.20}$$

To get rid of the extremely large triangles at the outermost part of the glued mesh, we perform a simple truncation by removing the part far away from the unit disk $\mathbb{D}$. In practice, we remove all vertices and faces of $\widetilde{S}$ outside $\{z : |z| > 5\}$. 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.

Now we have constructed a natural complement surrounding our region of interest in $\mathbf{r}_0$. We proceed to set up the density distribution at the complement part. As suggested in GN, the density at the complement part should equal the mean density at the interior part. For every face $T \in \widetilde{\mathcal{F}} \setminus \mathcal{F}$, we set

$$\rho^{\widetilde{\mathcal{F}}}(T) = \text{mean}_{T'\in\mathcal{F}}\rho^{\mathcal{F}}(T'). \tag{4.21}$$

This completes our construction of the sea. The above procedures are summarized in Algorithm 4. An graphical illustration of the construction is shown in Figure 4.2.

Algorithm 4: Construction of sea via reflection.

Input: An initial flattening map $\mathbf{r}_0$.
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 $g(z) = \frac{1}{\bar{z}}$ 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 $\{z : |z| > 5\}$;
7 Rescale $\mathbf{r}_0$ to restore the size of the flattening map;

We now highlight the advantages of our construction of sea. One advantage of our construction is that the mesh size of the constructed sea is adaptive. Unlike the approach in GN, which used an 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 $\mathbb{D}$. It can be observed that under the reflection $z \mapsto \frac{1}{\bar{z}}$, we have

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

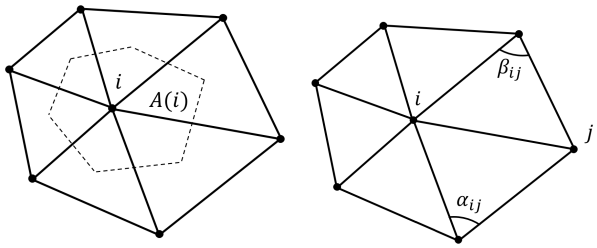

FIG. 4.3. *Left: the vertex area $A(i)$ of a vertex i. Right: the two angles α_{ij} and β_{ij} opposite to the edge $[i, j]$.*

This implies that if z_1, z_2 are located near the origin, the distance between the reflected points $\frac{1}{\overline{z_1}}$ and $\frac{1}{\overline{z_2}}$ will satisfy

$$\left|\frac{1}{\overline{z_1}} - \frac{1}{\overline{z_2}}\right| \gg |z_1 - z_2|, \tag{4.23}$$

since $|z_1 z_2| \ll 1$. On the other hand, if z_1, z_2 are located near the unit circle $\mathbb{S}^1$, we have

$$\left|\frac{1}{\overline{z_1}} - \frac{1}{\overline{z_2}}\right| \approx |z_1 - z_2|, \tag{4.24}$$

since $|z_1 z_2| \approx 1$.

One important consequence of the above observation is that 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 reducing 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.

Finally, note that the sea is constructed simply for the diffusion process and we are only interested in the interior region. In the following discussions, by an abuse of notation, we continue using the alphabets $\mathcal{V}$, $\mathcal{F}$ without tilde whenever referring to the discrete mesh structure.

4.3. Iterative scheme for producing density-equalizing maps. Given any simply-connected open triangular 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 ρ on each triangle element of the flattened map by $\frac{\text{Given population}}{\text{Area of the triangle}}$. As introduced before, after constructing the adaptive sea surrounding the map, we extend the density ρ to the whole domain by setting ρ at the sea to be the mean density at the original flattened map. In this subsection, we develop an iterative scheme for deforming the flattened map based on density diffusion.

To solve the diffusion equation on triangular meshes, one important issue is to discretize the Laplacian. Let $u : \mathcal{V} \to \mathbb{R}$ be a function. To compute the Laplacian of u at every vertex i, we make use of the cotangent Laplacian formulation [31]

$$\Delta u(i) = \frac{1}{2A(i)} \sum_{j \in \mathcal{N}(i)} (\cot \alpha_{ij} + \cot \beta_{ij}) \, (u(j) - u(i)), \tag{4.25}$$

where $A(i)$ is the vertex area of the vertex i, and α_{ij} and β_{ij} are the two angles opposite to the edge $[i, j]$. More specifically,

$$A(i) = \frac{1}{3} \sum_{T \in \{T \in \mathcal{F} : i \text{ is a vertex of the triangle } T\}} \text{Area}(T). \tag{4.26}$$

It is easy to observe that

$$\sum_{i \in \mathcal{V}} A(i) = \sum_{T \in \mathcal{F}} \text{Area}(T). \tag{4.27}$$

This shows that the vertex area is a good discretization of the total surface area at the vertex set. The graphical illustrations of $A(i)$ and α_{ij}, β_{ij} are given in Figure 4.3.

Note that the density ρ is originally defined on the triangular faces while the above $|\mathcal{V}| \times |\mathcal{V}|$ Laplacian is only applicable for vertices. To handle this discrepancy, we develop a natural transition between the value of ρ on triangular faces and that on vertices. Let $\rho^{\mathcal{F}}$ and $\rho^{\mathcal{V}}$ be respectively the value of ρ on faces and that on vertices in the form of column vectors. Given $\rho^{\mathcal{V}}$ on the vertices, the discretization $\rho^{\mathcal{F}}$ on the triangular faces can be obtained by considering

$$\rho^{\mathcal{F}} = M^{\mathcal{V}\mathcal{F}} \rho^{\mathcal{V}}, \tag{4.28}$$

where $M^{\mathcal{V}\mathcal{F}}$ is a $|\mathcal{F}| \times |\mathcal{V}|$ transition matrix defined by

$$M_{ij}^{\mathcal{V}\mathcal{F}} := \begin{cases} \frac{1}{3} & \text{if the } i\text{-th triangle contains the } j\text{-th vertex,} \\ 0 & \text{otherwise.} \end{cases} \tag{4.29}$$

Similarly, given $\rho^{\mathcal{F}}$ on the triangular faces, the discretization $\rho^{\mathcal{V}}$ on the vertices can be obtained by

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

where $M^{\mathcal{F}\mathcal{V}}$ is a $|\mathcal{V}| \times |\mathcal{F}|$ transition matrix. This time, we denote

$$\widetilde{M}_{ij}^{\mathcal{F}\mathcal{V}} := \begin{cases} 1 & \text{if the } j\text{-th triangle contains the } i\text{-th vertex,} \\ 0 & \text{otherwise,} \end{cases} \tag{4.31}$$

and define

$$M^{\mathcal{F}\mathcal{V}} := \begin{pmatrix} \widetilde{M}_{1,:}^{\mathcal{F}\mathcal{V}} / \| \widetilde{M}_{1,:}^{\mathcal{F}\mathcal{V}} \|_0 \\ \widetilde{M}_{2,:}^{\mathcal{F}\mathcal{V}} / \| \widetilde{M}_{2,:}^{\mathcal{F}\mathcal{V}} \|_0 \\ \vdots \\ \widetilde{M}_{|\mathcal{F}|,:}^{\mathcal{F}\mathcal{V}} / \| \widetilde{M}_{|\mathcal{F}|,:}^{\mathcal{F}\mathcal{V}} \|_0 \end{pmatrix}. \tag{4.32}$$

It is easy to check that

$$M^{\mathcal{FV}} M^{\mathcal{VF}} = |\mathcal{F}|, \tag{4.33}$$

and

$$M^{\mathcal{VF}} M^{\mathcal{FV}} = |\mathcal{V}|. \tag{4.34}$$

Therefore, the two operators $M^{\mathcal{FV}}$ and $M^{\mathcal{VF}}$ are consistent with each other.

