Mathematics

https://www.aimspress.com/journal/Math

AIMS Mathematics, 10(11): 25756–25790.
DOI: 10.3934/math.20251140
Received: 10 June 2025
Revised: 23 October 2025
Accepted: 28 October 2025
Published: 10 November 2025

Research article

Learning-based density-equalizing map

Yanwen Huang, Lok Ming Lui and Gary P. T. Choi*

Department of Mathematics, The Chinese University of Hong Kong, Hong Kong, China

* **Correspondence:** Email: ptchoi@cuhk.edu.hk.

Abstract: A density-equalizing map (DEM) serves as a powerful technique for creating shape deformations with the area changes reflecting an underlying density function. In recent decades, DEMs have found widespread applications in fields such as data visualization, geometry processing, and medical imaging. Traditional approaches to the DEM primarily rely on iterative numerical solvers for diffusion equations or optimization-based methods that minimize handcrafted energy functionals. However, these conventional techniques often face several challenges: they may suffer from limited accuracy, produce overlapping artifacts in extreme cases, and require substantial algorithmic redesign when extended from 2D to 3D, due to the derivative-dependent nature of their energy formulations. In this work, we proposed a novel learning-based density-equalizing mapping framework (LDEM) using deep neural networks. Specifically, we introduced a loss function that enforces density uniformity and geometric regularity, and utilized a hierarchical approach to predict the transformations at both the coarse and dense levels. Our method demonstrated superior density-equalizing and bijectivity properties compared to prior methods for a wide range of simple and complex density distributions, and can be easily applied to surface remeshing with different effects. Also, it generalizes seamlessly from 2D to 3D domains without structural changes to the model architecture or loss formulation. Altogether, our work opens up new possibilities for scalable and robust computation of density-equalizing maps for practical applications.

Keywords: density-equalizing map; convolutional neural networks; diffusion; surface remeshing; volumetric deformation
Mathematics Subject Classification: 68U05, 65D18, 76R50

1. Introduction

Density-equalizing maps (DEMs) [1] are spatial transformations that reallocate area in proportion to a given density distribution based on the principle of density diffusion. They were originally developed for cartography, in which it is common to create value-by-area cartograms for mapping and data visualization [2–4]. In particular, under a DEM, different regions in a geographical map are often distorted so that their areas in the new map reflect certain prescribed data values, yet the mapping ideally remains continuous and recognizable.

DEMs have broad applications across different domains. In cartography, they have been extensively used in recent decades to visualize data such as census results [1], international factors of democratization [5], housing wealth [6], epidemiological spread [7], climate warming [8], trends in information and communication technology [9], social media usage [10], and academic publication and citation [11, 12]. More recently, DEMs have also been found useful for many important tasks in science and engineering. For instance, in medical imaging, density-equalizing map algorithms have been applied to flatten anatomical surfaces or register medical images while preserving area-based quantities of interest [13]. In geometry processing, similar principles have been used for surface parameterization and remeshing [14], where a mesh is mapped to a parameter domain with controlled area changes. These examples highlight the utility of DEMs in creating maps or flattenings that faithfully follow the prescribed shape change effects.

Traditionally, DEMs are computed through iterative physics-based procedures. Gastner and Newman [1] introduced an algorithm for generating diffusion-based cartograms: the map is treated as a medium in which a variable (e.g., population density) diffuses over time, causing regions to expand or contract until the density is uniform. This approach solves a series of partial differential equations (PDEs) to incrementally deform the domain. Later, Choi and Rycroft [14] extended the density diffusion principle to deform 3D surfaces, flattening them into 2D maps while equalizing a prescribed surface density. Despite the effectiveness of iterative DEM algorithms, they have notable limitations. First, solving the diffusion (or related) equations can be computationally intensive and may require careful tuning for stability and accuracy. High-resolution maps demand many iterations for the density to equalize, and the result might require extra smoothing procedures to avoid extreme local distortions [1]. Moreover, ensuring a bijective (overlap-free) deformation is nontrivial—small mesh overlaps or fold-overs can occur if the deformation is too aggressive or the numerical method is not carefully constrained. Extending these methods to more complex domains is also challenging, typically necessitating custom adaptations of the algorithm [15, 16].

In this paper, we propose a learning-based density-equalizing mapping approach (abbreviated as LDEM) that addresses these challenges. Instead of solving PDEs from scratch for each new dataset, we train a convolutional neural network to learn the mapping that equalizes density. Our method produces high-accuracy density-equalizing mappings for a wide range of prescribed density distributions. Also, the learned model inherently strives to preserve bijectivity, yielding mappings without mesh overlaps. Furthermore, the framework is flexible and can be easily generalized from two dimensions to higher dimensions. Our key contributions are as follows:

(1) We introduce a novel method for the computation of density-equalizing maps using deep neural networks. In particular, we introduce a loss function involving both the density uniformity and geometric regularity of the mapping. Also, we follow a hierarchical approach to capture both deformations at both the coarse and dense levels.

(2) Our proposed method can handle a wider range of population distributions and achieve a higher mapping accuracy when compared to the traditional diffusion-based approaches, while maintaining low local geometric distortion and bijectivity.

(3) Our method can be easily applied for surface remeshing to achieve different desired remeshing effects.

(4) The proposed learning-based framework naturally extends from 2D domains to 3D domains with minimal changes, demonstrating the versatility of the approach.

The rest of this paper is organized as follows. In Section 2, we review previous work on density-equalizing map techniques, including the diffusion-based cartogram and the extension to triangulated surface maps. Section 3 provides the mathematical background, covering the diffusion equation and the principle of density-equalizing maps, the Beltrami coefficient for quantifying geometric distortion and bijectivity, and an overview of convolutional neural networks. Section 4 details our proposed learning-based DEM methodology, and Section 5 presents experimental results comparing our method with prior approaches. In Section 6, we demonstrate the effectiveness of our method for surface remeshing. In Section 7, we describe the extension of our method for 3D mapping problems. Finally, Section 8 concludes the paper and discusses future directions.

2. Previous work

Over the past few decades, various computational methods have been developed to construct density-equalizing maps. Below, we highlight two representative approaches and discuss related developments.

2.1. Diffusion-based cartogram

Conventional methods for generating contiguous cartograms (density-adjusted maps) often required iterative "rubber-sheet" deformations that could produce overlapping or hard-to-read maps [17–19]. In 2004, Gastner and Newman [1] addressed these issues by introducing a diffusion-based cartogram algorithm. In their method, the input density (e.g., population per region) is treated as a fluid that diffuses across the map. As the density flows from higher to lower concentration areas, the map is continuously deformed—regions with surplus density expand while those with deficit contract—in order to conserve the total "mass" in any region. Formally, the method solves the heat diffusion equation $\partial\rho/\partial t = \nabla^2\rho$ on the map domain, with appropriate boundary conditions, and concurrently advects the map coordinates using the density flux (ensuring that areas change in proportion to the diffusing mass). The process is run until a steady state is reached where ρ becomes uniform; at that point, the deformation of the map yields a density-equalized cartogram.

This diffusion-based approach produces contiguous, non-overlapping cartograms that largely preserve relative locality and shape, thereby improving readability over earlier methods [1]. Gastner and Newman illustrated the technique with examples ranging from electoral maps to disease incidence, showing that it avoided the axis-alignment biases and region overlaps that plagued previous algorithms. One practical consideration is that fine-grained density variations can lead to very local distortions in the map; in practice, a slight smoothing of the input density (e.g., via Gaussian blur) is often applied to maintain cartographic readability at the expense of small accuracy loss [1]. The original algorithm also involves computationally intensive PDE integration, typically requiring many time steps on a grid, which can be slow for high-resolution outputs. Subsequent work by Gastner et al. [20] introduced a faster, flow-based implementation that significantly accelerates cartogram generation. Nonetheless, the diffusion-based method set a new standard for contiguous cartograms, and its success spurred further extensions to new domains.

2.2. *Density-equalizing maps for surface flattening*

Choi and Rycroft [14] extended the computation of density-equalizing maps to 3D triangulated surfaces, enabling the flattening of a curved surface into the plane while redistributing area according to a given density function. Their method operates analogously to the planar case: a scalar density defined on the surface is allowed to diffuse over time, and the surface's parameterization (mapping to 2D) is iteratively updated to reflect the diffusion. In practice, the algorithm starts with an initial flattening of the surface (for example, a conformal map of the 3D surface onto a 2D domain) and defines a density ρ on the surface (such as one proportional to some scalar field or the area element). As ρ diffuses across the surface, the 2D coordinates of the mesh vertices are adjusted so that areas in the parameterization change in proportion to the mass flow. By the end of the process, the surface is flattened in such a way that regions with higher initial density occupy larger areas in the planar map. This allows, for instance, one to obtain an area-preserving parameterization by simply setting ρ based on the area element of the surface. Choi and Rycroft demonstrated applications of the surface DEM method in data visualization and surface remeshing, laying the groundwork for subsequent improvements in surface and volumetric domains.

While effective, the surface DEM algorithm shares some limitations with its planar predecessor. It requires solving a time-dependent PDE on a triangle mesh, which can be computationally heavy for complex or high-resolution surfaces. The method is inherently restricted to simply connected open surfaces and does not directly handle domains with other topologies. Additionally, large variations in the prescribed density can cause significant distortion of the mesh, and ensuring that the flattening remains bijective (without any fold-overs) is nontrivial. In practice, careful implementation and possibly mesh refinement are needed to minimize element inversion.

2.3. *Other related works*

Beyond the above approaches, there have been numerous refinements and extensions of DEM techniques. In cartography, researchers have also examined the usability and perception of cartograms; for instance, user studies have evaluated the effectiveness of contiguous area cartograms in conveying information [21]. On the algorithmic side, Gastner et al.'s flow-based method [20] significantly improved the runtime of diffusion cartograms, making high-resolution density-equalizing maps more practical. DEM algorithms have also been adapted to different domain topologies. Li and Aryana [22] proposed a diffusion-based density-equalizing map for spherical surfaces, effectively creating cartograms on the globe. More recently, the computation of surface density-equalizing maps has also been developed for genus-0 closed surfaces [23, 24] and surfaces with other topologies [15, 25–27]. In addition, volumetric extensions have been explored: Li and Aryana [28] developed a volumetric method for handling subsurface data. Choi and Rycroft [16] also presented a general density-equalizing mapping framework for volumetric datasets, deforming a 3D volume so that a given density becomes uniform throughout the interior. Each of these extensions addresses specific challenges (such as enforcing bijectivity or handling complex geometries) with tailored algorithmic modifications. However, all of the above methods still rely on solving physical diffusion equations or iterative optimizations for each new input, underscoring the need for more efficient solutions. This has motivated interest in data-driven approaches, as we propose in this paper, to learn the mapping function directly, bypassing the costly iterative solves for each instance.

3. Mathematical background

In this section, we review the key mathematical tools underpinning our proposed learning-based method for density-equalizing maps. These include the basic concepts of the diffusion equation and density-equalizing maps, the Beltrami coefficient in quasi-conformal theory, and convolutional neural networks.

3.1. Diffusion equation and density-equalizing maps

The diffusion equation models how a density distribution $\rho(\mathbf{x}, t)$ spreads over time:

$$\frac{\partial \rho}{\partial t}(\mathbf{x}, t) = \Delta \rho(\mathbf{x}, t), \quad \mathbf{x} \in \Omega, \tag{3.1}$$

where Δ is the Laplacian operator. We impose the no-flux boundary conditions

$$\nabla \rho \cdot \mathbf{n} = 0 \quad \text{on } \partial\Omega, \tag{3.2}$$

where $\mathbf{n}$ is the unit outward normal, so that mass is conserved within the domain.

In density-equalizing maps, given a density distribution ρ, the above diffusion equation is solved iteratively, with the density gradient driving the shape deformation of the domain:

$$\mathbf{u} = -\frac{\nabla \rho}{\rho}, \tag{3.3}$$

so that the displacement of the vertices in the domain satisfies

$$\mathbf{x}(t) = \mathbf{x}(0) + \int_0^t \mathbf{u}(\mathbf{x}(\tau), \tau) d\tau. \tag{3.4}$$

As $t \to \infty$, the solution converges to

$$\rho(\mathbf{x}, t) \;\longrightarrow\; \bar{\rho} = \frac{1}{|\Omega|} \int_\Omega \rho(\mathbf{x}, 0)\, d\mathbf{x}, \tag{3.5}$$

i.e., a uniform density. In other words, this uniformization drives the domain deformation so that different regions expand or contract based on ρ, yielding a transformation with prescribed area changes.

3.2. Beltrami coefficient

A quasi-conformal map $f : \Omega \subset \mathbb{C} \to \mathbb{C}$ satisfies the Beltrami equation

$$f_{\bar{z}} = \mu(z) f_z, \tag{3.6}$$

where f_z and $f_{\bar{z}}$ are the Wirtinger derivatives

$$f_z = \tfrac{1}{2}(f_x - i\, f_y), \quad f_{\bar{z}} = \tfrac{1}{2}(f_x + i\, f_y), \tag{3.7}$$

and $\mu(z)$ is a complex-valued function (called the Beltrami coefficient) with $||\mu||_\infty < 1$. In other words, we have

$$\mu(z) = \frac{f_{\bar{z}}}{f_z}. \tag{3.8}$$

Intuitively, a quasi-conformal map f maps infinitesimal circles to infinitesimal ellipses, with the local aspect ratio change expressed as

$$K(f) = \frac{1 + |\mu|}{1 - |\mu|}. \tag{3.9}$$

Therefore, $|\mu|$ can effectively measure the quasi-conformal distortion of a mapping, with $|\mu| = 0$ if and only if f is conformal. Also, any local fold-overs can be captured by a value of $|\mu| > 1$. In other words, $|\mu|$ also provides us with a simple way for assessing the bijectivity of the mapping results produced by our method.

3.3. *Convolutional neural network (CNN)*

In our proposed method, the convolutional neural network (CNN) is employed to approximate the density transformation. CNNs are a class of deep neural networks designed to process data with grid-like structures, such as images or volumetric data. A CNN applies a series of convolution operations, which compute the weighted sum of local neighborhoods, to extract hierarchical features from the input data [29].

Mathematically, a convolution operation between an input $I(x)$ and a kernel $K(x)$ is defined as:

$$(F * K)(x) = \int_{\mathbb{R}^d} I(u)K(x - u)\, du, \tag{3.10}$$

where $F * K$ denotes the convolution, d is the dimensionality of the input space, and $K(x)$ represents the learnable filter or kernel. For discrete data, this becomes:

$$(F * K)[i] = \sum_j I[j]K[i - j], \tag{3.11}$$

where the summation is over the neighborhood defined by the kernel size.

