Metamaterials, Morphometrics, Morphogenesis, and Mappings

A DISSERTATION PRESENTED
BY
PUI TUNG CHOI
TO
THE JOHN A. PAULSON SCHOOL OF ENGINEERING AND APPLIED SCIENCES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN THE SUBJECT OF
APPLIED MATHEMATICS

HARVARD UNIVERSITY
CAMBRIDGE, MASSACHUSETTS
MAY 2020

Dissertation Advisors: Prof. L. Mahadevan, Prof. Chris H. Rycroft Pui Tung Choi

Metamaterials, Morphometrics, Morphogenesis, and Mappings

Abstract

In this thesis, we develop mathematical tools to tackle important problems in metamaterials, morphometrics, morphogenesis, and mappings. Specifically, we propose novel methods with a solid theoretical foundation to control the geometry and topology of kirigami and origami metamaterials. We also analyze the growth and form of biological shapes including insect wing and ferret brain using computational geometry. We further propose physically-based methods for producing mappings and deformations of two- and three-dimensional objects and explore their applications to geometry processing, medical imaging and data visualization. Altogether, this thesis brings in new mathematical insights and techniques for advancing our understanding of the physical world, the biological world, and the digital world.

Contents

Acknowledgments

I wish to express my deepest gratitude to my advisors Prof. L. Mahadevan and Prof. Chris Rycroft, for their guidance, support and encouragement over the past four years at Harvard. I am extraordinarily fortunate to have worked with them on many fascinating research projects and have learned a lot from them. I would also like to thank Prof. Katia Bertoldi for helpful suggestions on this thesis. My appreciation is further extended to my collaborators Prof. Ronald Lui and Prof. Bernard Chiu for their support over the past several years.

I take this opportunity to thank the Croucher Foundation and the NSF-Simons Center for Mathematical & Statistical Analysis of Biological Systems for the financial support for my research.

I am grateful to everyone in the Mahadevan group and the Rycroft group. I have had the pleasure of collaborating with many excellent colleagues, including Levi Dudte, Salem Al Mosleh, Aditi Chakrabarti, Siheng Chen, Molly Edwards, and many others.

I am very thankful to my old friends in Hong Kong, especially Anselm Hui, Sherwood Fung, Farley Law, Marwin Law, and Thomas Tang, who have been a constant source of energy and encouragement for over a decade.

I would like to thank my parents Pui-Kuen Choi and Lai-Oi Lam, and my brothers Chun-Sing Choi and Yiu-Hung Choi. This thesis would not have been possible without their strong support.

This thesis is dedicated to my fiancée, Mandy Man, for her love, patience and understanding.

0

Introduction

Mathematics is essential in the physical world, the biological world, and the digital world. In the physical world, the physical properties of architected materials are closely related to their geometry and topology. In the biological world, biological structures are distinguished by differences in their geometry which may either be the cause or the consequence (or both) of their different functional properties. In the digital world, mappings and deformations of two and three dimensional objects

are computed using geometric methods. In this thesis, we develop new mathematical models and tools to advance our understanding of these three worlds. More specifically, we use discrete and continuous geometry to tackle various important problems for interdisciplinary applications in science and engineering. While geometry has been studied since antiquity, it still underpins many questions in modern research areas including materials science, quantitative biology, medical imaging and geometry processing.

The first part of this thesis, consisting of Chapter 1–4, focuses on mathematical metamaterial design. *Kirigami*, the creative art of paper cutting originated in Japan, has recently emerged as prototypical mechanical metamaterials. Kirigami tessellations are deployable structures formed by introducing architected cuts on a flat, thin sheet of material. There has been a vast number of studies on their geometry [58,56,57,110,138], topology [16,83] and mechanics [134,108,109,98] and their use as materials with a range of unusual properties such as topological insulators [72,133] and auxetic structures [137,53]. Moreover, kirigami has been applied for the design of graphene structures [8], nanocomposites [119], shape-morphing and super-stretchable sheets [75,95,117,66,76,100,15], inflatable structures [77], soft robots [111] etc. However, almost without exception, the above works have focused on the forward problem of quantifying the geometry and kinematics of deployment of a prescribed kirigami pattern, where the prescribed pattern is usually motivated by ancient art or nature and designed manually. From a mathematical and technological perspective, a more interesting question is the inverse problem of designing new kirigami patterns that achieve some prescribed shape-morphing properties. More specifically, can one design the geometry of the cuts in a given planar kirigami structure, so that it can be deployed into a prescribed final shape in two or three

dimensions? In Chapter 1, we solve this problem by developing a novel inverse design framework for generalizing the cut geometry of kirigami metamaterials. In particular, we identify geometric constraints in lengths and angles for designing deployable structures with highly nontrivial shape changes upon deployment, such as a square-to-circle transformation.