After discretizing the Laplacian, we propose the following semi-discrete backward Euler method for solving the diffusion equation (3.3):

$$\frac{\rho_n^{\mathcal{V}} - \rho_{n-1}^{\mathcal{V}}}{\delta t} = \Delta_{n-1} \rho_n^{\mathcal{V}}. \tag{4.35}$$

Here $\rho_n^{\mathcal{V}}$ is the value of ρ on the vertices at the n-th iteration, Δ_n is the cotangent Laplacian of the deformed map $\mathbf{r}_n$, and δt is the time step for the iterations. By a simple rearrangement, the above equation is equivalent to

$$\rho_n^{\mathcal{V}} = (I - \delta t \Delta_{n-1})^{-1} \rho_{n-1}^{\mathcal{V}}. \tag{4.36}$$

Note that the above semi-discrete backward Euler method is unconditionally stable and ensures the convergence of our algorithm. Also, the cotangent Laplacian Δ_n is a symmetric positive definite matrix and hence (4.36) 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 discrete the gradient operator ∇. Consider the face-based discretization $(\nabla\rho)_n^{\mathcal{F}}(T)$ defined on every triangle element $T = [i, j, k]$ at the n-th iteration. Note that $(\nabla\rho)_n^{\mathcal{F}}(T)$ should satisfy

$$\begin{cases} \langle (\nabla\rho)_n^{\mathcal{F}}(T), e_{jk} \rangle = \rho_n^{\mathcal{V}}(k) - \rho_n^{\mathcal{V}}(j), \\ \langle (\nabla\rho)_n^{\mathcal{F}}(T), e_{ki} \rangle = \rho_n^{\mathcal{V}}(i) - \rho_n^{\mathcal{V}}(k), \\ \langle (\nabla\rho)_n^{\mathcal{F}}(T), e_{ij} \rangle = \rho_n^{\mathcal{V}}(j) - \rho_n^{\mathcal{V}}(i), \\ \langle (\nabla\rho)_n^{\mathcal{F}}(T), N \rangle = 0, \end{cases} \tag{4.37}$$

where $e_{ij} = [i, j], e_{jk} = [j, k], e_{ki} = [k, i]$ are the three directed edges of T in the form of vectors, and N is a unit normal vector of T. Since

$$e_{jk} + e_{ki} + e_{ij} = 0, \tag{4.38}$$

the third equation in (4.37) automatically follows from the first two equations. Note that (4.37) can be solved analytically with the solution

$$(\nabla\rho)_n^{\mathcal{F}}(T) = -\frac{1}{2\text{Area}(T)} N \times \left(\rho_n^{\mathcal{V}}(i) e_{jk} + \rho_n^{\mathcal{V}}(j) e_{ki} + \rho_n^{\mathcal{V}}(k) e_{ij} \right). \tag{4.39}$$

This gives us an accurate approximation of the gradient operator on triangulated surfaces.

Note that the above approximation is developed on the triangle elements but not on the vertices. For the face-to-vertex conversion, we again make use of a matrix multiplication. Note that the previously developed matrices $M^{\mathcal{FV}}$ and $M^{\mathcal{VF}}$ are purely combinatorial as the directions are not important in their uses. By contrast, the directions are important for computing the gradient operator. Therefore, we need to

take the geometry of the mesh into account in designing the conversion matrix for the gradient operator. To emphasize the weight of different directions in the conversion, we use the triangle area as a weight function. More specifically, we denote

$$\widetilde{W}_{ij}^{\mathcal{FV}} = \begin{cases} \text{Area}(T_j) & \text{if the } j\text{-th triangle } T_j \text{ contains the } i\text{-th vertex,} \\ 0 & \text{otherwise.} \end{cases} \tag{4.40}$$

Then we can define

$$W^{\mathcal{FV}} := \begin{pmatrix} \widetilde{W}_{1,:}^{\mathcal{FV}} / \|\widetilde{W}_{1,:}^{\mathcal{FV}}\|_1 \\ \widetilde{W}_{2,:}^{\mathcal{FV}} / \|\widetilde{W}_{2,:}^{\mathcal{WV}}\|_1 \\ \vdots \\ \widetilde{W}_{|\mathcal{F}|,:}^{\mathcal{FV}} / \|\widetilde{W}_{|\mathcal{F}|,:}^{\mathcal{FV}}\|_1 \end{pmatrix}. \tag{4.41}$$

With this weighted face-to-vertex conversion matrix, we have

$$(\nabla\rho)_n^{\mathcal{V}} := W^{\mathcal{FV}}(\nabla\rho)_n^{\mathcal{F}}. \tag{4.42}$$

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 (4.36) and compute the induced gradient $(\nabla\rho)_n^{\mathcal{V}}$ based on the abovementioned procedures. Then, we deform the map by

$$\mathbf{r}_n = \mathbf{r}_{n-1} + \delta t(\nabla\rho)_n^{\mathcal{V}}. \tag{4.43}$$

For the stopping criterion, we consider the quantity $\frac{\text{sd}(\rho_n^{\mathcal{F}})}{\text{mean}(\rho_n^{\mathcal{F}})}$. Note that the standard deviation $\text{sd}(\rho_n^{\mathcal{F}})$ measures the dispersion of the updated density $\rho_n^{\mathcal{F}}$, and we normalize it using $\text{mean}(\rho_n^{\mathcal{F}})$ to remove the effect of arbitrary scaling of $\rho_n^{\mathcal{F}}$. Also, it is easy to note that $\frac{\text{sd}(\rho_n^{\mathcal{F}})}{\text{mean}(\rho_n^{\mathcal{F}})} = 0$ 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 on the diffusion equation (3.3), an appropriate dimension of δt would be L^2. Also, note that δt should be independent to the magnitude of ρ. Therefore, a reasonable choice of δt is

$$\delta t = \min\left\{\frac{\min(\rho_0^{\mathcal{F}})}{\text{mean}(\rho_0^{\mathcal{F}})}, \frac{\text{mean}(\rho_0^{\mathcal{F}})}{\max(\rho_0^{\mathcal{F}})}\right\} \times \text{Area}(S). \tag{4.44}$$

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. Algorithm 5 summarizes our proposed method for producing density-equalizing maps of simply-connected open surfaces.

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.