The network is composed of multiple layers, including convolutional layers, activation functions, pooling layers, and fully connected layers. Specifically, the *convolutional layers* apply filters to extract spatial features. The *activation functions* introduce non-linearity, commonly using ReLU:

$$\sigma(x) = \max(0, x). \tag{3.12}$$

The *pooling layers* reduce the spatial dimension for computational efficiency, typically using max pooling:

$$P(x) = \max_{i \in \mathcal{N}(x)} F[i], \tag{3.13}$$

where $\mathcal{N}(x)$ is the neighborhood of x. The *fully connected layers* combine features for final predictions.

The CNN is trained to minimize a loss function, typically defined as the mean squared error (MSE) between the predicted and target values:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \left(v_{\text{predicted}}(x_i) - v_{\text{target}}(x_i)\right)^2, \tag{3.14}$$

where N is the number of training samples, and $v_{\text{predicted}}(x_i)$ and $v_{\text{target}}(x_i)$ are the predicted and target values at x_i, respectively.

By leveraging the ability of CNNs to learn complex transformations, our method effectively computes density-equalizing maps with improved scalability and generalizability.

4. Proposed method

The proposed learning-based density-equalizing mapping method (LDEM) follows a hierarchical pipeline consisting of five main stages, as illustrated in Figure 1:

(1) **Data initialization**: The process begins with an input map represented on an $n \times n$ meshgrid. Population values are assigned to each triangular element formed by a triangulation of this grid.

(2) **Dense-to-coarse transformation**: The fine-grained input is downsampled into a coarser representation to reduce computational complexity and capture global structural information.

(3) **Coarse model processing**: A neural network model processes the coarse data to produce a preliminary transformation field.

(4) **Coarse-to-dense transformation**: The intermediate output is upsampled using an interpolation scheme, transferring the coarse prediction back to the original resolution.

(5) **Fine-tuning with a dense model**: A separate dense-level model further refines the interpolated output, resulting in a final transformation map that achieves high spatial accuracy.

In the following sections, we will describe the five stages in detail.

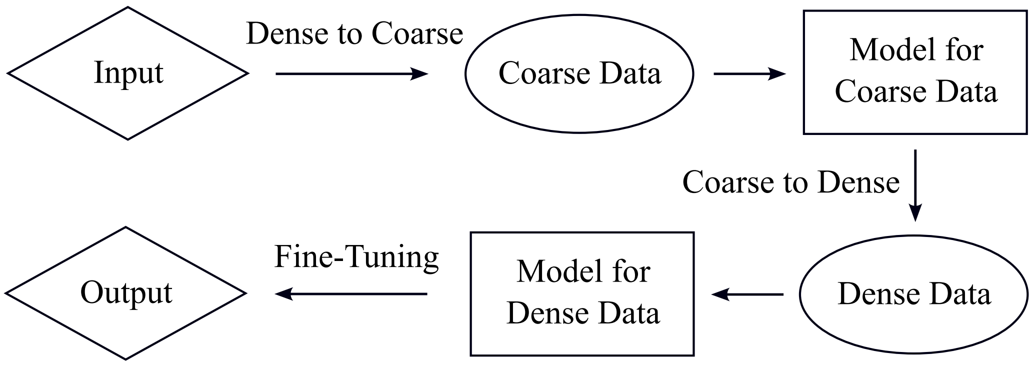


Figure 1. An overview of the proposed learning-based density-equalizing mapping (LDEM) method.

4.1. Data initialization

We initialize the domain using a regular $n \times n$ meshgrid. A triangulation is applied to divide the domain into non-overlapping triangles, each associated with a given population (a real positive scalar). An example of the initial configuration is shown in Figure 2(a).

The objective of the transformation is to relocate the mesh points such that the resulting map equalizes the population density: Each triangle is transformed in a way that its area becomes approximately proportional to the original population weight, effectively producing a near-uniform density across the domain. Figure 2(b) illustrates the result of this transformation. Visually, the adjusted triangles vary in size to accommodate different population levels, while maintaining continuity and avoiding overlaps.

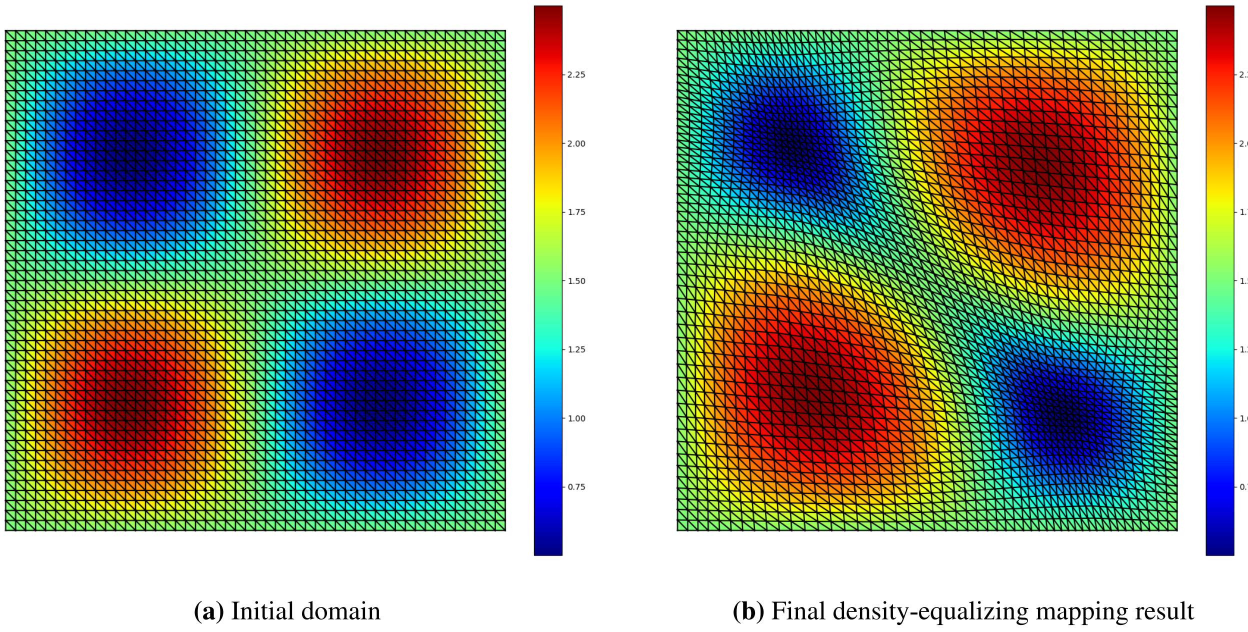


Figure 2. An illustration of density-equalizing maps. Both the initial domain and final mapping result are triangulated and color-coded with the given population. The transformation of the initial triangulated domain adjusts the vertex positions so that the population density, i.e., the population per unit area in the deformed domain, is equalized.

4.2. Dense-to-coarse transformation

To improve computational efficiency, we first approximate the dense input data using a coarser representation. This step enables fast processing while retaining essential global structures of the original data distribution.

We start by generating a coarse grid in the domain. Specifically, a coarse grid of resolution $D_{\text{coarse}} \times D_{\text{coarse}}$ is defined over the unit square $[0, 1]^2$, with uniformly spaced vertices:

$$x_i, y_j \in \left\{0, \frac{1}{D_{\text{coarse}} - 1}, \ldots, 1\right\}, \quad i, j = 1, 2, \ldots, D_{\text{coarse}}. \tag{4.1}$$

The vertex set is denoted as:

$$\mathbf{v}_{\text{coarse}} = \left\{(x_i, y_j) \mid \ i, j = 1, 2, \ldots, D_{\text{coarse}}\right\}. \tag{4.2}$$

Then, each square cell in the coarse grid is divided into two right-angled triangles. Let $k = i \cdot D_{\text{coarse}} + j$ denote the linear index of the grid point (i, j). Then the two triangles covering the cell are:

$$\begin{aligned} \mathbf{f}_{k,1} &= [k, k+1, k+D_{\text{coarse}}], \\ \mathbf{f}_{k,2} &= [k+1, k+D_{\text{coarse}}+1, k+D_{\text{coarse}}]. \end{aligned} \tag{4.3}$$

After performing the triangulation of the coarse grid, for each triangle $\mathbf{f}_i = [\mathbf{v}_a, \mathbf{v}_b, \mathbf{v}_c]$, where $\mathbf{v}_a, \mathbf{v}_b, \mathbf{v}_c$ are the three vertices of the triangle $\mathbf{f}_i$, we calculate the centroid by:

$$\mathbf{c}_i = \frac{\mathbf{v}_a + \mathbf{v}_b + \mathbf{v}_c}{3}. \tag{4.4}$$

Next, we perform the following population aggregation step to define the population on the coarse grid. Given dense triangle centroids $\mathbf{c}_{\text{dense}}$ with corresponding population values $\mathbf{p}_{\text{dense}}$, the population value for each coarse triangle is computed via neighborhood averaging. For a coarse centroid $\mathbf{c}_{\text{coarse},i}$, we define its neighborhood as:

$$\mathcal{N}_i = \left\{ j \mid \left\| \mathbf{c}_{\text{dense},j} - \mathbf{c}_{\text{coarse},i} \right\| < \frac{1}{D_{\text{coarse}}} \right\}. \tag{4.5}$$

The aggregated population is then:

$$\mathbf{p}_{\text{coarse},i} = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \mathbf{p}_{\text{dense},j}. \tag{4.6}$$

This preserves local density structure while significantly reducing resolution.

4.3. Coarse model processing

To learn the transformation fields on the coarse and dense grids, we design an efficient neural network architecture with bottleneck and sparse convolutional layers. The model is lightweight, flexible, and effective for grid-based population data.

4.3.1. Model configuration

Below, we describe the configuration of our model.

As for the input and output dimensions, the model receives a vector of population values as input and outputs a corresponding transformation vector. Input/output dimensions are adjusted according to the grid resolution.

For the dimensionality reduction, a fully connected layer reduces the input dimension I to a low-dimensional bottleneck space B, typically set as $B = 1$:

$$\mathbf{z} = \sigma(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1), \quad \mathbf{z} \in \mathbb{R}^B. \tag{4.7}$$

Here, σ denotes the sigmoid activation.

The model performs sparse processing via 1D convolution. Specifically, a 1D convolution layer with group-wise operations is used to extract nonlinear features:

$$\mathbf{h} = \text{ReLU}(\text{Conv1D}(\mathbf{z})). \tag{4.8}$$

The feature vector is then mapped back to the original output dimension using another fully connected layer:

$$\mathbf{y} = \mathbf{W}_2 \mathbf{h} + \mathbf{b}_2. \tag{4.9}$$

The end-to-end computation of the model is summarized as:

$$\mathbf{y} = \mathbf{W}_2 \cdot \text{ReLU}\,(\text{Conv1D}\,(\sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1))) + \mathbf{b}_2. \tag{4.10}$$

Two independent models are constructed for coarse and dense grid processing. Here we first focus on the problem on the coarse grids. Later, we will extend it for the dense model.

Altogether, this configuration ensures the model is both scalable and adaptable to grid resolution changes, supporting the hierarchical structure of the proposed framework.

4.3.2. Training process

The model training process consists of three key components: loss function design, model initialization, and multi-stage training.

As for the loss function, recall that density-equalizing maps aim to create shape deformations based on the prescribed population. Therefore, one fundamental element in the loss function is the density equalization measure. Also, it is desired that the shape deformations are smooth and natural, without sharp changes or even mesh fold-overs. Hence, our proposed training objective combines population consistency and geometric regularity criteria.

Specifically, consider an $N \times N$ coarse grid in a 2D domain discretized into $2(N-1)^2$ triangles, in which a population value p_i is defined on each triangle t_i. The density equalization aims to produce a new configuration where the area of each transformed triangle t_i' is proportional to the associated population p_i, thereby yielding a uniform density. To assess the uniformity of the density, we define the *density uniformity loss* as follows:

$$\mathcal{L}_{\text{density}} = \frac{\text{std}(\boldsymbol{\rho}_f)}{\text{mean}(\boldsymbol{\rho}_f)}, \tag{4.11}$$

where $\boldsymbol{\rho}_f = (\rho_{f,1}, \rho_{f,2}, \dots, \rho_{f,2(N-1)^2})$ is the vector of face-wise densities defined by

$$\rho_{f,i} = \frac{p_i}{\text{Area}(t_i')}, \tag{4.12}$$

with $i = 1, 2, \dots, 2(N-1)^2$. It is easy to see that $\mathcal{L}_{\text{density}} = 0$ if and only if the density is fully equalized in the entire domain.

Next, to ensure that the shape deformation is smooth and natural, we consider the following two *geometric regularization loss* terms:

$$\mathcal{L}_{\text{slope}} = \frac{1}{N} \sum_{i=1}^{N} \left(\text{slope}_{x,i} + \text{slope}_{y,i}\right) \tag{4.13}$$

and

$$\mathcal{L}_{\text{distance}} = \frac{1}{N} \sum_{i=1}^{N} \left(\text{distance}_{x,i} + \text{distance}_{y,i}\right), \tag{4.14}$$

where the terms are defined as follows:

- Each group i corresponds to a one-dimensional sequence of vertices extracted either *along the x-axis* (a horizontal row) or *along the y-axis* (a vertical column) of the mesh grid. Specifically, an

x-axis group contains all vertices sharing the same y-coordinate index and ordered by increasing x, while a y-axis group contains vertices sharing the same x-coordinate index and ordered by increasing y. In this way, each group represents a line of connected grid points used to measure local geometric variation along one coordinate direction.

- $\text{Slope}_{x,i} = \sum_{j=1}^{N-2} \left| s_{j+1}^{(x)} - s_j^{(x)} \right|$, where $s_j^{(x)} = \dfrac{y_{j+1} - y_j}{x_{j+1} - x_j + \varepsilon}$ is the slope between consecutive points in the i-th group along the x-axis. Here, $\varepsilon = 10^{-8}$ is a small constant to ensure numerical stability in slope calculations.
- $\text{Slope}_{y,i} = \sum_{j=1}^{N-2} \left| s_{j+1}^{(y)} - s_j^{(y)} \right|$, where $s_j^{(y)} = \dfrac{x_{j+1} - x_j}{y_{j+1} - y_j + \varepsilon}$ is the slope between interleaved points in the i-th group along the y-axis, and $\varepsilon = 10^{-8}$.
- $\text{Distance}_{x,i} = \sum_{j=1}^{N-2} \left| d_{j+1}^{(x)} - d_j^{(x)} \right|$, where $d_j^{(x)} = (x_{j+1} - x_j)^2 + (y_{j+1} - y_j)^2$ is the squared distance between neighboring points in x-axis groups.
- $\text{Distance}_{y,i} = \sum_{j=1}^{N-2} \left| d_{j+1}^{(y)} - d_j^{(y)} \right|$, where $d_j^{(y)} = (x_{j+1} - x_j)^2 + (y_{j+1} - y_j)^2$ is the squared distance between neighboring points in y-axis groups (arranged interleaved).