In recent years, several studies have focused on the design of reconfigurable metamaterials that admit multiple folded states [120,134,46,102,36] inspired by *origami*, the art of paper folding. On the contrary, the study of such reconfigurable structures for kirigami has only been limited to a few periodic patterns [129,50]. In Chapter 2, we extend our inverse design framework by further identifying the geometric constraints that guarantee the reconfigurability and rigid-deployability of generalized kirigami patterns. This allows us to create a wide range of kirigami structures that admit multiple contracted states.

To complement our study on the geometric control of kirigami in the first two chapters, we explore the topological control of kirigami in Chapter 3. Instead of changing the cut geometry, i.e. the edge lengths and angles of the tiles in a kirigami structure to achieve different shapes, we change the cut topology of the kirigami structure by connecting or disconnecting neighboring tiles, thereby controlling the rigidity and connectivity of the structure. In particular, we propose a hierarchical construction approach for creating minimum rigidifying and connecting link patterns of arbitrary size using smaller patterns.

While the above-mentioned methods are capable of controlling the geometry and topology of kirigami, the global design problems involve all nodes in a structure and hence are time-consuming. In Chapter 4, we devise a method for the additive design of origami and kirigami as an alternative

approach for metamaterial design. Instead of solving the design problems over the entire structure, we identify the constraints in lengths and angles at the growth front of any structure, thereby reducing the global problems to much simpler local design problems.

The second part of this thesis, consisting of Chapter 5–6, focuses on morphometrics and morphogenesis. A common approach to compare two two-dimensional (2D) shapes is to make use of mathematical transformations [139]. The Procrustes superimposition [55] uses rigid transformations up to rescaling to match prescribed landmarks but does not allow for exact landmark or boundary matching. The Thin Plate Spline (TPS) method [11] uses non-rigid landmark-based transformations to match two shapes but again does not allow for exact boundary matching or guarantee bijectivity. While the Large Deformation Diffeomorphic Method (LDDMM) [29,69,5] allows for the computation of diffeomorphic mappings of shapes with landmarks, the computation is expensive and thus hinders the pairwise comparison between a large set of shapes. More recently, conformal maps, which are angle-preserving maps, have been used for describing planar biological shape changes [113,1,94]. However, the rigidity of these maps imposed by the Riemann mapping theorem limits their flexibility in matching prescribed landmarks exactly. To get around with this problem, recent attempts [87,68] have considered the use of quasi-conformal maps, a generalization of conformal maps which allow certain degree of angular distortions with the presence of landmark constraints. In Chapter 5, we develop a landmark-matching mapping method for insect wing morphometry using quasi-conformal theory, which enables the pairwise comparison between insect wing shapes and provides a natural way for shape classification. We deploy the method for analyzing the phenotypic variation of *Drosophila* wings and the temporal development of Lepidoptera wings,

thereby yielding new insights into the growth and form of planar biological shapes.

After studying 2D biological shapes, we consider three-dimensional (3D) biological shapes. A prime example of 3D biological shapes are the highly convoluted brains. In Chapter 6, we study the gyrification of ferret brains. In recent decades, ferret brains have been widely studied for understanding neurodevelopmental processes [122,123,99,4,49]. By extending recent approaches based on the theory of differential growth [135,136], we model the growth of ferret brains using a physical gel model and a numerical simulation model. We compare the results with real ferret brain data to understand the process of brain development.