Algorithm 5: Density-equalizing map 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 $f : S \to \mathbb{R}^2$.

if *S is planar* **then**
  Set $\mathbf{r}_0 = S$;
else
  Compute a curvature-based flattening map $\phi : S \to \mathbb{C}$ using Algorithm 2 or Algorithm 3. 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 $\mathbf{r}_0$;
Update $\mathbf{r}_0$ with an adaptive sea constructed using Algorithm 4;
Extend $\rho_0^{\mathcal{F}}$ to the whole domain by setting $\rho_0^{\mathcal{F}}$ at the sea to be the mean of the original $\rho_0^{\mathcal{F}}$;
Compute $\rho_0^{\mathcal{V}} = M^{\mathcal{FV}} \rho_0^{\mathcal{F}}$;
Set $\delta t = \min\left\{ \frac{\min(\rho_0^{\mathcal{F}})}{\text{mean}(\rho_0^{\mathcal{F}})}, \frac{\text{mean}(\rho_0^{\mathcal{F}})}{\max(\rho_0^{\mathcal{F}})} \right\} \times \text{Area}(S)$;
Set $n = 0$;
repeat
  Update $n = n + 1$;
  Solve $\rho_n^{\mathcal{V}} = (I - \delta t \Delta_{n-1})^{-1} \rho_{n-1}^{\mathcal{V}}$;
  Compute the face-based discrete gradient $(\nabla\rho)_n^{\mathcal{F}}(T) = -\frac{1}{2\text{Area}(T)} N \times \left(\rho_n^{\mathcal{V}}(i) e_{jk} + \rho_n^{\mathcal{V}}(j) e_{ki} + \rho_n^{\mathcal{V}}(k) e_{ij}\right)$;
  Perform the conversion $(\nabla\rho)_n^{\mathcal{V}} = W^{\mathcal{FV}} (\nabla\rho)_n^{\mathcal{F}}$;
  Update $\mathbf{r}_n = \mathbf{r}_{n-1} + \delta t (\nabla\rho)_n^{\mathcal{V}}$;
  Compute $\rho_n^{\mathcal{F}} = M^{\mathcal{VF}} \rho_n^{\mathcal{V}}$;
until $\frac{sd(\rho_n^{\mathcal{F}})}{mean(\rho_n^{\mathcal{F}})} < \epsilon$;
Obtain $f(S) = \mathbf{r}_n \times \frac{\text{Area}(\mathbf{r}_0)}{\text{Area}(\mathbf{r}_n)}$;

(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

$$\frac{\text{Initial area}}{\text{Final area}} = \frac{\text{Given population}}{\text{Final area}} = \text{Density} = \text{Constant}. \tag{4.45}$$

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. The linear systems in our algorithm are solved using the backslash operator in MATLAB. All experiments are performed on a PC with Intel i7-6700K CPU and 16 GB RAM. All surfaces are discretized in the form of triangular meshes. In all experiments, the stopping parameter ϵ is set to be 10^{-3}. Some of the surface meshes are adapted from the AIM@SHAPE Shape Repository [32].

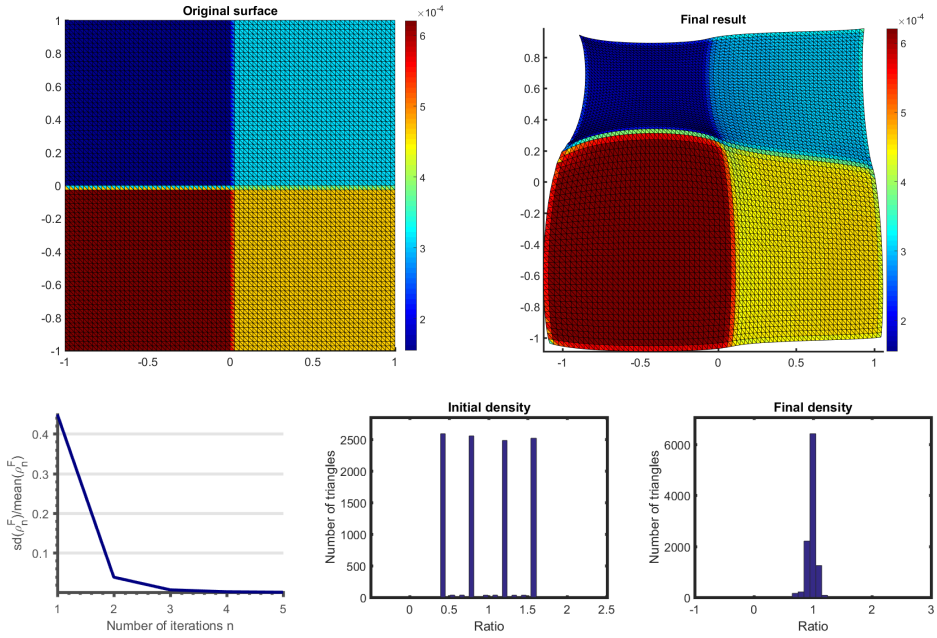

FIG. 5.1. *Density-equalization on a square. Top left: the initial shape colored with a given population distribution. Top right: the density-equalizing map colored with the final area of each triangle element. Bottom left: the values of* $\frac{sd(\rho_n^{\mathcal{F}})}{mean(\rho_n^{\mathcal{F}})}$. *Bottom middle: the histogram of the initial density* $\frac{Given\ population}{Initial\ area}$ *on each triangle element. Bottom right: the histogram of the final density* $\frac{Given\ population}{Final\ area}$ *on each triangle element.*

5.1. Examples of density-equalizing maps produced by our algorithm. We begin with two synthetic examples of regular polygons on $\mathbb{R}^2$. Figures 5.1 and 5.2 respectively show a square and a hexagon with a given population on every triangle element, and the density-equalizing results obtained by our proposed 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 $\frac{\text{sd}(\rho_n^{\mathcal{F}})}{\text{mean}(\rho_n^{\mathcal{F}})}$ show that 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 5.3 shows the initial surface and the mapping result obtained by our density-equalizing mapping algorithm. The plots indicate that the density is well equalized by our 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 algorithm should result in an area-preserving flattening map. Again, Algorithm 2 is used for the initialization of the density-equalization algorithm. Figure 5.4 shows the initial surface and the mapping result obtained by our density-equalizing mapping algorithm. The flattening map effectively preserves the area ratios. We compare our density-equalizing mapping result with the state-of-the-art conformal parameterization algorithms [30, 23]. Figure 5.5 shows the parameterization