Intuitively, the terms $\text{slope}_{x,i}$ and $\text{slope}_{y,i}$ in $\mathcal{L}_{\text{slope}}$ effectively capture the spatial variation of the slope of the edges in each cell in the grid. Also, the terms $\text{distance}_{x,i}$ and $\text{distance}_{y,i}$ in $\mathcal{L}_{\text{distance}}$ assess the spatial variation in the length of the edges in each cell in the grid. By organizing the vertices into horizontal and vertical groups, the losses separately evaluate the smoothness of deformation along both coordinate directions, ensuring that the mapping preserves geometric coherence and avoids local distortions. By minimizing $\mathcal{L}_{\text{slope}}$ and $\mathcal{L}_{\text{distance}}$, we can ensure a smooth change across the entire domain, thereby effectively enhancing the geometric regularity and naturally avoiding mesh fold-overs.

Combining the above loss functions, we have the overall loss function for training as follows:

$$\mathcal{L} = \lambda_d \cdot \mathcal{L}_{\text{density}} + \lambda_s \cdot \mathcal{L}_{\text{slope}} + \lambda_l \cdot \mathcal{L}_{\text{distance}}, \tag{4.15}$$

where $\lambda_d, \lambda_s, \lambda_l$ are nonnegative parameters.

As for the model initialization, in the initialization phase, we use the Adam optimizer and the mean squared error (MSE) loss to establish a stable baseline.

Finally, as for the model training phases, the model is first trained with MSE loss in the initialization phase:

$$\mathcal{L}_{\text{init}}^{\text{coarse}} = \frac{1}{D_{\text{coarse}}} \sum_{i=1}^{D_{\text{coarse}}} (\hat{\mathbf{y}}_i - \mathbf{y}_i)^2. \tag{4.16}$$

Here, $\mathbf{y}_i \in \mathbb{R}^2$ denotes the target output coordinates corresponding to the identity mapping, and $\hat{\mathbf{y}}_i$ is the model's predicted output for the input point $\mathbf{x}_i$. In this initialization process, the goal is to encourage the network to behave approximately as the identity function over the spatial domain, i.e., $\hat{\mathbf{y}}_i \approx \mathbf{x}_i$. This provides a stable starting point for subsequent training phases that enforce density equalization, helping to prevent large initial distortions in the learned transformation. Then, in the fine-tuning phase, training proceeds using the proposed loss function $\mathcal{L}$ in Eq (4.15). Gradient clipping is applied in both stages to enhance training stability.

Altogether, the above coarse model produces a preliminary coarse transformation field, which is then used in the subsequent stages.

4.4. Coarse-to-dense transformation

To bridge the coarse and dense resolutions, we employ interpolation techniques [30] that transfer the transformation field predicted on the coarse grid to the dense grid. This enables high-resolution refinement while preserving coarse-level consistency.

First, the coarse function values $f_{\text{coarse}}(x_i, y_j)$ are defined on the coarse grid in Eq (4.1). Then, we construct a bilinear interpolation function:

$$f(x, y) = \sum_{i,j} w_{ij}(x, y) \cdot f_{\text{coarse}}(x_i, y_j), \tag{4.17}$$

where $w_{ij}(x, y)$ are bilinear weights based on the proximity of (x, y) to coarse grid nodes. A dense grid is defined as:

$$x_k, y_l \in \left\{0, \frac{1}{D_{\text{dense}} - 1}, \dots, 1\right\}, \quad k, l = 1, 2, \dots, D_{\text{dense}}. \tag{4.18}$$

The interpolated function values on this grid are:

$$f_{\text{dense}}(x_k, y_l) = f(x_k, y_l). \tag{4.19}$$

Finally, the interpolation is applied to both coordinate components independently:

$$\begin{aligned} X_{\text{dense}}(x_k, y_l) &= f_x(x_k, y_l), \\ Y_{\text{dense}}(x_k, y_l) &= f_y(x_k, y_l), \end{aligned} \tag{4.20}$$

where

$$\begin{aligned} f_x(x, y) &= \sum_{i,j} w_{ij}(x, y) \cdot [f_{\text{coarse}}(x_i, y_j)]_1, \\ f_y(x, y) &= \sum_{i,j} w_{ij}(x, y) \cdot [f_{\text{coarse}}(x_i, y_j)]_2, \end{aligned} \tag{4.21}$$

with $[\cdot]_1$ and $[\cdot]_2$ denoting the first and second components, respectively. The final dense transformation grid is then:

$$\mathbf{V}_{\text{dense}} = \{(X_{\text{dense}}(x_k, y_l), Y_{\text{dense}}(x_k, y_l)) \mid k, l = 1, 2, \dots, D_{\text{dense}}\}. \tag{4.22}$$

4.5. Fine-tuning with a dense model

After the interpolation, we use a separate dense-level model to fine-tune the interpolated output and result in the final transformation. The model undergoes a two-stage training process similar to that of the coarse model in Section 4.3.

First, an initialization phase is conducted using the MSE loss to bring the dense model close to the interpolated map. The objective is to preserve spatial coherence and ensure a smooth starting point for subsequent fine-tuning. This phase uses a relatively high learning rate to allow faster convergence:

$$\mathcal{L}_{\text{init}}^{\text{dense}} = \frac{1}{D_{\text{dense}}} \sum_{i=1}^{D_{\text{dense}}} (\hat{\mathbf{Y}}_i - \mathbf{Y}_i)^2, \tag{4.23}$$

where $\mathbf{Y}_i \in \mathbb{R}^2$ denotes the interpolated coordinates from the coarse model, and $\hat{\mathbf{Y}}_i$ is the predicted output from the dense model.

Following this, we perform a fine-tuning phase with a smaller learning rate to refine the transformation. The training objective again uses the same customized loss function defined in Section 4.3.2, which balances density-equalization accuracy and geometric regularity. The fine-tuning stage allows the model to capture finer spatial details and reduce residual distortion in the transformation output.

This two-phase training procedure enables the dense model to achieve high-fidelity density-equalizing transformations while remaining consistent with the initialization provided by the coarse model.

4.6. Hyperparameter settings

To ensure stable and effective learning, we configure hyperparameters specifically for the coarse and dense models. Each model undergoes two training phases: an initialization phase and a main training phase.

For the coarse model, we have the following settings:

- For the initialization phase, we set the learning rate as init_lr_coarse = 1×10^{-2} and the number of epochs as init_epochs_coarse = 800. This stage provides a well-conditioned starting point for later optimization.
- For the training phase, we set the learning rate as train_lr_coarse = 3×10^{-3}, the maximum number of epochs as max_epochs_coarse = 5000, the early stopping patience as 500 epochs, the minimum improvement threshold as min_delta = 1×10^{-4}, and the warm-up period as 150 epochs (during which early stopping is disabled). Early stopping is used to prevent overfitting, and the warm-up period helps stabilize training before convergence is monitored.

For the dense model, we have the following settings:

- For the initialization phase, we set the learning rate as init_lr_dense = 1×10^{-2} and the number of epochs as init_epochs_dense = 800.
- For the training phase, we set the learning rate as train_lr_dense = 2×10^{-4} and the number of epochs as train_epochs_dense = 300. Here, the lower learning rate ensures fine-grained adjustments to high-resolution predictions.

These hyperparameters are empirically selected to balance convergence speed, stability, and final model accuracy for both coarse and dense transformations.

5. Experimental results

5.1. Experimental setup

The proposed method is implemented in Python, leveraging the PyTorch deep learning framework for model definition, training, and optimization. Key packages used in our implementation include `torch`, `torch.nn`, `torch.optim` for neural network components and training routines, `numpy` for numerical array operations and grid generation, and `matplotlib.pyplot` for visualization.

In the following experiments, we generate test cases by creating coarse and dense grids and defining different population distributions on their triangular faces. The grid dimensions are set as $D_{\text{dense}} = 51$

for the dense grid and $D_{\text{coarse}} = 16$ for the coarse grid. For the hyperparameters in the loss function, we choose $\lambda_d = D_{\text{coarse}}$, $\lambda_s = 1$, and $\lambda_l = 10$ for the coarse model, and $\lambda_d = D_{\text{dense}}$, $\lambda_s = 1$, and $\lambda_l = 10$ for the dense model.

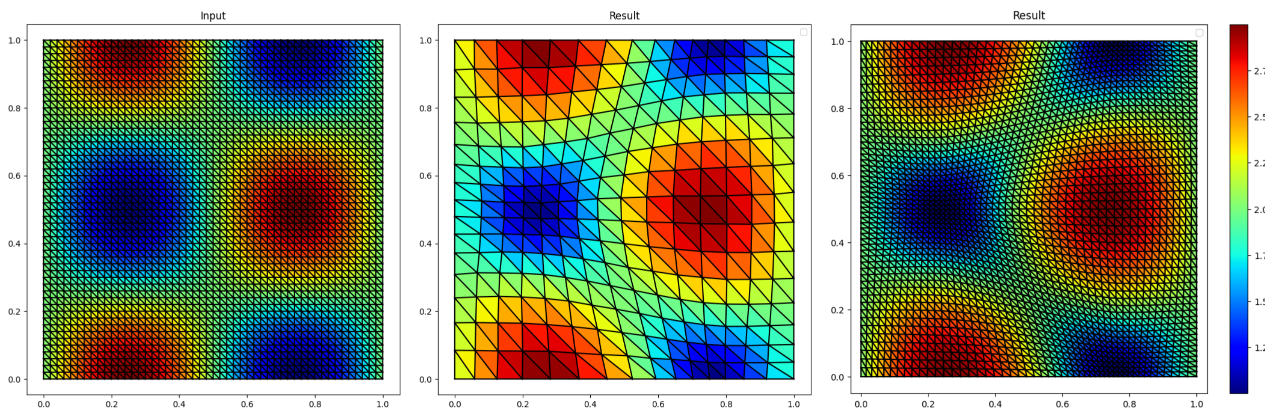
5.2. Test cases and results

Below, we consider six scenarios (Figures 3–8) with a diverse set of population distributions to evaluate the model's ability to handle both simple and complex variations. The *Input* shows the input population distribution on the dense grid, the *Coarse Model Output* shows the intermediate output of the coarse model on the coarse grid, and the *Dense Model Output* shows the final output of the dense model after refinement. From the color maps of the population distribution, we can have a qualitative assessment of the performance of the density-equalizing maps.

We start by considering an example of *Basic Sinusoidal Variation* as shown in Figure 3, with a smooth variation generated using sinusoidal functions:

$$\rho(\mathbf{c}) = 2 + \sin(2\pi c_x) \cdot \cos(2\pi c_y), \tag{5.1}$$

where c_x and c_y are the x- and y-coordinates of the centroid $\mathbf{c}$. From the coarse and dense model outputs, we can easily see that regions with a lower population shrink and regions with a higher population expand. This suggests that the method can effectively produce area changes based on the prescribed population distributions.



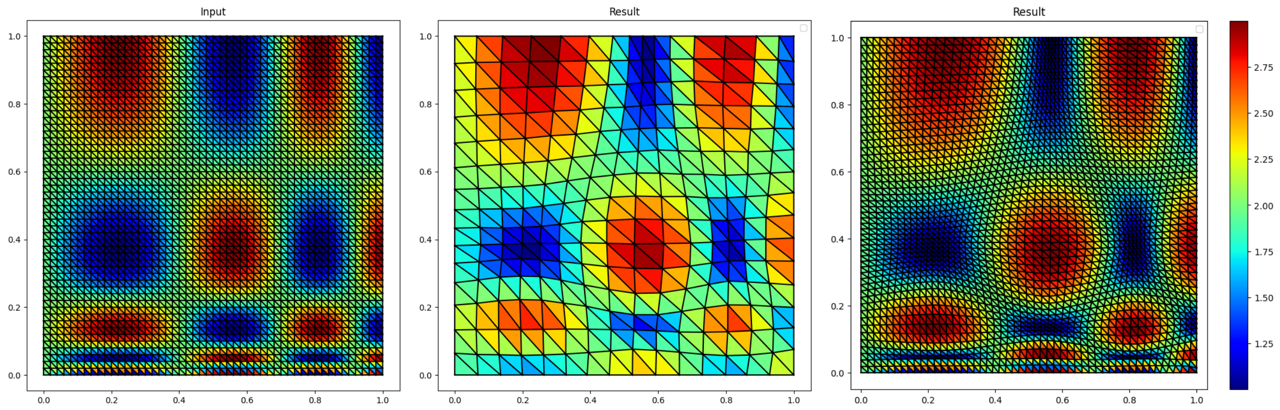
(a) Input **(b)** Coarse Model Output **(c)** Dense Model Output

Figure 3. The experimental result obtained by our proposed LDEM method for the *Basic Sinusoidal Variation* test case.

Then, we consider an example of *Complex Sinusoidal Variation* in Figure 4. Specifically, to introduce additional complexity, exponential and logarithmic transformations are applied to the coordinates:

$$\rho(\mathbf{c}) = 2 + \sin\left(\exp(c_x) \cdot 2\pi\right) \cdot \cos\left(\log(c_y) \cdot \pi\right). \tag{5.2}$$

In the mapping result, one can see that the method can successfully generate shape deformations with the desired area changes even for the nonlinear population variations with sharp transitions and unique patterns across the grid.