The third part of this thesis, consisting of Chapter 7–9, focuses on mathematical mapping techniques with applications to geometry processing and medical imaging. The problem of generating maps has been extensively studied by scientists and cartographers for centuries. A classical problem in map-making is about flattening the globe onto a plane. The well-known Mercator projection creates a planar map of the globe in which angles and small objects are well-preserved while the area near the poles is largely distorted. In recent decades, there has been a vast number of works on cartogram generation [38,41,73,52,74]. In particular, Gastner and Newman [52] proposed a method for producing cartograms based on density diffusion. The method deforms a given map according to certain prescribed data defined on each part of the map by solving the diffusion equation. This approach has been widely used for visualizing sociological and biological data such as the global population, income and age-of-death [39], the disease evolution for epidemics [30], the amphibian species diversity [144], the democracies and autocracies of different countries [54], the race/ethnicity distribution of Twitter users [93], the rate of obesity [142], and the world

citation network [103]. Based on the above concept of diffusion-based map-making, we develop a novel method for producing surface density-equalizing maps in Chapter 7. More specifically, our method flattens any simply-connected open surfaces onto a free- or fixed-boundary domain on the plane, with different parts magnified or shrunk according to the prescribed density. The applications of the method to data visualization and surface remeshing are explored.

Noticing the importance of a flattened representation of organs in medical analysis, we develop an area-preserving mapping method for carotid artery visualization in Chapter 8. Carotid artery is highly related to stroke, a leading cause of death and disability worldwide that causes an annual mortality of nearly 133,000 in United States [6] and over 1.6 million in China [82]. For facilitating the visualization and analysis of carotid plaques, various methods for flattening carotid artery surfaces onto the plane have been developed [3,152,21,23,24,18,25]. We develop a carotid flattening method that minimizes the area distortion and achieves a standardized boundary shape by extending the density-equalizing map and combining it with the reference map technique [71,141,114]. Our method allows for an accurate 2D representation of 3D carotid data, thereby aiding the prognosis and diagnosis of carotid diseases.

After extending the concept of density-equalizing maps for surface mapping, we further generalize it to 3D in Chapter 9. We develop a novel method for computing volumetric density-equalizing maps in a solid domain. The proposed method can be effectively applied to volumetric data visualization, deformation-based shape modeling, object morphing and adaptive remeshing.

Much of the work in this thesis appears in the following publications/preprints:

- (Ch. 1) G. P. T. Choi, L. H. Dudte, L. Mahadevan, *Programming shape using kirigami tessellations*. **Nature Materials**, 18, 999–1004 (2019). doi:10.1038/s41563-019-0452-y.
- (Ch. 2) G. P. T. Choi, L. H. Dudte, L. Mahadevan, *Reconfigurable kirigami*. In preparation.
- (Ch. 3) S. Chen*, G. P. T. Choi*, L. Mahadevan, *Deterministic and stochastic control of kirigami topology*. **Proceedings of the National Academy of Sciences**, 117(9), 4511–4517 (2020). doi:10.1073/pnas.1909164117. (*equal contribution)
- (Ch. 3) G. P. T. Choi, S. Chen, L. Mahadevan, *Control of connectivity and rigidity in prismatic assemblies*. Preprint, arXiv:2005.05845.
- (Ch. 4) L. H. Dudte, G. P. T. Choi, L. Mahadevan, *Additive origami*. Preprint, arXiv:2005.05846.
- (Ch. 4) G. P. T. Choi, L. H. Dudte, L. Mahadevan, *Additive kirigami*. In preparation.
- (Ch. 5) G. P. T. Choi, L. Mahadevan, *Planar morphometrics using Teichmüller maps*. **Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science**, 474(2217), 20170905 (2018). doi:10.1098/rspa.2017.0905.
- (Ch. 6) (with G. Séjourné, J. Y. Chung, R. S. Smith, C. A. Walsh, L. Mahadevan) *Ferret brain morphogenesis*. In preparation.
- (Ch. 7) G. P. T. Choi, C. H. Rycroft, *Density-equalizing maps for simply connected open surfaces*. **SIAM Journal on Imaging Sciences**, 11(2), 1134–1178 (2018). doi:10.1137/17m1124796.
- (Ch. 8) G. P. T. Choi, B. Chiu, C. H. Rycroft, *Area-preserving mapping of 3D carotid ultrasound images using density-equalizing reference map*. **IEEE Transactions on Biomedical Engineering** (2020). doi:10.1109/TBME.2019.2963783.
- (Ch. 9) G. P. T. Choi, C. H. Rycroft, *Volumetric density-equalizing reference map with applications*. Preprint, arXiv:2003.09725.