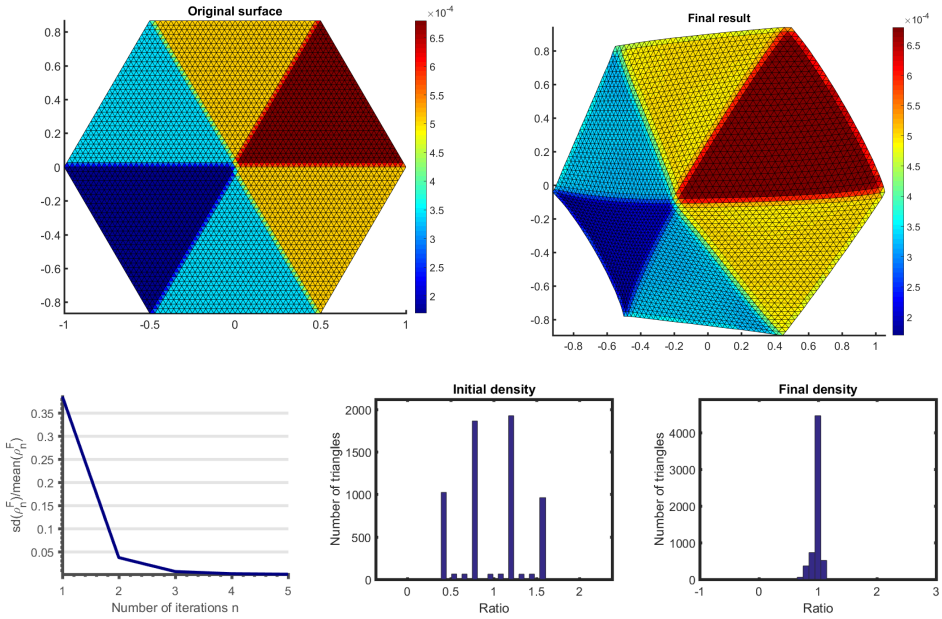

FIG. 5.2. *Density-equalization on a hexagon. Top left: the initial shape colored with a given population distribution. Top right: the density-equalizing map colored with the final area of each triangle element. Bottom left: the values of* $\frac{sd(\rho_n^F)}{mean(\rho_n^F)}$. *Bottom middle: the histogram of the initial density* $\frac{Given\ population}{Initial\ area}$ *on each triangle element. Bottom right: the histogram of the final density* $\frac{Given\ population}{Final\ area}$ *on each triangle element.*

results, whereby the peaks are substantially shrunk for conformal 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 algorithm.

Now consider computing the area-preserving mapping for a real surface mesh of a lion face in $\mathbb{R}^3$ using our 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 5.6 shows the initial surface and the mapping result obtained by our density-equalizing mapping 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 effects by changing the input population. Figure 5.7 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. It can be observed that the resulting density-equalizing maps are

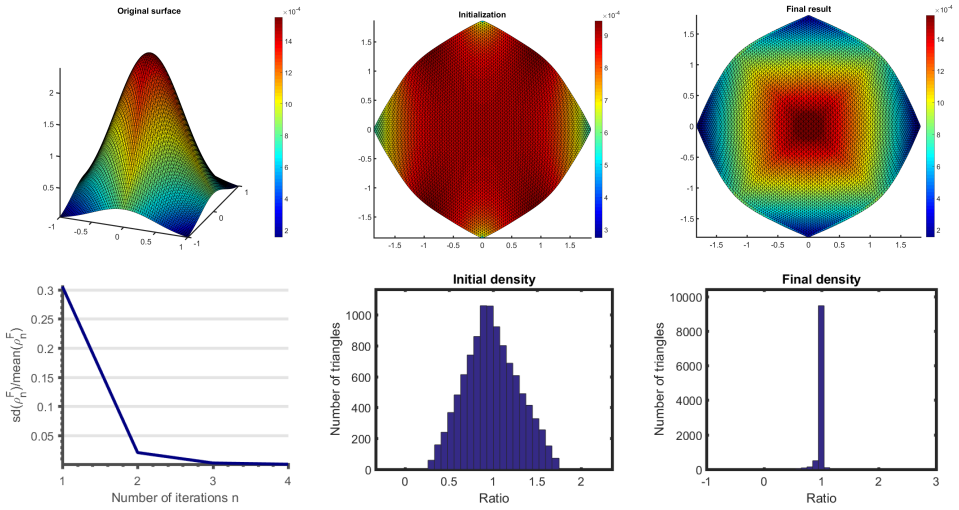

FIG. 5.3. *Density-equalizing map for a surface in* $\mathbb{R}^3$ *with Gaussian shape. Top left: the initial shape colored with a given population distribution. Top middle: the curvature-based Tutte flattening initialization colored with the area of each flattened triangle element. Top right: the final density-equalizing map colored with the final area of each triangle element. Bottom right: the values of* $\frac{sd(\rho_n^{\mathcal{F}})}{mean(\rho_n^{\mathcal{F}})}$. *Bottom middle: the histogram of the density* $\frac{Given\ population}{Initial\ flattened\ area}$ *on each flattened triangle element after the Tutte flattening initialization. Bottom right: the histogram of the density* $\frac{Given\ population}{Final\ area}$ *on each triangle element of the final result.*

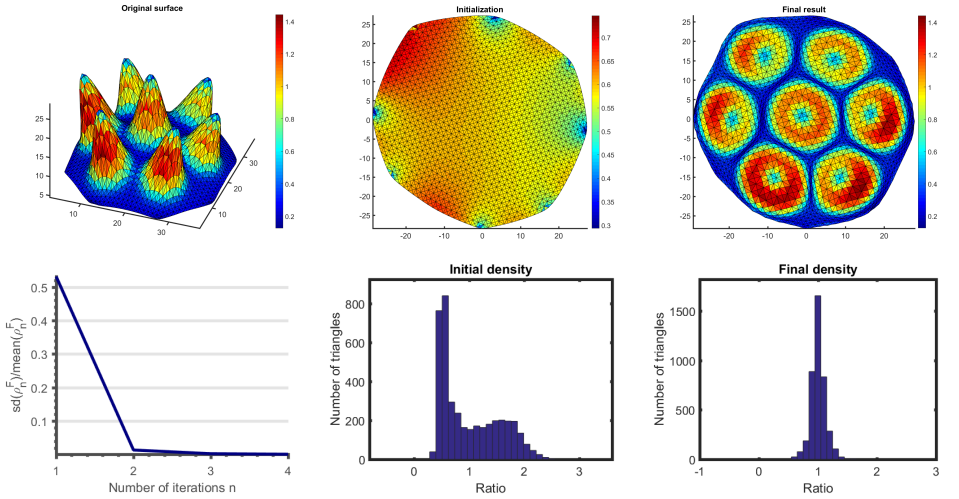

FIG. 5.4. *Area-preserving parameterization of a surface with multiple peaks in* $\mathbb{R}^3$. *Top left: the initial shape colored with the initial area of each triangle element. Top middle: the curvature-based Tutte flattening initialization colored with the area of each flattened triangle element. Top right: the final density-equalizing map colored with the final area of each triangle element. Bottom right: the values of* $\frac{sd(\rho_n^{\mathcal{F}})}{mean(\rho_n^{\mathcal{F}})}$. *Bottom middle: the histogram of the density* $\frac{Initial\ area}{Initial\ flattened\ area}$ *on each flattened triangle element after the Tutte flattening initialization. Bottom right: the histogram of the density* $\frac{Initial\ area}{Final\ area}$ *on each triangle element of the final result.*

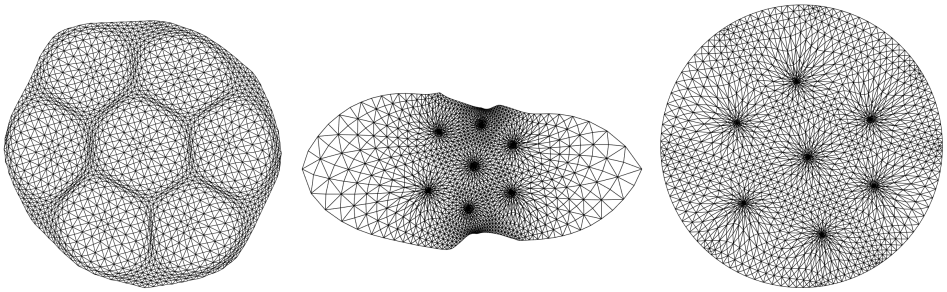

FIG. 5.5. *Comparison of different parameterization schemes for a surface with multiple peaks in* $\mathbb{R}^3$ *shown in Figure 5.4. Left: The area-preserving parameterization by our method. Middle: The free-boundary conformal parameterization by Desbrun et al. [30]. Right: The disk conformal parameterization by Choi and Lui [23].*