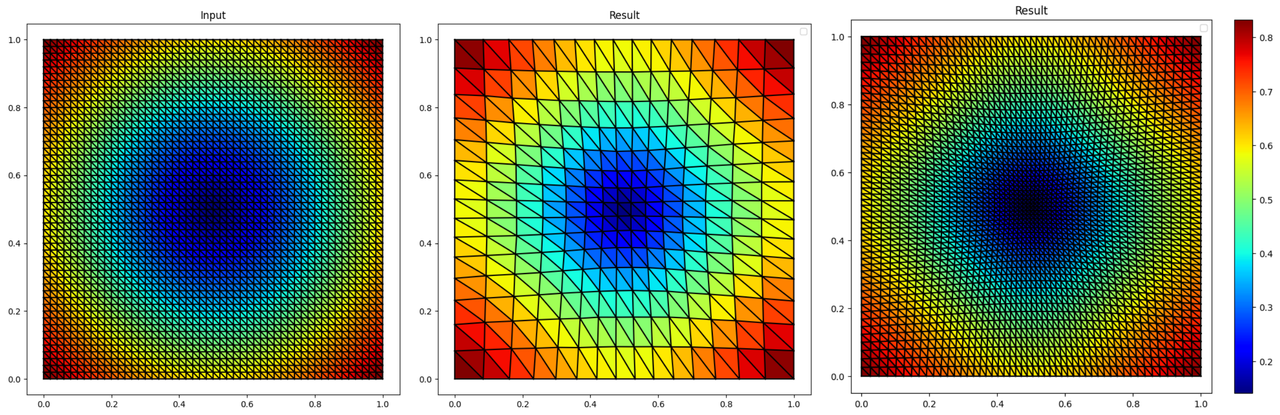
(**a**) Input (**b**) Coarse Model Output (**c**) Dense Model Output

Figure 4. The experimental result obtained by our proposed LDEM method for the *Complex Sinusoidal Variation* test case.

Another example we consider is the *Ring-Shaped Population Distribution* in Figure 5. Here, a ring-like population distribution is introduced, centered at (0.5, 0.5) with a specified radius and thickness. The population decreases smoothly as the distance from the center of the ring deviates from the target radius. The distribution is defined as:

$$\rho(\mathbf{c}) = \exp\left(-\frac{(d(\mathbf{c}) - R)^2}{2 \cdot T^2}\right), \tag{5.3}$$

where $d(\mathbf{c}) = \sqrt{(c_x - 0.5)^2 + (c_y - 0.5)^2}$ is the distance from the centroid $\mathbf{c}$ to the center of the ring, R is the radius of the ring and T is the thickness of the ring. This distribution creates a smooth circular band of population. From the density-equalizing mapping result, we can see that the central part is significantly shrunk while the four corner regions are enlarged.



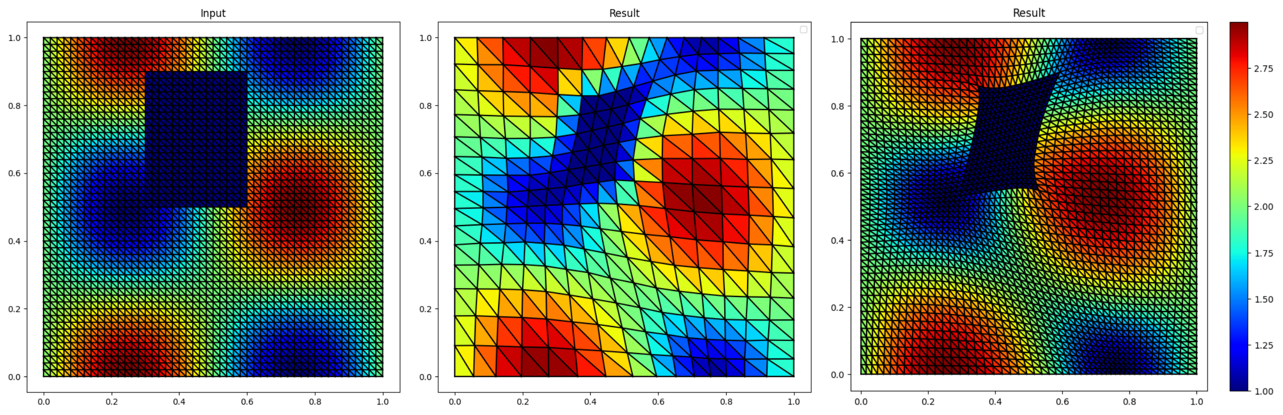
(**a**) Input (**b**) Coarse Model Output (**c**) Dense Model Output

Figure 5. The experimental result obtained by our proposed LDEM method for the *Ring-Shaped Population Distribution* test case.

Next, to simulate more complex scenarios, we consider the *Localized Population Peaks* example in Figure 6. Here, specific rectangular regions within the grid are assigned higher population values. For a rectangle defined by $[x_{\min}, x_{\max}]$ and $[y_{\min}, y_{\max}]$, the population is updated as:

$$\rho(\mathbf{c}) = \begin{cases} P_{\text{peak}}, & \text{if } c_x \in [x_{\min}, x_{\max}] \text{ and } c_y \in [y_{\min}, y_{\max}], \\ \rho(\mathbf{c}), & \text{otherwise.} \end{cases} \tag{5.4}$$

It can be observed that even for such an input population with sharp localized peaks, the proposed method can effectively create a density-equalizing map with different regions enlarged or shrunk, respectively.



(a) Input **(b)** Coarse Model Output **(c)** Dense Model Output

Figure 6. The experimental result obtained by our proposed LDEM method for the *Localized Population Peaks* test case.

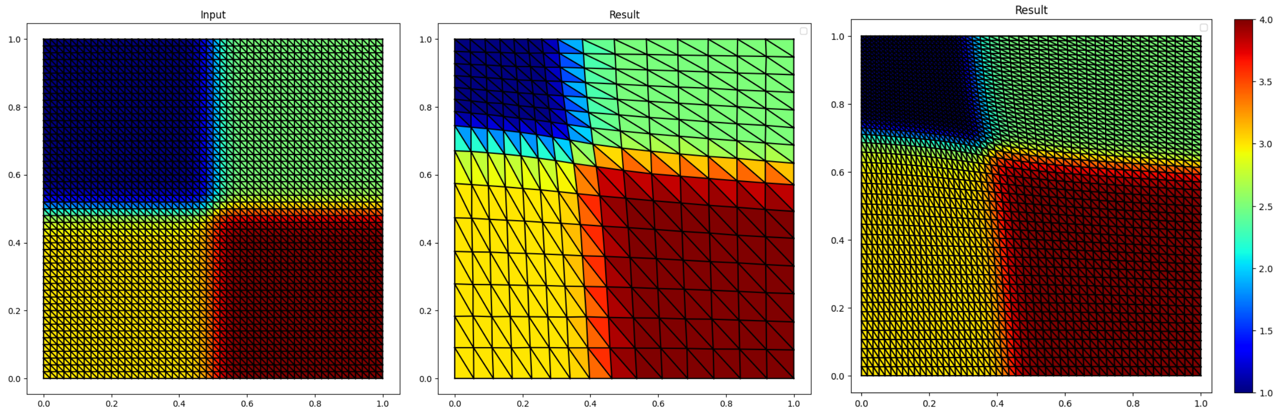
Another complex scenario we consider is the *Smooth Blended Quadrants* as shown in Figure 7. To create smoother variations, a blend function is used to transition between different population levels across the grid quadrants. The smooth blending function is defined as:

$$S(x; c, w) = \frac{1}{1 + \exp\left(-\frac{x-c}{w}\right)}, \tag{5.5}$$

where c is the center, and w controls the blending width. The population is then defined for each quadrant:

$$\begin{aligned} \rho(\mathbf{c}) = {} & 1 \cdot (1 - S(c_x; 0.5, 0.02)) \cdot S(c_y; 0.5, 0.02) \\ & + 2.5 \cdot S(c_x; 0.5, 0.02) \cdot S(c_y; 0.5, 0.02) \\ & + 3 \cdot (1 - S(c_x; 0.5, 0.02)) \cdot (1 - S(c_y; 0.5, 0.02)) \\ & + 4 \cdot S(c_x; 0.5, 0.02) \cdot (1 - S(c_y; 0.5, 0.02)). \end{aligned} \tag{5.6}$$

This gives an input population distribution consisting of four extreme regions with a smooth transition in between. From the mapping result, we can see that the regions are deformed smoothly, with the top-left region significantly shrunk and the bottom-right region enlarged.



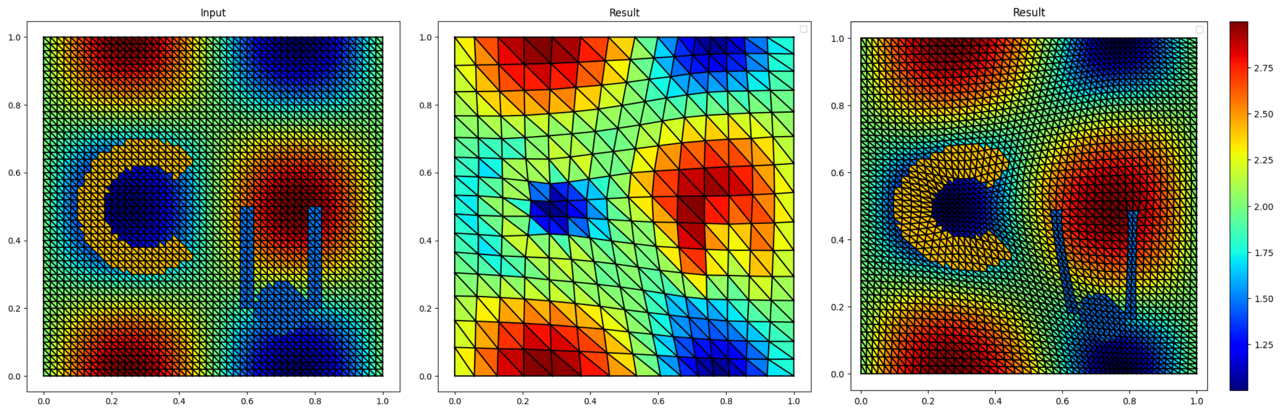
(**a**) Input (**b**) Coarse Model Output (**c**) Dense Model Output

Figure 7. The experimental result obtained by our proposed LDEM method for the *Smooth Blended Quadrants* test case.

Finally, we consider the *Complex Patterns* example in Figure 8. Here, the population distribution is further extended to include predefined patterns, such as the characters "CU". A binary mask is used to define the locations of the pattern, and the population values within these regions are set to higher values to simulate localized emphasis:

$$\rho(\mathbf{c}) = \rho_{\text{base}} + \Delta_{\text{pattern}}, \quad \text{for centroids within the pattern.} \tag{5.7}$$

Even for such a complex input population distribution, it can be observed in the final output that the shape deformation satisfies the desired effect very well.



(**a**) Input (**b**) Coarse Model Output (**c**) Dense Model Output

Figure 8. The experimental result obtained by our proposed LDEM method for the *CU Pattern* test case.

The above examples demonstrate the effectiveness of our proposed method for generating density-equalizing maps with a large variety of desired effects. Next, for a more quantitative assessment of the quality of the mappings, we compute the density distribution using the given population and the initial or final mapping result:

$$\rho = \frac{\text{Population on face}}{\text{Area of face}}. \tag{5.8}$$

In Figures 9–14, we plot the histograms of the initial density ($\frac{\text{Population}}{\text{Initial area}}$) and final density ($\frac{\text{Population}}{\text{Final area}}$) for all mapping examples. For a better visualization, we further divide ρ by mean(ρ) in the histograms. It can be observed that for all examples, the final density distribution is highly concentrated and significantly sharper than the initial density distribution. This indicates that the mapping results produced by our proposed method are highly density-equalizing.

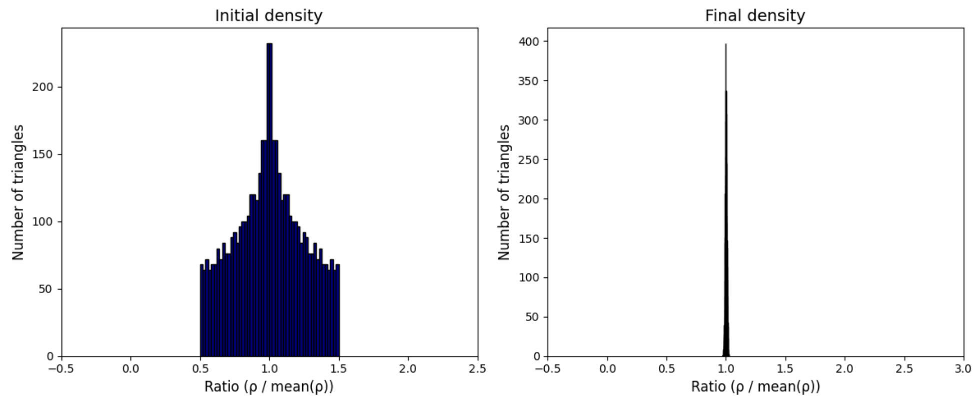


Figure 9. The histograms of the density distribution for the *Basic Sinusoidal Variation* test case.

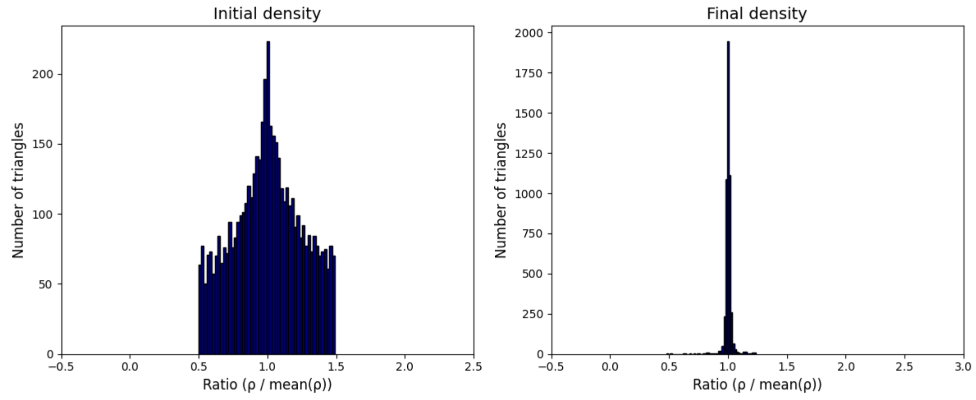


Figure 10. The histograms of the density distribution for the *Complex Sinusoidal Variation* test case.

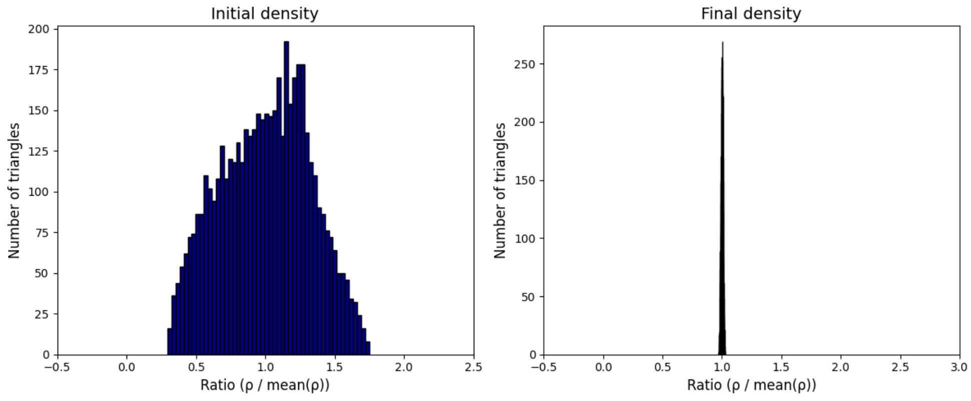


Figure 11. The histograms of the density distribution for the *Ring-Shaped Population Distribution* test case.