A mathematician, like a painter or poet, is a maker of patterns.

G. H. Hardy

1

Inverse kirigami design

Traditional kirigami structures consist of a periodic tiling of some prescribed deployable unit cell patterns. It is well known that the only regular polygons that can tile the plane are triangle, square and hexagon. All of them can used for the design of kirigami metamaterials:

- As shown in Figure 1.1**a**, the triangle kirigami tessellation (also known as the kagome

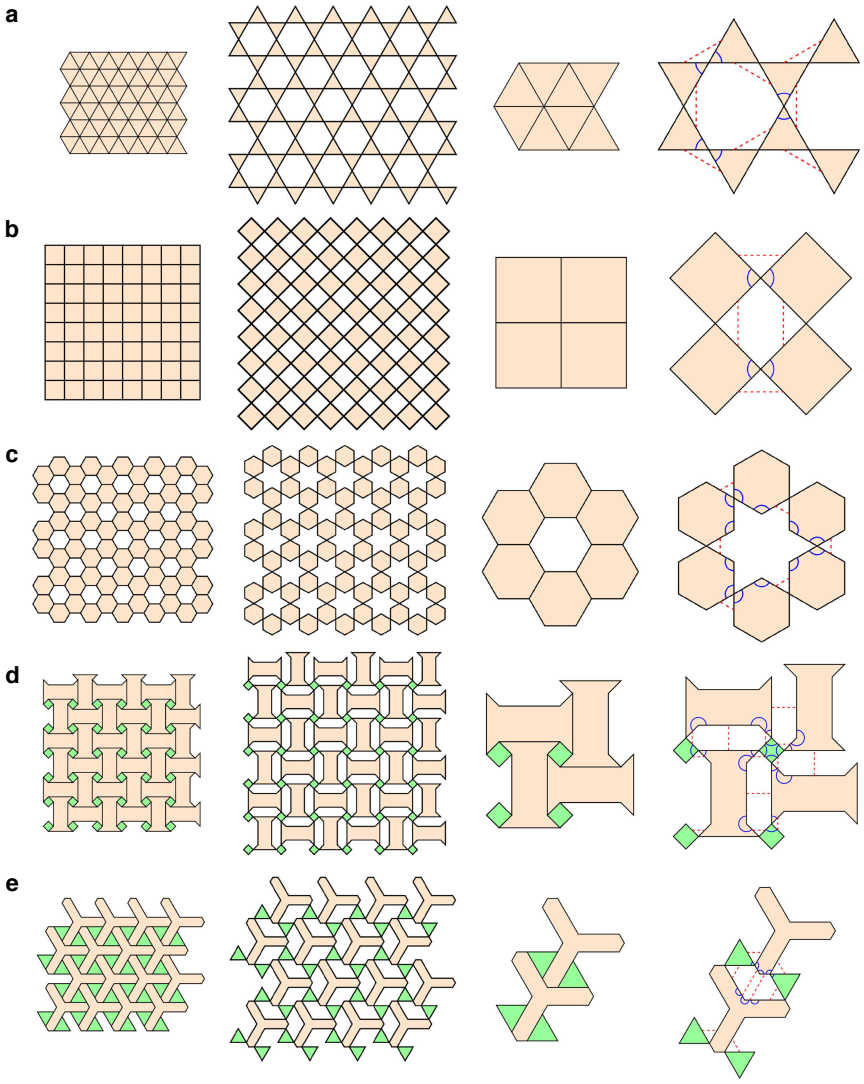

Figure 1.1: Five example deployable patterns. a, A triangle kirigami tessellation. **b**, A quad kirigami tessellation. **c**, A hexagon kirigami tessellation. **d**,**e**, Two multiple-cell Islamic kirigami tessellations. Corresponding edge pairs are connected by red dotted lines, and angles involved in the angle constraints are highlighted in blue.

pattern) is a floppy auxetic pattern with six triangles surrounding a single node.