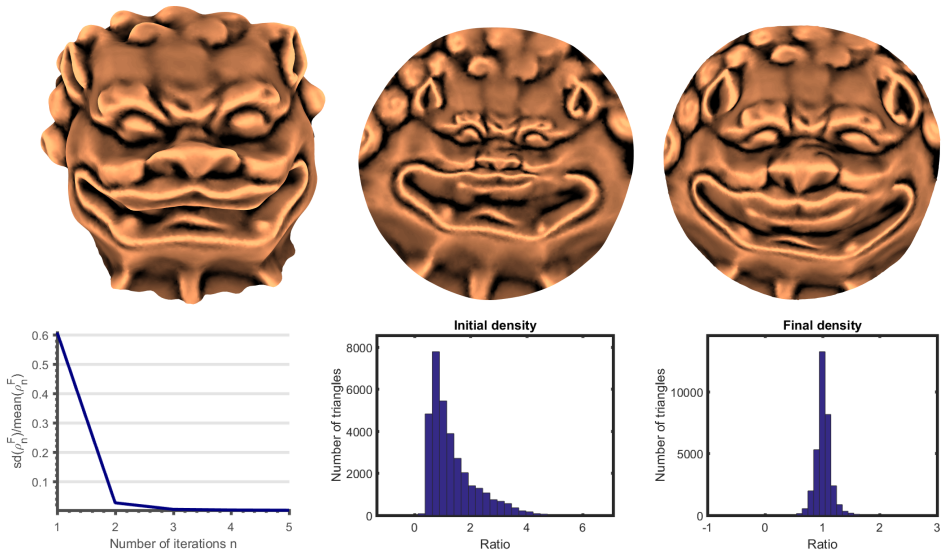

FIG. 5.6. *Area-preserving parameterization of a lion face in* $\mathbb{R}^3$*. Top left: the initial shape. Top middle: the curvature-based locally authalic flattening initialization. Top right: the final density-equalizing map. Bottom left: the values of* $\frac{sd(\rho_n^{\mathcal{F}})}{mean(\rho_n^{\mathcal{F}})}$*. Bottom middle: the histogram of the density* $\frac{Initial\ area}{Initial\ flattened\ area}$ *on each flattened triangle element after the flattening initialization. Bottom right: the histogram of the density* $\frac{Initial\ area}{Final\ area}$ *on each triangle element of the final result.*

respectively with the eyes and the nose magnified.

5.2. Numerical results of our algorithm. For a quantitative analysis, Table 5.1 lists the detailed statistics of the performance of our algorithm on a number of simply-connected open meshes. From the time spent and the number of iterations needed, it can be observed that the convergence of our proposed algorithm is fast. Also, the median and the inter-quartile range of the density show that the density is well equalized under our algorithm. The experiments reflect the efficiency and accuracy of our proposed algorithm.

We are also interested in analyzing the difference in the performance of our

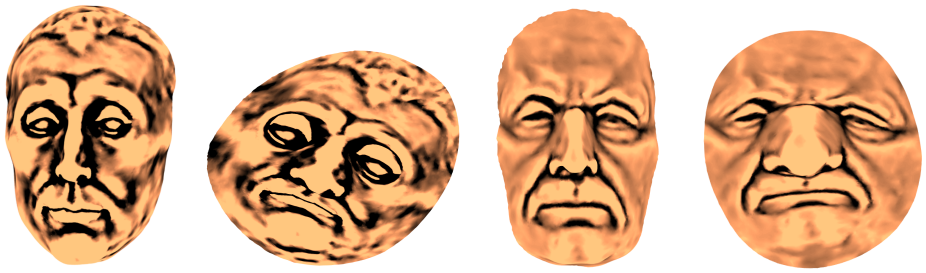

FIG. 5.7. *Density-equalizing flattening maps with different effects obtained by our proposed algorithm. Left: the Niccolò da Uzzano model and the density-equalizing flattening map with the eyes magnified. Right: the Max Planck model and the density-equalizing flattening map with the nose magnified.*

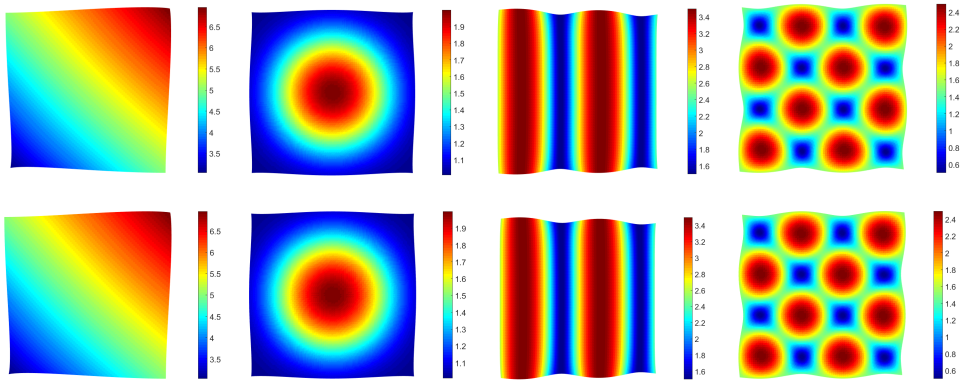

FIG. 5.8. *The density-equalizing maps produced by our proposed algorithm and GN with various input population functions. Each column shows a set of experimental results color-coded by the input population function as described in Table 5.2. Top row: the results by GN. Bottom row: the results by our method. It can be observed that our method produces results as accurate as those by GN.*

algorithm and GN with implementation available online [33]. 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. 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 5.8 shows several density-equalizing mapping results produced by the two methods. The statistics of the experiments are recorded in Table 5.2. With the accuracy well preserved, our method demonstrates an improvement on the computational time by over 60% when compared to GN.

We make a remark about the deformation of the sea under the density-equalizing process. Let r be the displacement of every point at the sea from the origin before the deformation, and $\Delta r = r_{\text{final}} - r$ be the change in displacement of the point under the density-equalizing process. Figure 5.9 shows several log–log plots of Δr against r outside the unit circle. We observe that Δr and r are related by the relationship $\Delta r \propto r^{-2}$ at the outer part of the sea. This suggests that setting a coarser sea at the outermost part does not affect the accuracy of the density-equalizing map.