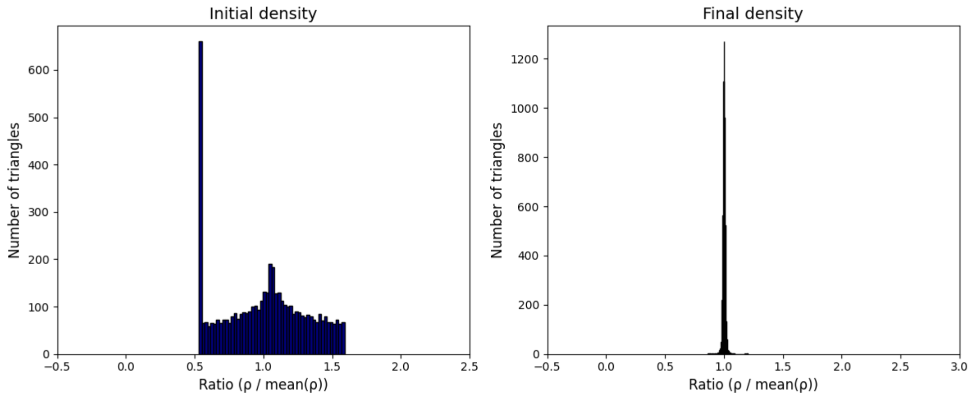


Figure 12. The histograms of the density distribution for the *Localized Population Peaks* test case.

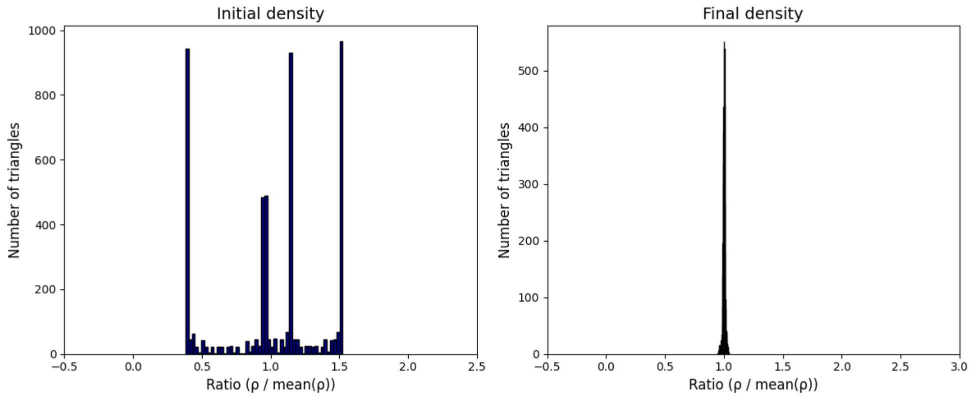


Figure 13. The histograms of the density distribution for the *Smooth Blended Quadrants* test case.

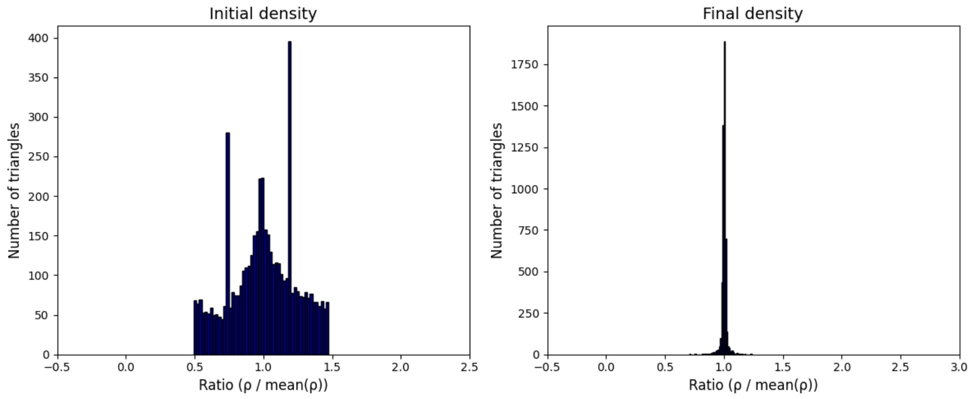


Figure 14. The histograms of the density distribution for the *CU Pattern* test case.

5.3. Comparison with the traditional diffusion-based iterative method

In Table 1, we further compare our proposed DNN-based method with the traditional diffusion-based iterative method [14], which requires iteratively solving the diffusion equation and updating the vertex positions. First, we define the *density-equalizing error* (DE error) as the ratio std(ρ)/mean(ρ) and use it to quantify how uniformly the population is distributed in the mapping result. It can be observed that the proposed method achieves comparable DE error values with the traditional iterative method in all examples. Moreover, for the examples with sharp or complex population transitions (Figures 4, 6, and 8), our method can significantly reduce the DE error by over 30% when compared to the traditional method. We also assess the conformal distortion and the bijectivity of the mappings using the Beltrami coefficient. Specifically, the average norm of the Beltrami coefficient of the mapping, denoted as the BC-mean, gives a measure of the conformal distortion. It can be observed that the BC-mean values achieved by our method are all close to 0 and comparable to those obtained from the traditional method. This suggests that the local geometric distortion of the mappings is small. Also, the maximum value of the norm of the Beltrami coefficient (BC-max) is less than 1 for all examples, which indicates that the mappings are locally homeomorphic, i.e., the mappings do not contain any local mesh fold-overs.

Furthermore, to better understand the generalization capability of our model, we also evaluate the performance of the proposed LDEM method *without fine-tuning*. As shown in Table 1, even without any case-specific optimization, the model still produces stable, bijective mappings with low DE error and minimal geometric distortion. This demonstrates that the learned network captures a transferable density-equalizing prior that can generalize well to unseen population distributions, while the optional fine-tuning step further refines local accuracy.

We also performed a quantitative comparison of the computational efficiency between the proposed methods and the traditional diffusion-based approach, as summarized in Table 2. The three grid resolutions ($D_{\text{dense}} = 51, 101, 151$) correspond to progressively denser discretizations. All experiments were implemented in Python 3.8.10 and executed on an Intel(R) Xeon(R) Gold 6238R CPU @ 2.20 GHz. All methods were run on the CPU without GPU acceleration. For each method, the recorded runtime measures the duration from initialization to near convergence, defined as the point where the relative change in density uniformity std(ρ)/mean(ρ) falls below 10^{-4}, where ρ denotes the density field. The diffusion-based iterative solver adopts the default step size specified in Eq (4.39) of [14], while both variants of our LDEM model use the default hyperparameter configuration described in

Section 4.6. From the results, we see that our LDEM model without fine-tuning has a comparable computational cost to the diffusion-based method at low grid resolutions. At higher grid resolutions, our model becomes significantly more efficient than the diffusion-based method. Moreover, even with the fine-tuning included, our model is still much faster than the diffusion-based DEM method at high grid resolutions. This demonstrates the efficiency of our proposed method.

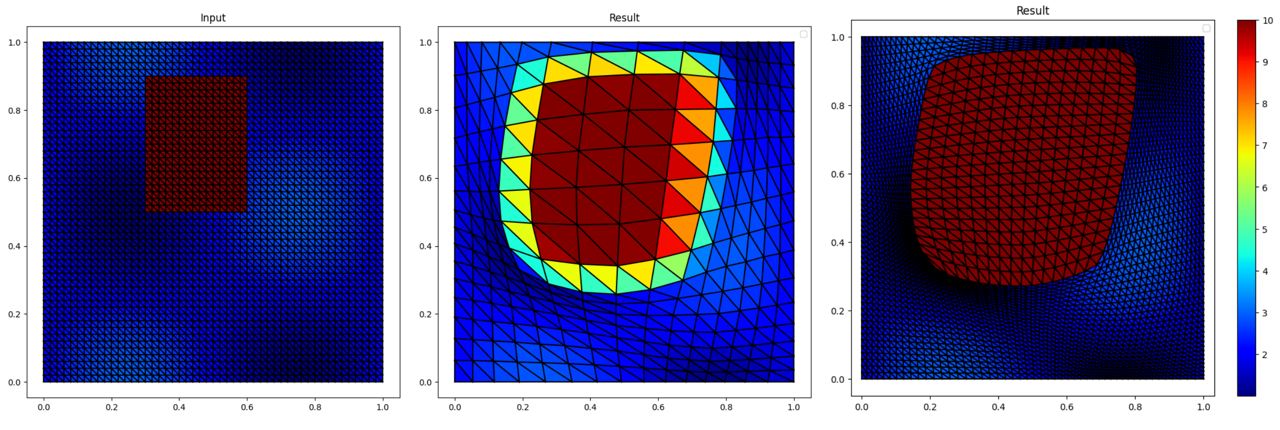
Table 1. Comparison among the proposed LDEM method without fine-tuning, the fine-tuned LDEM, and the diffusion-based iterative method [14] for six test cases, including *Basic Sinusoidal Variation* (Figure 3(a)), *Complex Sinusoidal Variation* (Figure 4(a)), *Ring-Shaped Population Distribution* (Figure 5(a)), *Localized Population Peaks* (Figure 6(a)), *Smooth Blended Quadrants* (Figure 7(a)), and *CU Pattern* (Figure 8(a)). Here, BC-mean represents the average norm of the Beltrami coefficient $|\mu|$, BC-max represents the maximum value of the norm of the Beltrami coefficient $|\mu|$, and the DE error represents the density-equalizing error, defined as the ratio std(ρ)/mean(ρ).

Test Case	**Metric**	**LDEM (no FT)**	**LDEM (FT)**	**Diffusion**
Figure 3(a)	BC-mean	0.1074	0.1144	0.1109
	BC-max	0.2454	0.2732	0.2600
	DE error	0.0341	0.0069	0.0057
Figure 4(a)	BC-mean	0.0936	0.1136	0.1125
	BC-max	0.3424	0.3836	0.3071
	DE error	0.0980	0.0436	0.0733
Figure 5(a)	BC-mean	0.1431	0.1479	0.1451
	BC-max	0.2716	0.2786	0.3146
	DE error	0.0385	0.0084	0.0071
Figure 6(a)	BC-mean	0.1319	0.1390	0.1413
	BC-max	0.3087	0.3964	0.4389
	DE error	0.0871	0.0127	0.0246
Figure 7(a)	BC-mean	0.1829	0.1847	0.1819
	BC-max	0.3742	0.3844	0.3564
	DE error	0.0495	0.0102	0.0088
Figure 8(a)	BC-mean	0.0956	0.1059	0.1042
	BC-max	0.2450	0.3512	0.3203
	DE error	0.0978	0.0233	0.0340

Table 2. Comparison of the computation time (in seconds) among the LDEM without fine-tuning, LDEM with fine-tuning, and diffusion-based iterative method for different test cases with different grid resolutions ($D_{\text{dense}} = 51, 101, 151$).

Test Case	**LDEM (no fine-tuning)**			**LDEM (fine-tuned)**			**Diffusion**		
	D_{dense}			D_{dense}			D_{dense}		
	51	101	151	51	101	151	51	101	151
Figure 3(a)	1.1910	1.4728	1.6723	4.5719	7.8785	13.6589	2.3995	9.7139	>30
Figure 4(a)	1.3664	1.1564	1.3583	4.7429	7.3036	13.5228	1.2269	7.0303	>30
Figure 5(a)	1.2757	1.9302	1.5887	4.5977	8.4441	13.5776	1.1874	7.1366	>30
Figure 6(a)	1.2034	1.1552	1.2314	4.5275	7.6101	13.4317	1.6513	13.7870	>30
Figure 7(a)	5.4161	5.7188	5.5220	8.7590	11.9857	17.3941	1.6966	>30	>30
Figure 8(a)	0.9622	1.3250	1.5933	4.3219	7.4733	13.3977	1.9210	13.1257	>30

Next, we further consider a test case with *Extreme* populations as shown in Figure 15(a). Specifically, we consider a prescribed population distribution with an extremely large value of 10 inside a specific rectangular region, and much smaller values outside the rectangular region. From Figures 15(b,c), it can be observed that our proposed method can handle such an extreme case very well, resulting in a large shape deformation without mesh overlaps.



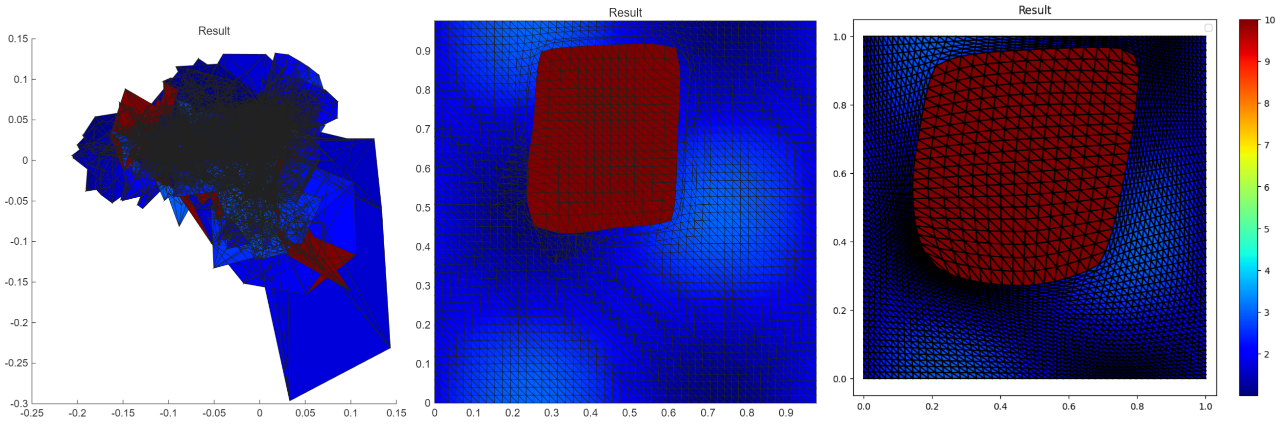
(a) Input **(b)** Coarse Model Output **(c)** Dense Model Output

Figure 15. The experimental result obtained by our proposed method for the *Extreme* test case.