- As shown in Figure 1.1**b**, the quad kirigami tessellation is a four-fold auxetic pattern with four squares surrounding a single node.

- As shown in Figure 1.1**c**, the hexagon kirigami tessellation is a six-fold auxetic pattern with

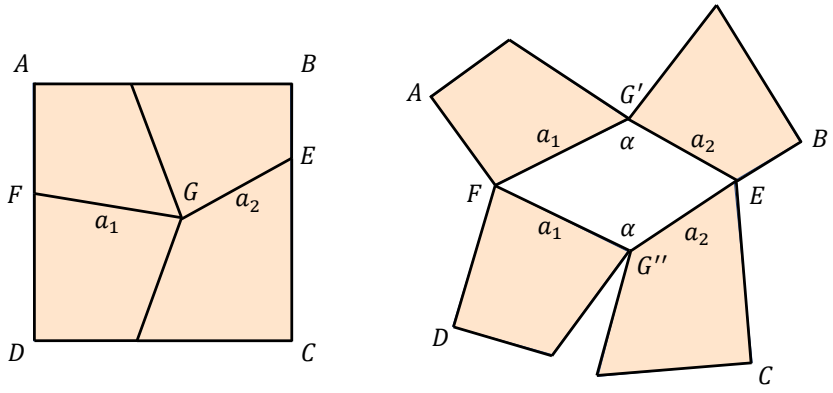

Figure 1.2: A generalized quad kirigami pattern. By changing the geometry of the four quads, it can be observed that the deployed shape (right) is no longer a simple enlarged version of the original kirigami structure (left).

six hexagons surrounding a hexagonal hole.

Besides the above simple tilings, there are many other more complex planar motifs [59,110] that can be the basis for a deployable kirigami pattern (e.g. the multiple-cell kirigami patterns derived from Islamic decorative tilings [110] as shown in Figure 1.1**d**,**e**).

Note that the repetitive kirigami patterns introduced above are limited in the sense that their deployed shape is fixed. No matter how we duplicate the unit cell along the vertical and the horizontal directions, the resulting deployable structures can only undergo a simple shape change upon deployment. For instance, the standard quad kirigami patterns can only be transformed from a closed and compact contracted rectangular shape into an enlarged rectangular shape with holes. In many situations, it is more desirable to have a kirigami structure that deploys and conforms to a *prescribed shape*.

As shown in Figure 1.2, if we change the geometry of the four tiles of a 2×2 quad kirigami