Surface	No. of triangles	Time (s)	No. of iterations	Median of density	IQR of density
Square	10368	0.9858	5	1.0126	0.0799
Hexagon	6144	0.4394	5	1.0227	0.0556
Gaussian	10368	0.8667	4	1.0070	0.0307
Peaks	4108	0.2323	4	1.0024	0.1325
Lion	33369	2.4075	5	1.0220	0.1277
Niccolò da Uzzano	25900	3.0077	8	1.0305	0.0823
Max Planck	26452	3.2395	11	1.0252	0.0633
Human face	6912	1.2356	6	1.0084	0.0571
US Map (Romney)	46587	10.8280	3	1.0027	0.0146
US Map (Obama)	46587	12.8330	4	0.9998	0.0147
US Map (Trump)	46587	12.5154	4	1.0024	0.0176
US Map (Clinton)	46587	12.8733	4	1.0003	0.0248

TABLE 5.1

The performance of our 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 $\frac{\textit{Given population}}{\textit{Final area}}$.

Input population	Time by GN (s)	Time by our method (s)	Map difference
$5+\frac{(x-\bar{x})+(y-\bar{y})}{50}$	4.843	1.753	0.0009
$1+e^{-\frac{(x-\bar{x})^2+(y-\bar{y})^2}{1000}}$	4.452	1.501	0.0015
$2.5+\sin\frac{\pi(x-\bar{x})}{25}$	4.959	1.776	0.0013
$1.5+\sin\frac{\pi(x-\bar{x})}{25}\sin\frac{\pi(y-\bar{y})}{25}$	4.592	1.488	0.0026

TABLE 5.2

Comparing the performance of our algorithm and GN deployed on a 100×100 *square mesh with various input population functions. Here, the map difference is given by* $\textit{mean}\left(\frac{|z_{prev}-z_{ours}|}{\textit{side length of square}}\right)$, *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.*

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. Similar to GN, our density-equalizing mapping 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

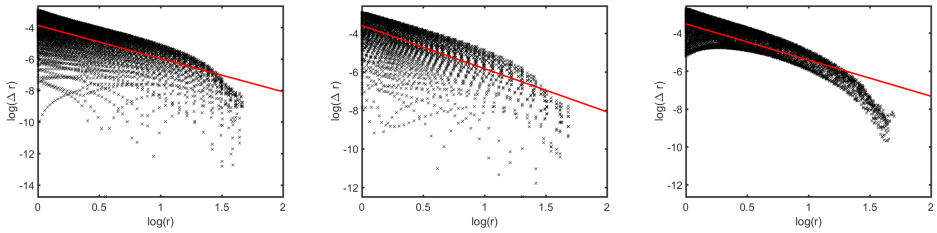

FIG. 5.9. *The log–log plot of the displacement of the sea under our density-equalizing algorithm. To study the effect at the outer sea, only the region outside the unit circle is considered. The x-axis represents the logarithm of the displacement of every point at the sea from the origin. The y-axis 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.*

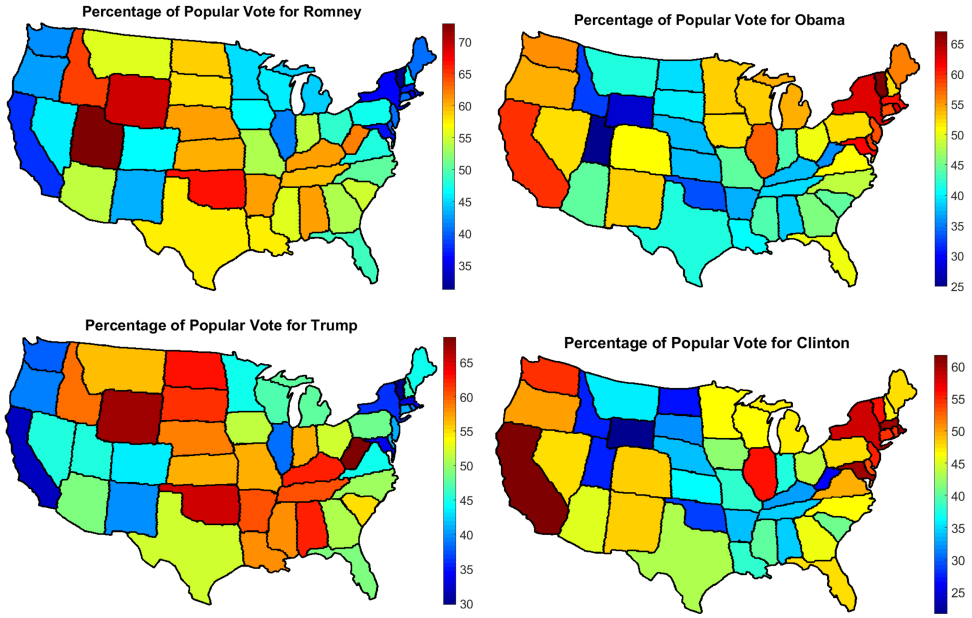

FIG. 6.1. *Percentage of popular vote in each state visualized on density-equalizing US maps (only including the contiguous 48 states). The triangulations are set to be transparent for enhancing the visual quality.*

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 6.1 shows the density-equalizing results. It can be observed that 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 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.

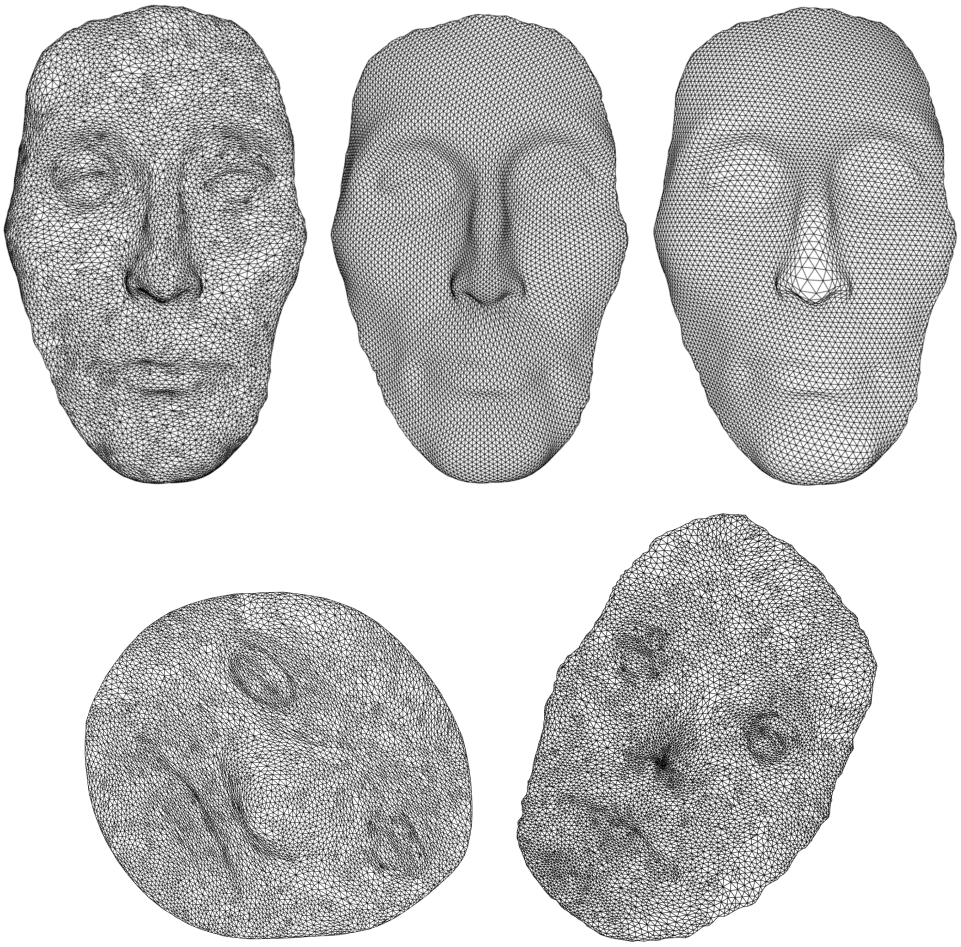

FIG. 6.2. *Remeshing a human face. Top left: the original human face. Top middle: the remeshing result via our parameterization. Top right: the remeshing result via the free-boundary conformal parameterization by Desbrun et al. [30]. Bottom left: the density-equalizing parameterization by our algorithm. Bottom right: the free-boundary conformal parameterization by Desbrun et al. [30].*