On the contrary, for the traditional diffusion-based iterative method, using the default step size (defined by Eq (4.39) in [14]) will lead to severe mesh overlaps (Figure 16(a)). A possible explanation is that the default step size δt in [14] is dependent on the distribution of the initial density ρ, and for this extreme test case, δt becomes very large and hence the algorithm produces severe overlaps. We also consider reducing the step size to a much smaller value ($\delta t = 5 \times 10^{-5}$) in the diffusion-based method to alleviate the overlapping issue, but that leads to an inaccurate mapping result in which the high-density region is not sufficiently enlarged in the final mapping result (Figure 16(b)). In Table 3, we further compare the mapping results obtained by different approaches in terms of the density-equalizing error

(DE error), local geometric distortion (BC-mean), and bijectivity measure (BC-max). It is easy to see that the diffusion-based method with the default step size gives a highly inaccurate and non-bijective result. While reducing the step size can alleviate the overlapping issue, the result is still non-bijective, and the density-equalizing error remains large. By contrast, our proposed method can significantly reduce the density-equalizing error by over 98% and preserve the bijectivity.

Altogether, the experimental results show that our proposed LDEM method not only achieves accuracy comparable to or better than the traditional diffusion-based approaches but also exhibits remarkable efficiency and robustness. Even without fine-tuning, the model generalizes well across various population distributions, while fine-tuning further enhances precision in highly nonuniform cases. This combination of adaptability, speed, and stability underscores the advantage of the deep learning framework for density-equalizing mappings.



(a) Diffusion (Default step size) **(b)** Diffusion (Reduced step size) **(c)** Our Method

Figure 16. The mapping results obtained by the traditional diffusion-based iterative method [14] with different step sizes and our proposed method for the *Extreme* test case.

Table 3. Quantitative comparison between the traditional diffusion-based iterative method [14] and our proposed method for the *Extreme* test case.

Method	**BC-mean**	**BC-max**	**DE error**
Diffusion with default step size (Figure 16(a))	1.4108	25.8513	46.6316
Diffusion with reduced step size (Figure 16(b))	0.1127	1.8490	2.6494
Our proposed LDEM method (Figure 16(c))	0.3028	0.7778	0.0371

6. Application to surface remeshing

In engineering and graphics, it is common to remesh surfaces to control the triangulation density and quality. This is important for many applications such as shape modeling, solving PDEs on surfaces, and visualization. Using the proposed LDEM method, we can easily perform surface remeshing with different desired effects.

More specifically, given a simply-connected open triangulated surface $\mathcal{M}$, we can first follow the procedure in [14] and parameterize it onto a square domain using the Tutte embedding method.

Denote this initial mapping as $g : \mathcal{M} \to [0,1]^2$. Then, we can easily compute a density-equalizing map $f : [0,1]^2 \to [0,1]^2$ on the unit square using the proposed LDEM method, with the prescribed population controlling the desired effect. Here, for regions that we desire to have a denser triangulation, a higher population can be set. Conversely, for regions that we desire to have a coarser triangulation, a lower population can be set. Then, under the LDEM mapping f, different regions will be enlarged or shrunk accordingly. After getting the mapping result, we can use the inverse mapping $(f \circ g)^{-1}$ to map a uniform triangulation generated on the unit square back to the original surface $\mathcal{M}$. Because of the shape deformation achieved by the LDEM, the new triangulation on $\mathcal{M}$ will generally become non-uniform, with different regions having higher or lower mesh densities following the prescribed effect. This completes the surface remeshing process.

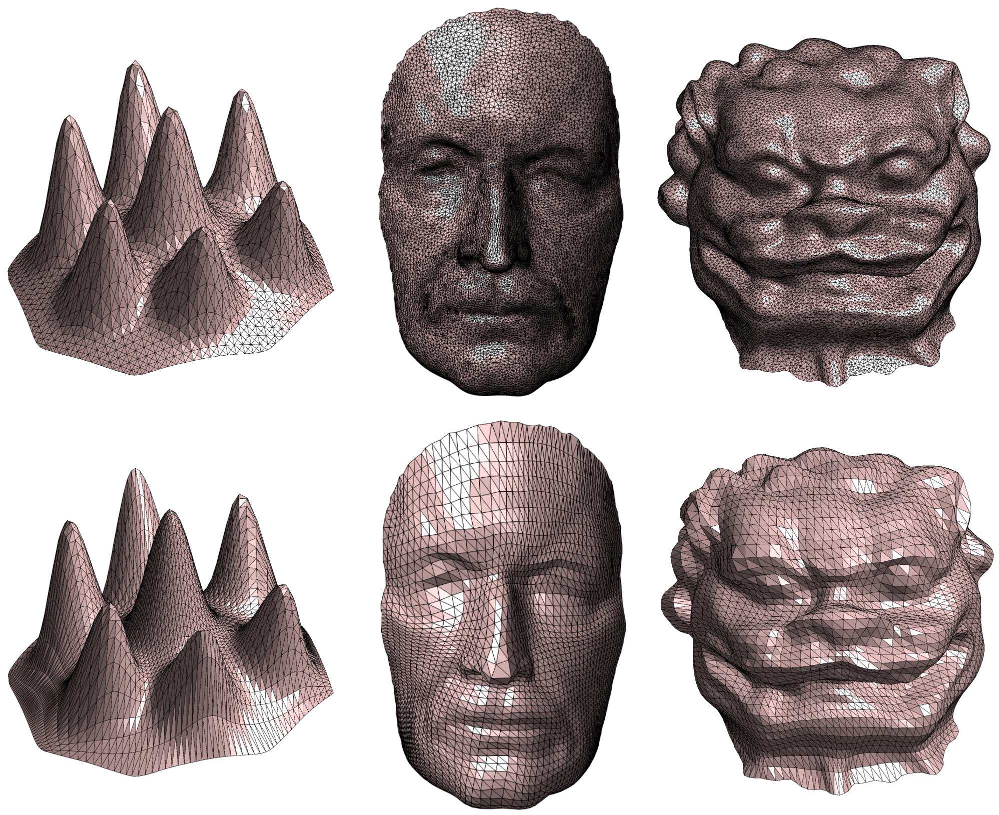


Figure 17. Surface remeshing achieved by our proposed LDEM method. Left to right: *Peaks*, *Max Planck*, and *Chinese Lion*. The top row shows the original surfaces, and the bottom row shows the remeshed surfaces with different effects.

Figure 17 shows three sets of surface remeshing examples with different effects achieved by our method. For the *Peaks* example, which is a surface with multiple peaks, we would like to have a higher mesh resolution at the central peak in the remeshing result. To achieve this, we set the population in the central region to be much larger than that in the other regions and apply the above-mentioned approach. It can be observed that because of the density-equalizing property of our method, the remeshed surface achieves the desired effect very well. In the *Max Planck* example, we set the population in the top left region of the surface to be lower than that in the top right region. Consequently, the mesh density in the top left region is lower in the remeshing result. Alternatively, we can also achieve a relatively uniform remeshing result, as shown in the *Chinese Lion* example, by adjusting the population in different regions. Besides, as mentioned in the previous section, our proposed method preserves the bijectivity very well. Here in the surface remeshing experiments, it can also be observed that the remeshed surfaces are all folding-free.

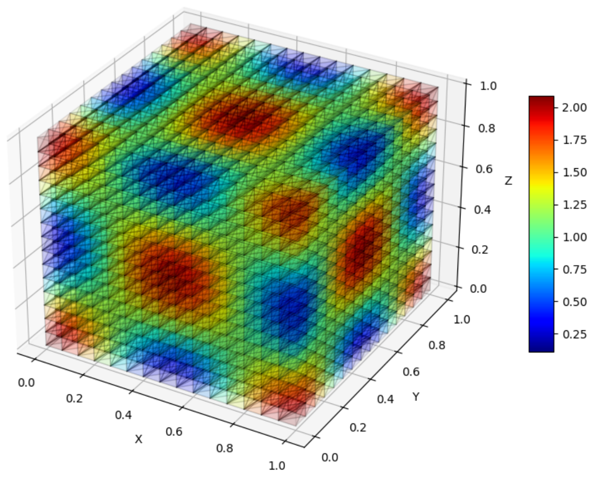
Altogether, the experiments demonstrate the advantage of our proposed method for surface remeshing for engineering applications.

7. Extension to 3D

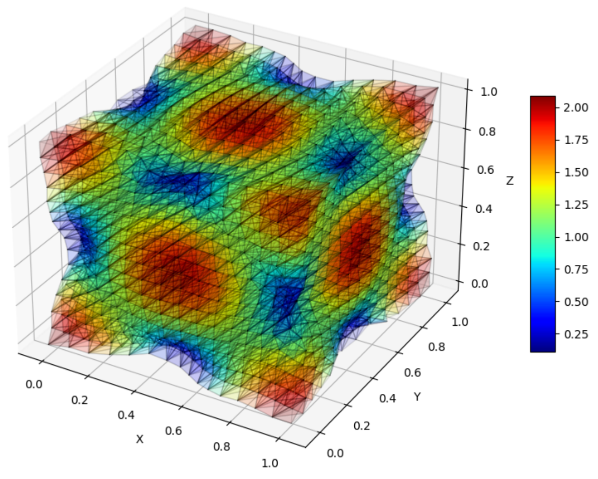
Our proposed LDEM method can be naturally extended to 3D. In particular, note that the extension to 3D (denoted as LDEM-3D) does not require modifying the general model structure. Instead, we will only need to minimally adjust the training data and the loss function for the 3D case.

7.1. Model formulation

Extending the LDEM approach to 3D naturally follows the methodology developed for the 2D case. The primary objective remains to transform an initial 3D domain into one with uniform population density, achieved by adjusting local volumes proportional to assigned population values. To achieve this, we initialize the domain using a regular $N \times N \times N$ meshgrid. A tetrahedralization is applied to divide the domain into non-overlapping tetrahedra, each associated with a given population value. The 3D density-equalizing map aims to relocate the mesh points such that the resulting map equalizes the density, defined as the population per unit volume (see Figure 18 for an illustration).



(a) Initial domain



(b) Final 3D density-equalizing mapping result

Figure 18. An illustration of the 3D density-equalizing maps. Both the initial domain and final mapping result are tetrahedralized and color-coded with the given population. The transformation of the initial tetrahedralized domain adjusts the vertex positions so that the tetrahedron volumes match the target density values.

More specifically, consider a 3D domain discretized into tetrahedra. Each tetrahedron T_i has an initial population value p_i. The density equalization aims to produce a new configuration where the volume of each transformed tetrahedron T_i' satisfies:

$$\mathrm{vol}(T_i') \propto p_i. \tag{7.1}$$

Analogous to the 2D case, here we can consider a coarse model with the aid of a coarse 3D grid. The grid is constructed with $D_{\text{coarse}} \times D_{\text{coarse}} \times D_{\text{coarse}}$ vertices uniformly distributed over the unit cube $[0, 1]^3$. The vertex coordinates are given by:

$$x_i, y_j, z_k \in \left\{0, \frac{1}{D_{\text{coarse}} - 1}, \frac{2}{D_{\text{coarse}} - 1}, \dots, 1\right\}, \quad i, j, k = 1, 2, \dots, D_{\text{coarse}}. \tag{7.2}$$

Tetrahedral elements are defined by connecting adjacent vertices to form multiple tetrahedra within each cubic cell. This ensures that the dense 3D grid is fully tetrahedralized. The population distribution is modeled as a continuous function over the centroids of tetrahedral elements. For a tetrahedron with vertices $\mathbf{v}_a, \mathbf{v}_b, \mathbf{v}_c, \mathbf{v}_d$, the centroid is given by:

$$\mathbf{c} = \frac{\mathbf{v}_a + \mathbf{v}_b + \mathbf{v}_c + \mathbf{v}_d}{4}. \tag{7.3}$$

A population value p_i can then be defined at every centroid $\mathbf{c}_i$.

To quantify the density-equalizing effect, we define the density uniformity loss analogous to the 2D case, but now employing volume measures:

$$\mathcal{L}_{\text{density3D}} = \frac{\mathrm{std}(\boldsymbol{\rho}_T)}{\mathrm{mean}(\boldsymbol{\rho}_T)}, \tag{7.4}$$

where $\boldsymbol{\rho}_T$ is the vector of tetrahedron-wise densities defined by

$$\rho_{T,i} = \frac{p_i}{\mathrm{vol}(T_i')}. \tag{7.5}$$

For geometric regularization, the loss function $\mathcal{L}_{\text{distance}}$ in Eq (4.14) can be extended naturally to distances in $\mathbb{R}^3$:

$$\mathcal{L}_{\text{distance3D}} = \frac{1}{N} \sum_{i=1}^{N} \left(\text{distance}_{x,i} + \text{distance}_{y,i} + \text{distance}_{z,i}\right), \tag{7.6}$$

where $\text{distance}_{x,i}$, $\text{distance}_{y,i}$, $\text{distance}_{z,i}$ are defined based on the squared distance between neighboring points in the x-, y-, and z-axis groups, respectively. Note that one could also generalize the $\mathcal{L}_{\text{slope}}$ in Eq (4.13) to 3D, but the consideration of slope in 3D involves several more terms. For simplicity, we omit this function in the subsequent discussion and experiments.

Altogether, we have the following overall loss function for training in the 3D case:

$$\mathcal{L}_{\text{3D}} = \lambda_d \cdot \mathcal{L}_{\text{density3D}} + \lambda_l \cdot \mathcal{L}_{\text{distance3D}}. \tag{7.7}$$

By following this formulation, the model directly generalizes to three-dimensional scenarios without altering its core architecture, requiring only adjustments to the input data representation and dimensionality of computations.