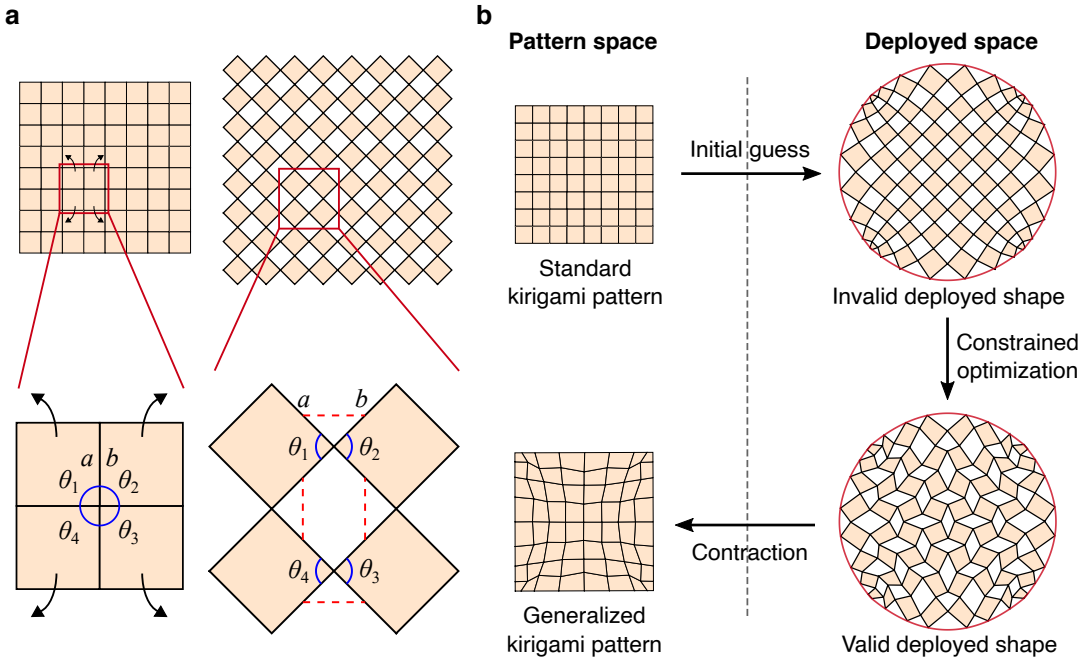

Figure 1.3: Inverse design framework. a, A quad kirigami tessellation and its deployed configuration, with a zoom-in of the unit cell of the quad kirigami tessellation and its deployed configuration. Every pair of corresponding edges are connected by a red dotted line. The set of angles corresponding to the same node are highlighted in blue. In a valid deployed configuration of a generalized kirigami pattern, every pair of edges should be equal in length, i.e. $a = b$, and every set of corresponding angles should add up to 2π, i.e. $\theta_1 + \theta_2 + \theta_3 + \theta_4 = 2\pi$. **b**, Our inverse design framework. Given a standard kirigami tessellation, we start with an initial guess in the deployed space. Here the initial guess shown is a conformal map from the standard deployed configuration to the disk. The initial guess is usually invalid, violating either the edge length constraint or the angle constraint, or not exactly matching the target boundary shape. We then solve a constrained optimization problem to morph the initial guess until it becomes a valid deployed shape, satisfying all constraints. Finally, we use a simple contraction procedure to obtain the generalized kirigami pattern.

pattern, the deployed shape will no longer be simply an enlarged version of the original pattern. This motivates us to consider the problem of *inverse kirigami design*: Given a standard kirigami pattern and a prescribed target boundary shape (such as a circle), how can we change the geometry of the tiles (i.e. the cut geometry) so that the generalized kirigami pattern deploys and conforms to the prescribed shape?

To answer this question, we propose an inverse design framework that involves solving a constrained optimization problem as outlined in Figure 1.3. More specifically, denote the fully deployed configuration of the given standard kirigami pattern by $\mathcal{D}$, and the target boundary shape together with its interior by $\mathcal{S}$. Our goal is to suitably deform $\mathcal{D}$ so that its boundary matches $\partial\mathcal{S}$, while ensuring that the deformed shape is a valid kirigami structure.

1.1 Generalized kirigami tessellations

In the following subsections, we introduce the constraints and then describe the procedure for solving the inverse design problem for generalized kirigami tessellations.

1.1.1 Contractibility constraints

To search for potentially admissible patterns in the deployed space, we first need to determine the constraints that allow a generalized deployed shape to be validly *contracted* (undeployed) into a closed and compact state. In particular, there should not be any mismatch or overlap in lengths and angles when the configuration is contracted. To achieve this, the following two *contractibility* constraints should be satisfied in the deployed space:

- *Edge length* constraints: For every pair of edges with edge lengths a, b in the deployed space that correspond to the same cut (e.g. the edge pairs connected by red dotted lines in Figure 1.1), we must have
$$a^2 - b^2 = 0. \tag{1.1}$$
- *Angle sum* constraints: Every set of angles in the deployed space that correspond to an interior node (e.g. the angles in the deployed space highlighted in blue in Figure 1.1) must