For Trump, California has further shrunk on the map while for Clinton, it has further expanded on the map. Another example is West Virginia. It can be observed that 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 election shows the usefulness of our density-equalizing mapping algorithm in data visualization.

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 $\mathcal{T}$. 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 details at a region of S, we can set a larger population at 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 6.2 shows an example of remeshing a triangulated human face using the abovementioned 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 [30]. 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. 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 triangular 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 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.

Since the density diffusion process is solved on a triangular mesh, the triangle quality affects the accuracy of the discretization and hence the final density-equalization result. If the triangular 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 affects the accuracy of the subsequent results. In other words, triangle meshes with moderate triangle quality and input population are desired. Besides, for surfaces in $\mathbb{R}^3$ with a highly tubular shape and with the boundary lying at one end, the flattening step may causes extremely squeezed regions on the planar domain. In this case, the accuracy of the subsequent computations for density equalization may be affected.

Our current work primarily focuses on simply-connected surfaces in $\mathbb{R}^3$, but 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.

Acknowledgments. This work was partially supported by a fellowship from the Croucher Foundation (GPT Choi).

REFERENCES

[1] M. T. Gastner and M. E. J. Newman. *Diffusion-based method for producing density-equalizing maps*. Proceedings of the National Academy of Sciences of the United States of America, Volume 101, Number 20, pp. 7499–7504, 2004.

[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*. Proceedings of the National Academy of Sciences of the United States of America, 103(7), pp. 2015–2020, 2006.

[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*. Proceedings of the National Academy of Sciences, 105(Supplement 1), pp. 11466–11473, 2008.

[5] K. S. Gleditsch and M. D. Ward, *Diffusion and the international context of democratization*. International organization, 60(04), pp. 911–933, 2006.

[6] A. Mislove, S. Lehmann, Y. Y. Ahn, J. P. Onnela, and J. N. Rosenquist. *Understanding the demographics of Twitter users*. ICWSM, 11, 5th, 2011.

[7] A. Vanasse, M. Demers, A. Hemiari, and J. Courteau, *Obesity in Canada: where and how many?*. International Journal of Obesity, 30(4), pp. 677–683, 2006.

[8] R. K. Pan, K. Kaski, and S. Fortunato, *World citation and collaboration networks: uncovering the role of geography in science*. Scientific Reports, 2, 2012.

[9] D. Dorling, *Area cartograms: their use and creation*. Concepts and Techniques in Modern Geography series, Number 59, University of East Anglia: Environmental Publications, 1996.

[10] H. Edelsbrunner and R. Waupotitsch, *A combinatorial approach to cartograms*. Computational Geometry: Theory and Applications, Volume 7, pp. 343–360, 1997.

[11] D. A. Keim, S. C. North, and C. Pans, *Cartodraw: A fast algorithm for generating contiguous cartograms*. IEEE Transactions on Visualization and Computer Graphics, Volume 10, Issue 1, pp. 95–110, 2004.

[12] D. A. Keim, C. Panse, and S. C. North, *Medial-axis-based cartograms*. IEEE Computer Graphics and Applications, Volume 25, Issue 3, pp. 60–68, 2005.

[13] M. Floater and K. Hormann, *Surface parameterization: a tutorial and survey*. Advances in Multiresolution for Geometric Modelling, pp. 157–186, 2005.

[14] A. Sheffer, E. Praun, and K. Rose, *Mesh parameterization methods and their applications*. Foundations and Trends in Computer Graphics and Vision, Volume 2, Issue 2, pp. 105–171, 2006.

[15] K. Hormann, B. Lévy, and A. Sheffer, *Mesh parameterization: theory and practice*. Proceeding of ACM SIGGRAPH 2007 courses, Article Number 1, pp. 1–122, 2007.

[16] M. Jin, J. Kim, F. Luo, and X. Gu, *Discrete surface Ricci flow*. IEEE Transaction on Visualization and Computer Graphics, Volume 14, Number 5, pp. 1030–1043, 2008.

[17] Y.L. Yang, R. Guo, F. Luo, S. M. Hu, and X. F. Gu, *Generalized discrete Ricci flow*. Computer Graphics Forum, Volume 28, Issue 7, pp. 2005–2014, 2009.

[18] M. Zhang, R. Guo, W. Zeng, F. Luo, S.-T. Yau, and X. Gu, *The unified discrete surface Ricci flow*. Graphical Models, Volume 76, Issue 5, pp. 321–339, 2014.

[19] P. T. Choi, K. C. Lam, and L. M. Lui, *FLASH: fast landmark aligned spherical harmonic parameterization for genus-0 closed brain surfaces*. SIAM Journal on Imaging Sciences, Volume 8, Issue 1, pp. 67–94, 2015.

[20] P. T. Choi and L. M. Lui, *Fast disk conformal parameterization of simply-connected open surfaces*. Journal of Scientific Computing, Volume 65, Issue 3, pp.1065–1090, 2015.

[21] G. P. T. Choi, K. T. Ho, and L. M. Lui, *Spherical conformal parameterization of genus-0 point clouds for meshing*. SIAM Journal on Imaging Sciences, Volume 9, Issue 4, pp. 1582–1618, 2016.

[22] T. W. Meng, G. P. T. Choi, and L. M. Lui, *TEMPO: Feature-endowed Teichmüller extremal mappings of point clouds*. SIAM Journal on Imaging Sciences, Volume 9, Issue 4, pp. 1922–1962, 2016.

[23] G. P. T. Choi and L. M. Lui, *A linear formulation for disk conformal parameterization of simply-connected open surfaces*. Advances in Computational Mathematics, Forthcoming 2017.

[24] G. Zou, J. Hu, X. Gu, and J. Hua, *Authalic parameterization of general surfaces using Lie advection*. IEEE Transactions on Visualization and Computer Graphics, Volume 17, Number 12, pp. 2005–2014, 2011.

[25] X. Zhao, Z. Su, X. D. Gu, A. Kaufman, J. Sun, J. Gao, and F. Luo, *Area-preservation mapping using optimal mass transport*. IEEE Transactions on Visualization and Computer Graphics, Volume 19, Number 12, pp. 2838–2847, 2013.