For the hyperparameters, we have the following settings:

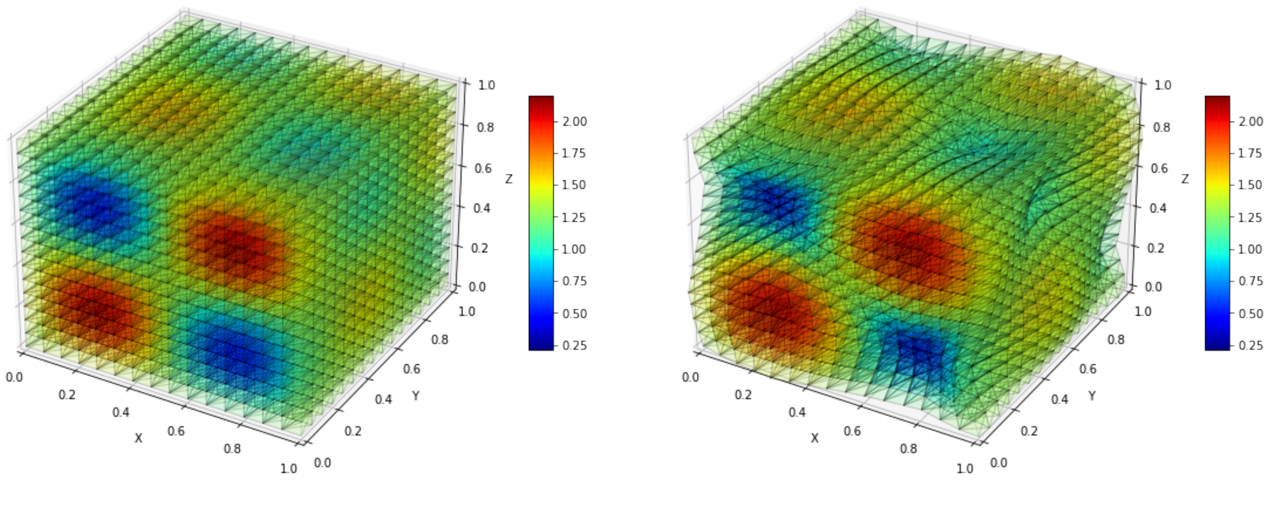
- For the initialization phase, we set the learning rate as init_lr_coarse = 1×10^{-2} and the number of epochs as init_epochs_coarse = 1500. This stage provides a well-conditioned starting point for later optimization.
- For the training phase, we set the learning rate as train_lr_coarse = 1×10^{-4} and the maximum number of epochs as max_epochs_coarse = 5000. The early stopping patience is set as 200 epochs. The minimum improvement threshold is min_delta = 1×10^{-4}, and the warm-up period is 150 epochs (during which early stopping is disabled). Early stopping is used to prevent overfitting, and the warm-up period helps stabilize training before convergence is monitored.

All other parts of the 2D framework, including the coarse-to-dense transformation and fine-tuning with a dense model, can be extended to 3D in a similar manner.

7.2. Experimental results

To simplify the experiment process, we only test the coarse model for the 3D extension as we could deal with the dense case similarly as we did in the 2D cases. Below, we set $D_{\text{coarse}} = 16$ and $\lambda_d = \lambda_l = 1$. Different population distributions are designed to capture various scenarios, ranging from simple to complex spatial variations.

More specifically, we consider four distinct test cases (Figures 19–22) to evaluate the model's capability in handling diverse 3D population distributions. The *Input* shows the initial 3D grid color-coded with the input population distribution, and the *Output* shows the final 3D mapping result obtained by our method.



(**a**) Input (**b**) Output

Figure 19. The experimental result obtained by our proposed LDEM-3D method for the *3D Basic Sinusoidal Variation* test case.

In Figure 19, we first consider an example of *3D Basic Sinusoidal Variation* generated using the sinusoidal function as follows:

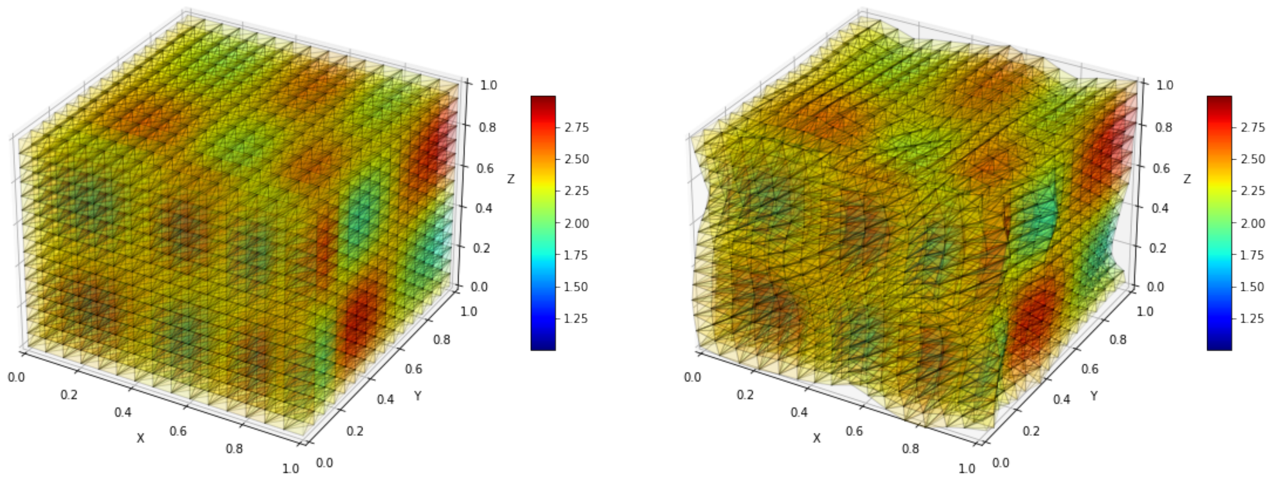
$$\rho(\mathbf{c}) = 1.2 + \sin(2\pi c_x) \cdot \cos(2\pi c_y) \cdot \sin(2\pi c_z), \tag{7.8}$$

where c_x, c_y, c_z denote the x-, y-, and z-coordinates of the centroid $\mathbf{c}$. It can be observed that, analogous to the 2D case, the proposed method can successfully create shape deformation with different regions enlarged or shrunk based on the input ρ.

Next, we consider the *3D Complex Sinusoidal Variation* in Figure 20, in which a more intricate population pattern is defined using exponential and logarithmic transformations in 3D:

$$\rho(\mathbf{c}) = 1.2 + \sin\big(\exp(c_x) \cdot 2\pi\big) \cdot \cos\big(\log(c_y + \epsilon) \cdot \pi\big) \cdot \sin(2\pi c_z), \tag{7.9}$$

where $\epsilon = 10^{-5}$ is a small constant added to avoid singularity at $c_y = 0$. This function produces sharper transitions and more irregular spatial structures. Again, an admissible mapping result with prominent shape deformation can be observed.



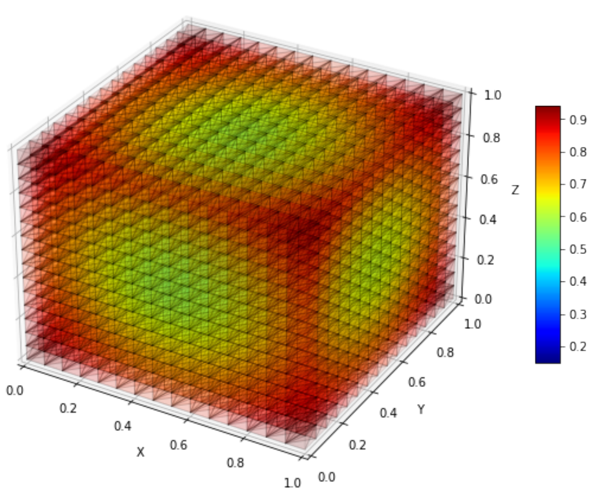
(a) Input **(b)** Output

Figure 20. The experimental result obtained by our proposed LDEM-3D method for the *3D Complex Sinusoidal Variation* test case.

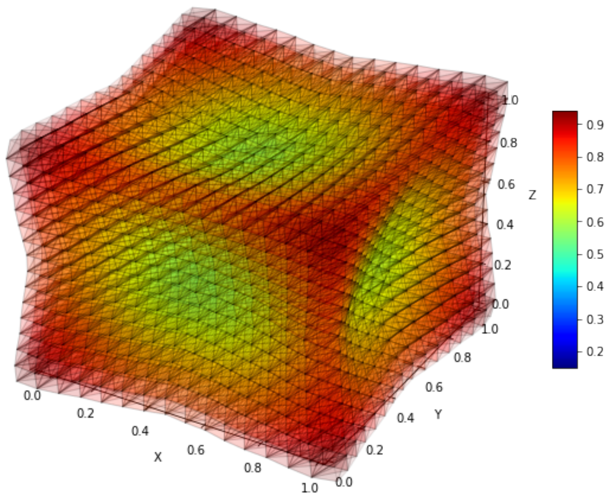
We then consider the *Spherical Shell Population Distribution* example (Figure 21). Here, the distribution is defined by a central axis and a fixed radius R:

$$\rho(\mathbf{c}) = \exp\left(-\frac{(d(\mathbf{c}) - R)^2}{2 \cdot T^2}\right), \tag{7.10}$$

where T controls the thickness of the shell. This simulates a spherical band of density concentrated around a 3D loop. It is easy to see that the central region of the 3D grid shrinks significantly in the mapping result, which matches the desired effect prescribed in ρ very well.



(a) Input



(b) Output

Figure 21. The experimental result obtained by our proposed LDEM-3D method for the *Spherical Shell Population Distribution* test case.

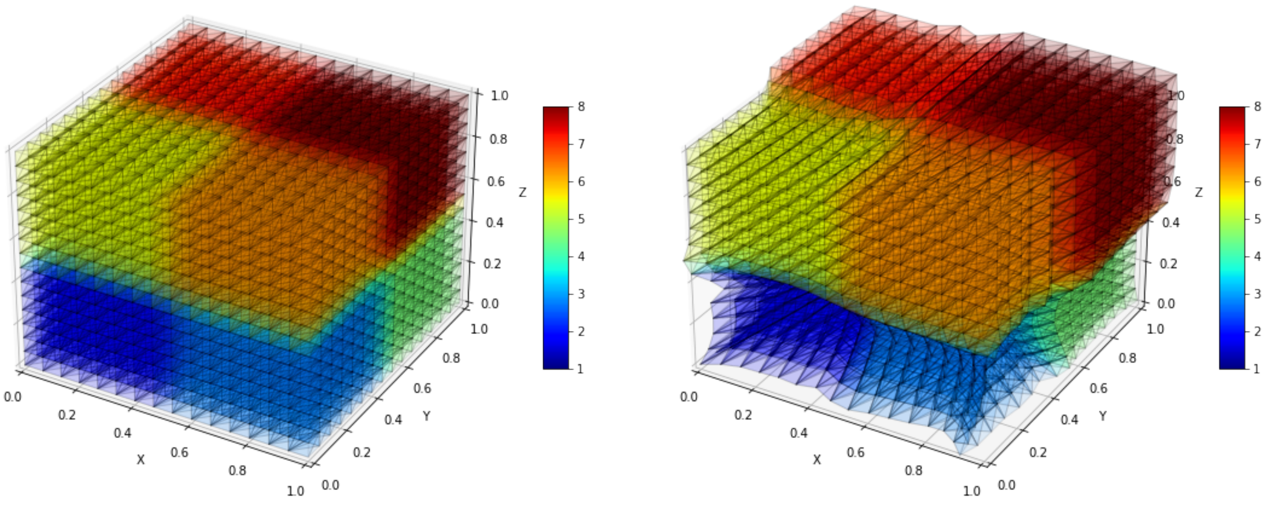
Finally, we consider the *Smooth Blended Octants* test case in Figure 22. Specifically, to model smooth transitions across eight spatial regions (octants), we use a 3D extension of the smooth blending function:

$$S(x; c, w) = \frac{1}{1 + \exp\left(-\frac{x-c}{w}\right)}, \tag{7.11}$$

where $c = 0.5$ is the center of the transition and $w = 0.02$ controls the sharpness of the blend. Then, given a point $\mathbf{c} = (c_x, c_y, c_z)$, the population is defined as:

$$\begin{aligned}\rho(\mathbf{c}) = {} & 1 \cdot (1 - S(c_x; 0.5, 0.02)) \cdot (1 - S(c_y; 0.5, 0.02)) \cdot (1 - S(c_z; 0.5, 0.02)) \\ & + 2 \cdot S(c_x; 0.5, 0.02) \cdot (1 - S(c_y; 0.5, 0.02)) \cdot (1 - S(c_z; 0.5, 0.02)) \\ & + 3 \cdot (1 - S(c_x; 0.5, 0.02)) \cdot S(c_y; 0.5, 0.02) \cdot (1 - S(c_z; 0.5, 0.02)) \\ & + 4 \cdot S(c_x; 0.5, 0.02) \cdot S(c_y; 0.5, 0.02) \cdot (1 - S(c_z; 0.5, 0.02)) \\ & + 5 \cdot (1 - S(c_x; 0.5, 0.02)) \cdot (1 - S(c_y; 0.5, 0.02)) \cdot S(c_z; 0.5, 0.02) \\ & + 6 \cdot S(c_x; 0.5, 0.02) \cdot (1 - S(c_y; 0.5, 0.02)) \cdot S(c_z; 0.5, 0.02) \\ & + 7 \cdot (1 - S(c_x; 0.5, 0.02)) \cdot S(c_y; 0.5, 0.02) \cdot S(c_z; 0.5, 0.02) \\ & + 8 \cdot S(c_x; 0.5, 0.02) \cdot S(c_y; 0.5, 0.02) \cdot S(c_z; 0.5, 0.02).\end{aligned} \tag{7.12}$$

This expression smoothly interpolates population values across all eight octants in the unit cube using only the coordinates (c_x, c_y, c_z). From the mapping result, it is easy to see that the eight regions are enlarged or shrunk proportionally.



(a) Input **(b)** Output

Figure 22. The experimental result obtained by our proposed LDEM-3D method for the *Smooth Blended Octants* test case.

For a more quantitative analysis, we also present the histograms of the initial density distribution ($\frac{\text{Population}}{\text{Initial volume}}$) and the final density distribution ($\frac{\text{Population}}{\text{Final volume}}$) for each test case (see Figures 23–26). In all examples, it can be observed that the final density distribution is much more concentrated at 1 when compared with the initial density distribution. This shows that our LDEM-3D method can effectively achieve 3D density-equalizing maps with the desired shape deformation effects.

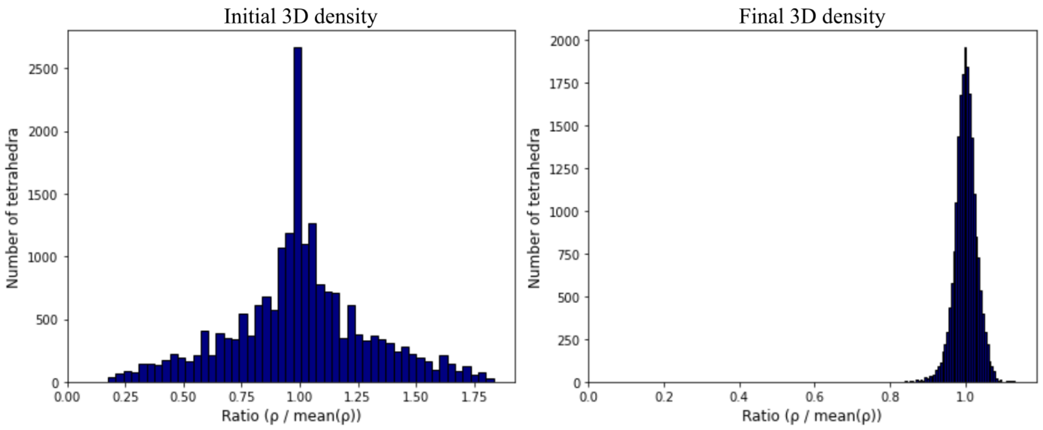


Figure 23. The histogram of the density distribution for the *3D Basic Sinusoidal Variation* test case.