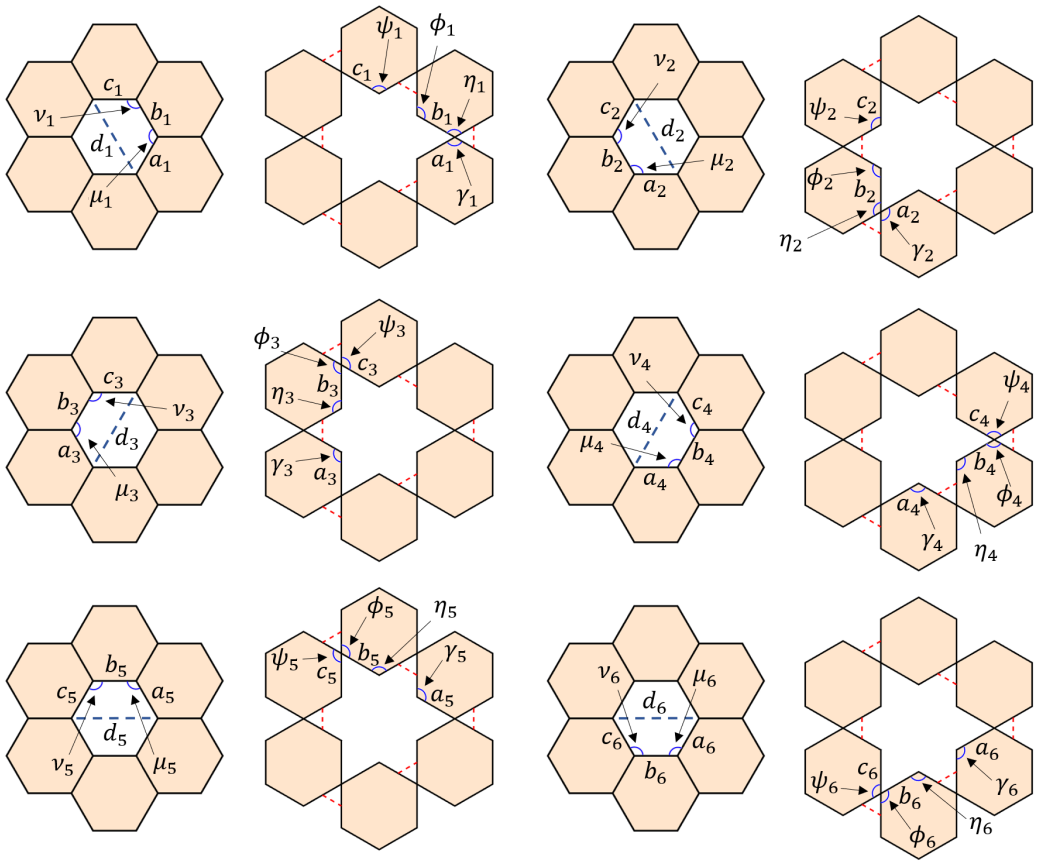

Figure 1.4: An illustration of the diagonal consistency constraints for the generalized hexagon kirigami patterns. Each row shows how the length of a diagonal of the hexagonal hole can be calculated in two ways using the angles and edge lengths in the deployed space.

sum to 2π:

$$\sum_i \theta_i = 2\pi. \tag{1.2}$$

In particular, for the quad kirigami pattern, the four pairs of edges at each unit cell in the deployed space (as shown in Figure 1.3) should be equal in length, and the four angles $\theta_1, \theta_2, \theta_3, \theta_4$ at each unit cell in the deployed space should satisfy $\theta_1 + \theta_2 + \theta_3 + \theta_4 = 2\pi$.

A modified version of the angle sum constraints for patterns with holes

Note that the angle sum constraints above are only applicable to kirigami base patterns that do not contain any hole in the contracted state. For those base patterns with holes (such as the hexagon tiling in Figure 1.1c), the angle sum constraints should be replaced by the following constraints:

- *One-ring angle sum* constraints: For the hexagon kirigami pattern, every hexagonal hole is surrounded by six hexagons. As the angle sum of an n-sided polygon is $(n-2)\pi$, the six angles of the hexagonal hole should add up to 4π. Note that the explementary angles of them can be expressed using the twelve angles in the hexagonal one-ring highlighted in blue in Figure 1.1c. Therefore, in a valid deployed configuration of a generalized hexagon kirigami pattern, we should have

$$6 \times 2\pi - \sum_{i=1}^{12} \theta_i = 4\pi \Leftrightarrow \sum_{i=1}^{12} \theta_i = 8\pi, \tag{1.3}$$

  where θ_i are the angles in the deployed space highlighted in blue in Figure 1.1c.