[26] 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*. Computer Aided Geometric Design, Volume 46, pp. 76–91, 2016.
[27] S. Nadeem, Z. Su, W. Zeng, A. Kaufman, and X. Gu, *Spherical parameterization balancing angle and area distortions*. IEEE Transactions on Visualization and Computer Graphics, Volume PP, Number 99, pp. 1–14, 2016.
[28] M. P. do Carmo, *Differential geometry of curves and surfaces*. Prentice-Hall, 1976.
[29] W. T. Tutte, *How to draw a graph*. Proceedings of the London Mathematical Society, Volume 13, pp. 743–767, 1963.
[30] M. Desbrun, M. Meyer, and P. Alliez, *Intrinsic parameterizations of surface meshes*. Computer Graphics Forum, Volume 21, Number 3, pp. 209–218, 2002.
[31] U. Pinkall and K. Polthier, *Computing discrete minimal surfaces and their conjugates*. Experimental mathematics, Volume 2, Number 1, pp. 15–36, 1993.
[32] AIM@SHAPE Shape Repository. http://visionair.ge.imati.cnr.it/ontologies/shapes/
[33] Cart: Computer software for making cartograms. http://www-personal.umich.edu/~mejn/cart/
[34] A. Gray, *Modern differential geometry of curves and surfaces with Mathematica*. Boca Raton: CRC Press, pp. 163–165, 1998.

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

PROPOSITION 7.1. *Let $\varphi : [0, l_\gamma] \to \mathbb{R}^2$ be the arclength parameterized curve defined as in Sec. 4.1. Consider the new curve $\Phi : [0, l_\gamma] \to \mathbb{R}^2$ defined by*

$$\Phi(s) = \varphi(s) - \frac{s}{l_\gamma}\left(\varphi(l_\gamma) - \varphi(0)\right). \tag{7.1}$$

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

$$\|\varphi(b) - \varphi(a)\| \le \int_a^b \|\varphi'(s)\| ds = b - a, \tag{7.2}$$

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 non-negative for all $s \in [0, l_\gamma]$. Denote $\varphi(s) = (x(s), y(s))$ and $\Phi(s) = (X(s), Y(s))$. We have

$$\begin{aligned}
\Phi' &= (X', Y') \\
&= \left(x'(s) - \frac{1}{l_\gamma}\left(x(l_\gamma) - x(0)\right), y'(s) - \frac{1}{l_\gamma}\left(y(l_\gamma) - y(0)\right)\right) \\
&= \left(\cos\theta(s) - \frac{1}{l_\gamma}\left(x(l_\gamma) - x(0)\right), \sin\theta(s) - \frac{1}{l_\gamma}\left(y(l_\gamma) - y(0)\right)\right)
\end{aligned} \tag{7.3}$$

and

$$\Phi'' = (X'', Y'') = (x'', y'') = \left(-k_\varphi(s)\sin\theta(s), k_\varphi(s)\cos\theta(s)\right). \tag{7.4}$$

Hence, we have

$$X'Y'' - X''Y' = k_\varphi(s)\left(1 - \frac{\left(x(l_\gamma) - x(0)\right)\cos\theta(s) - \left(y(l_\gamma) - y(0)\right)\sin\theta(s)}{l_\gamma}\right). \tag{7.5}$$

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

$$\begin{aligned}
&(x(l_\gamma)-x(0))\cos\theta(s)-(y(l_\gamma)-y(0))\sin\theta(s)\\
\leq&\sqrt{(x(l_\gamma)-x(0))^2+(y(l_\gamma)-y(0))^2}\sqrt{\cos^2\theta(s)+\sin^2\theta(s)}\\
=&\sqrt{(x(l_\gamma)-x(0))^2+(y(l_\gamma)-y(0))^2}\\
=&\|\varphi(l_\gamma)-\varphi(0)\|\\
\leq&l_\gamma.
\end{aligned} \tag{7.6}$$

Here, the first inequality follows from the Cauchy–Schwarz inequality, and the second inequality follows from (7.2). Therefore, we have

$$1-\frac{(x(l_\gamma)-x(0))\cos\theta(s)-(y(l_\gamma)-y(0))\sin\theta(s)}{l_\gamma}\geq 0 \text{ for all } s\in[0,l_\gamma], \tag{7.7}$$

and it follows that

$$k_\Phi(s)=\frac{X'Y''-X''Y'}{(X'^2+Y'^2)^{3/2}}\geq 0 \text{ for all } s\in[0,l_\gamma]. \tag{7.8}$$

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

$$\int_0^{l_\varphi} k_\Phi(s)ds=2\pi n_\Phi, \tag{7.9}$$

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

$$\begin{aligned}
2\pi n_\Phi&=\int_0^{l_\varphi}\frac{k_\varphi(s)\left(1-\frac{(x(l_\gamma)-x(0))\cos\theta(s)-(y(l_\gamma)-y(0))\sin\theta(s)}{l_\gamma}\right)}{(X'^2+Y'^2)^{3/2}}ds\\
&\leq\left(\int_0^{l_\varphi}k_\varphi(s)ds\right)\max_{s\in[0,l_\gamma]}\left|\frac{\left(1-\frac{(x(l_\gamma)-x(0))\cos\theta(s)-(y(l_\gamma)-y(0))\sin\theta(s)}{l_\gamma}\right)}{(X'^2+Y'^2)^{3/2}}\right|\\
&=2\pi\max_{s\in[0,l_\gamma]}\left(\frac{1-\frac{(x(l_\gamma)-x(0))\cos\theta-(y(l_\gamma)-y(0))\sin\theta}{l_\gamma}}{\left(1+\frac{(x(l_\gamma)-x(0))^2+(y(l_\gamma)-y(0))^2}{l_\gamma}-\frac{2((x(l_\gamma)-x(0))\cos\theta+(y(l_\gamma)-y(0))\sin\theta)}{l_\gamma}\right)^{3/2}}\right).
\end{aligned} \tag{7.10}$$

Substituting $A=\frac{x(l_\gamma)-x(0)}{l_\gamma}$ and $B=\frac{y(l_\gamma)-y(0)}{l_\gamma}$, the above equation becomes

$$\begin{aligned}
&2\pi\max_{s\in[0,l_\gamma]}\left(\frac{1-A\cos\theta(s)+B\sin\theta(s)}{(1+A^2+B^2-2A\cos\theta(s)-2B\sin\theta(s))^{3/2}}\right)\\
=&2\pi\max_{s\in[0,l_\gamma]}\left(\frac{1-C\cos(\theta(s)+\eta)}{(1+C^2-2C\cos(\theta(s)-\eta)))^{3/2}}\right),
\end{aligned} \tag{7.11}$$

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

$$\max_{s\in[0,l_\gamma]}\left(\frac{1-C\cos(\theta(s)+\eta)}{(1+C^2-2C\cos(\theta(s)-\eta)))^{3/2}}\right)\leq\frac{1+C}{(1-C)^3}<2. \tag{7.12}$$

This implies that

$$2\pi n_{\Phi} < 4\pi \Rightarrow n_{\Phi} < 2. \tag{7.13}$$

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 [34]. From (7.8), we conclude that Φ is a simple closed convex curve. □