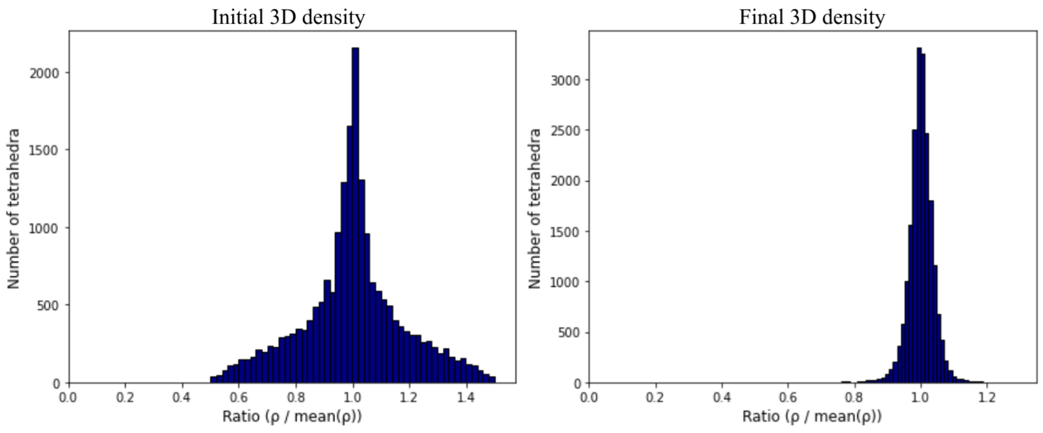


Figure 24. The histogram of the density distribution for the *3D Complex Sinusoidal Variation* test case.

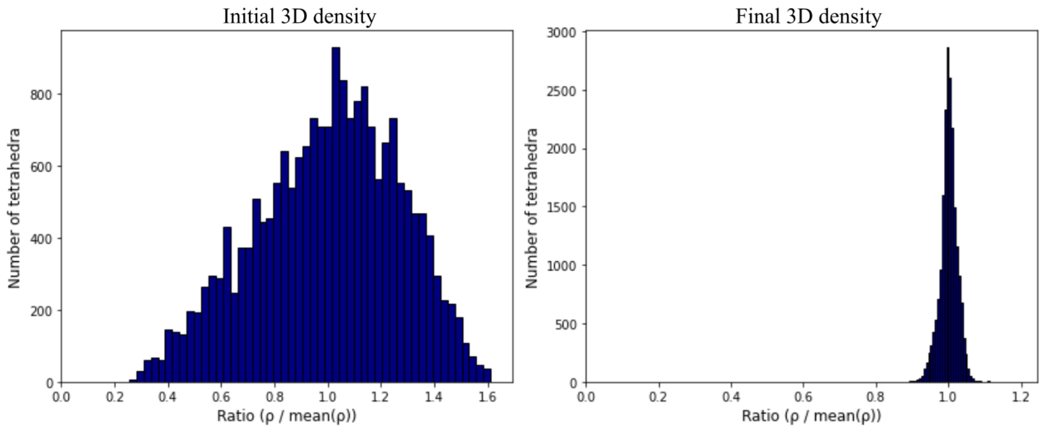


Figure 25. The histogram of the density distribution for the *Spherical Shell Population Distribution* test case.

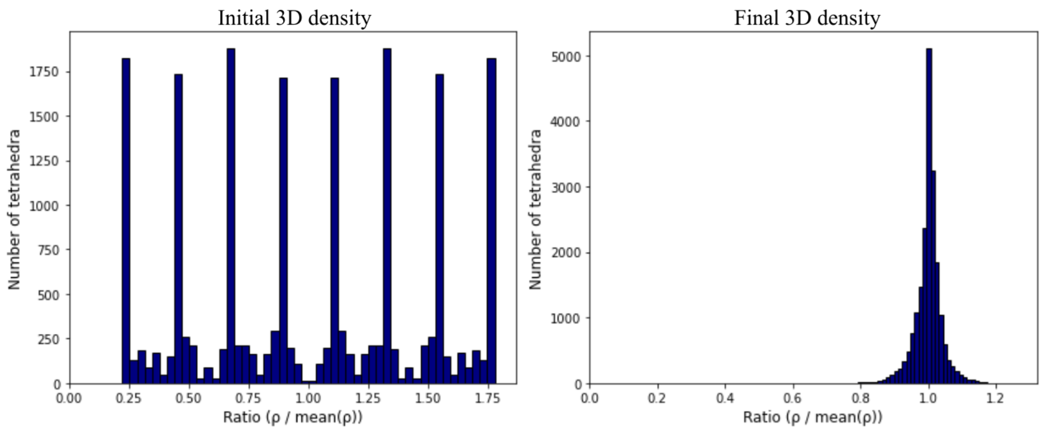


Figure 26. The histogram of the density distribution for the *Smooth Blended Octants* test case.

8. Conclusions

In this paper, we have developed a novel learning-based framework for constructing density-equalizing maps. By utilizing deep neural networks, we allow the model to generalize across arbitrary continuous population distributions, achieving robust performance without retraining for each new case. Our approach combines density equalization objectives with smoothness regularization, ensuring both fidelity and geometric plausibility of the resulting maps. Numerical experiments demonstrate the effectiveness of our method in producing accurate and visually coherent transformations for a wide range of prescribed population distributions. We have also applied our method for surface remeshing with different desired remeshing effects. Altogether, our method allows for scalable and robust computation of density-equalizing maps for practical applications.

As shown in the previous section, our proposed method can be easily extended from 2D to 3D for producing volumetric deformations based on prescribed population distributions. A natural next step is to further extend the proposed framework for more complex surface and volumetric domains to handle a wider class of shape mapping problems. Also, while the experimental results for both the 2D and 3D cases do not exhibit mesh overlaps, we do not have a theoretical guarantee for the positivity of the Jacobian determinants currently. Therefore, we plan to explore more of the theoretical properties of the learned mappings in our future work to better understand their limitations.

Data availability

The code and data are available in the CUHK Research Data Repository at `https://doi.org/10.48668/QJWAB7`.

Author contributions

Yanwen Huang: Data curation, Formal analysis, Investigation, Methodology, Software, Visualization, Writing—original draft, Writing—review and editing; Lok Ming Lui: Conceptualization, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Supervision, Writing—review and editing; Gary P. T. Choi: Conceptualization, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Supervision, Writing—review and editing. All authors have read and approved the final version of the manuscript for publication.

Use of Generative-AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work was supported in part by the CUHK Research Data Management Development Fund under Project Code 4730356 (to G. P. T. Choi) and HKRGC GRF (Project ID 14306721) (to L. M. Lui).

Conflict of interest

Prof. Gary P. T. Choi is the Guest Editor of special issue "Applied Mathematics and Scientific Computing in Engineering and Biology" for AIMS Mathematics. Prof. Gary P. T. Choi was not involved in the editorial review and the decision to publish this article.

All authors declare no conflicts of interest in this paper.

References

1. M. T. Gastner, M. E. J. Newman, Diffusion-based method for producing density-equalizing maps, *Proc. Natl. Acad. Sci.*, **101** (2004), 7499–7504. https://doi.org/10.1073/pnas.0400280101

2. H. Sun, Z. Li, Effectiveness of cartogram for the representation of spatial data, *Cartogr. J.*, **47** (2010), 12–21. https://doi.org/10.1179/000870409X12525737905169

3. S. Nusrat, S. Kobourov, The state of the art in cartograms, *Comput. Graph. Forum*, **35** (2016), 619–642.

4. M. Hogräfer, M. Heitzler, H. J. Schulz, The state of the art in map-like visualization, *Comput. Graph. Forum*, **39** (2020), 647–674.

5. K. S. Gleditsch, M. D. Ward, Diffusion and the international context of democratization, *Int. Organ.*, **60** (2006), 911–933. https://doi.org/10.1017/S0020818306060309

6. R. Arundel, C. Hochstenbach, Divided access and the spatial polarization of housing wealth, *Urban Geogr.*, **41** (2020), 497–523. https://doi.org/10.1080/02723638.2019.1681722

7. V. Colizza, A. Barrat, M. Barthélemy, A. Vespignani, The role of the airline transportation network in the prediction and predictability of global epidemics, *Proc. Natl. Acad. Sci.*, **103** (2006), 2015–2020. https://doi.org/10.1073/pnas.0510525103

8. J. R. M. H. Lenoir, R. R. Bertrand, L. Comte, L. Bourgeaud, T. Hattab, J. Murienne, et al., Species better track climate warming in the oceans than on land, *Nat. Ecol. Evol.*, **4** (2020), 1044–1059. https://doi.org/10.1038/s41559-020-1198-2

9. M. Pratt, O. L. Sarmiento, F. Montes, D. Ogilvie, B. H. Marcus, L. G. Perez, et al., The implications of megatrends in information and communication technology and transportation for changes in global physical activity, *Lancet*, **380** (2012), 282–293. https://doi.org/10.1016/S0140-6736(12)60736-3

10. A. Mislove, S. Lehmann, Y. Y. Ahn, J. P. Onnela, J. N. Rosenquist, Understanding the demographics of Twitter users, *Proceedings of the International AAAI Conference on Web and Social Media*, 2011, 554–557.

11. J. Murphy, J. Zhu, Neo-colonialism in the academy? Anglo-American domination in management journals, *Organization*, **19** (2012), 915–927.

12. R. K. Pan, K. Kaski, S. Fortunato, World citation and collaboration networks: uncovering the role of geography in science, *Sci. Rep.*, **2** (2012), 902. https://doi.org/10.1038/srep00902

13. G. P. T. Choi, B. Chiu, C. H. Rycroft, Area-preserving mapping of 3D carotid ultrasound images using density-equalizing reference map, *IEEE Trans. Biomed. Eng.*, **67** (2020), 1507–1517. https://doi.org/10.1109/TBME.2019.2963783

14. G. P. T. Choi, C. H. Rycroft, Density-equalizing maps for simply connected open surfaces, *SIAM J. Imaging Sci.*, **11** (2018), 1134–1178. https://doi.org/10.1137/17M1124796

15. Z. Lyu, G. P. T. Choi, L. M. Lui, Bijective density-equalizing quasiconformal map for multiply connected open surfaces, *SIAM J. Imaging Sci.*, **17** (2024), 706–755. https://doi.org/10.1137/23M1594376

16. G. P. T. Choi, C. H. Rycroft, Volumetric density-equalizing reference maps with applications, *J. Sci. Comput.*, **86** (2021), 41. https://doi.org/10.1007/s10915-021-01411-4

17. W. R. Tobler, A continuous transformation useful for districting, *Ann. N. Y. Acad. Sci.*, **219** (1973), 215–220. https://doi.org/10.1111/j.1749-6632.1973.tb41401.x

18. J. A. Dougenik, N. R. Chrisman, D. R. Niemeyer, An algorithm to construct continuous area cartograms, *Prof. Geogr.*, **37** (1985), 75–81. https://doi.org/10.1111/j.0033-0124.1985.00075.x

19. D. Dorling, *Area cartograms: their use and creation*, Concepts and Techniques in Modern Geography series no. 59, Environmental Publications, University of East Anglia, 1996.

20. M. T. Gastner, V. Seguy, P. More, Fast flow-based algorithm for creating density-equalizing map projections, *Proc. Natl. Acad. Sci.*, **115** (2018), E2156–E2164. https://doi.org/10.1073/pnas.1712674115

21. A. Arranz-López, J. A. Soria-Lara, A. Ariza-Álvarez, An end-user evaluation to analyze the effectiveness of cartograms for mapping relative non-motorized accessibility, *Environ. Plan B: Urban Anal. City Sci.*, **48** (2021), 2880–2897. https://doi.org/10.1177/2399808321991541

22. Z. Li, S. Aryana, Diffusion-based cartogram on spheres, *Cartogr. Geogr. Inf. Sci.*, **45** (2018), 464–475. https://doi.org/10.1080/15230406.2017.1408033

23. Z. Lyu, L. M. Lui, G. P. T. Choi, Spherical density-equalizing map for genus-0 closed surfaces, *SIAM J. Imaging Sci.*, **17** (2024), 2110–2141. https://doi.org/10.1137/24M1633911

24. Z. Lyu, L. M. Lui, G. P. T. Choi, Ellipsoidal density-equalizing map for genus-0 closed surfaces, *arXiv Preprint*, 2024. https://doi.org/10.48550/arXiv.2410.12331

25. S. Yao, G. P. T. Choi, Toroidal density-equalizing map for genus-one surfaces, *J. Comput. Appl. Math.*, **472** (2026), 116844. https://doi.org/10.1016/j.cam.2025.116844

26. M. Shaqfa, G. P. T. Choi, G. Anciaux, K. Beyer, Disk harmonics for analysing curved and flat self-affine rough surfaces and the topological reconstruction of open surfaces, *J. Comput. Phys.*, **522** (2025), 113578. https://doi.org/10.1016/j.jcp.2024.113578

27. G. P. T. Choi, M. Shaqfa, Hemispheroidal parameterization and harmonic decomposition of simply connected open surfaces, *J. Comput. Appl. Math.*, **461** (2025), 116455. https://doi.org/10.1016/j.cam.2024.116455

28. Z. Li, S. A. Aryana, Visualization of subsurface data using three-dimensional cartograms, *Advances in Remote Sensing and Geo Informatics Applications: Proceedings of the 1st Springer Conference of the Arabian Journal of Geosciences (CAJG-1), Tunisia 2018*, Springer, 2019, 17–19. https://doi.org/10.1007/978-3-030-01440-7_4

29. I. Goodfellow, Y. Bengio, A. Courville, *Deep learning*, MIT Press, 2016. Available from: `http://www.deeplearningbook.org`.

30. A. Weiser, S. E. Zarantonello, A note on piecewise linear and multilinear table interpolation in many dimensions, *Math. Comput.*, **50** (1988), 189–196.