- *Diagonal consistency* constraints: The ring angle sum constraints are insufficient to guarantee that the one-ring hexagonal faces form a closed loop, as there is no control on the edge lengths of the hexagonal holes. To ensure the closed loop condition, we impose the diagonal consistency constraints which involve the edge lengths of the hexagonal holes. As depicted in Figure 1.4, at every hole enclosed by six hexagonal faces, we should have

$$\begin{cases} d_1^2 - d_2^2 = 0, \\ d_3^2 - d_4^2 = 0, \\ d_5^2 - d_6^2 = 0, \end{cases} \tag{1.4}$$

  where each pair $\{d_1, d_2\}$, $\{d_3, d_4\}$, $\{d_5, d_6\}$ refers to a diagonal of the hole calculated in two ways. More explicitly, we have

$$d_i^2 = \left(a_i - \tfrac{b_i \sin \nu_i}{\sin(\mu_i+\nu_i)}\right)^2 + \left(c_i - \tfrac{c_i \sin \mu_i}{\sin(\mu_i+\nu_i)}\right)^2 + 2\left(a_i - \tfrac{b_i \sin \nu_i}{\sin(\mu_i+\nu_i)}\right)\left(c_i - \tfrac{c_i \sin \mu_i}{\sin(\mu_i+\nu_i)}\right)\cos(\mu_i+\nu_i), \tag{1.5}$$

  where

$$\mu_i = 2\pi - \gamma_i - \eta_i \tag{1.6}$$

and

$$\nu_i = 2\pi - \varphi_i - \psi_i. \tag{1.7}$$

Note that all the edge lengths a_i, b_i, c_i and the angles $\gamma_i, \eta_i, \varphi_i, \psi_i$ are information in the deployed space. Therefore, the diagonal consistency constraints can be imposed in our constrained optimization problem, which takes place in the deployed space.

While the above constraints are discussed in the setting of hexagon kirigami patterns, similar constraints can be established for other kirigami patterns with holes.

1.1.2 Boundary shape matching constraints

Suppose we are given a target deployed shape in terms of a 2D curve $\partial\mathcal{S}$ (such as the circle as shown in Figure 1.3), we need to enforce *boundary shape matching* constraints on the boundary of the deployed configuration so that all boundary nodes lie on the target shape. More explicitly, for every boundary node $\mathbf{p}_i$, we should have

$$\|\mathbf{p}_i - \widetilde{\mathbf{p}}_i\|^2 = 0, \tag{1.8}$$

where $\widetilde{\mathbf{p}}_i$ is the projection of $\mathbf{p}_i$ onto $\partial\mathcal{S}$ and $\|\cdot\|$ is the Euclidean 2-norm.

1.1.3 Non-overlap constraints

Note that the contractibility constraints ensure the consistency between corresponding angles and edges but do not prevent the existence of overlapping tiles in the configuration. To avoid such overlaps, we enforce the following *non-overlapping* constraints at every angle between two adjacent tiles:

$$\langle(\mathbf{b} - \mathbf{a}) \times (\mathbf{c} - \mathbf{a}), \vec{n}\rangle \geq 0. \tag{1.9}$$

Here, $\mathbf{a}$ and $\mathbf{b}$ are two nodes of a tile, $\mathbf{a}$ and $\mathbf{c}$ are two nodes of another tile, $(\mathbf{b}, \mathbf{a}, \mathbf{c})$ form a positive (right-hand ordered) angle between the two faces, and $\vec{n} = (0, 0, 1)$ is the outward unit normal.

1.1.4 Additional constraints for further controlling the geometry of the generalized kirigami patterns

The three sets of constraints described above are necessary for guaranteeing an admissible deployed configuration. Optionally, one can enforce additional constraints to further control the geometry of the generalized kirigami pattern. Below, we propose constraints for achieving different effects for each kirigami base pattern.

Rectangular and square boundary constraints for generalized quad kirigami patterns

For instance, for the generalized quad kirigami patterns, we can enforce the boundary of its contracted configuration to be a rectangle or even a square. This is achieved by imposing the following additional constraints.

- *Boundary angle* constraints: For every set of two angles ζ_1, ζ_2 in the deployed configuration that correspond to the same boundary node in the contracted configuration, we enforce

$$\zeta_1 + \zeta_2 = \pi. \tag{1.10}$$

  For the four angles $\xi_1, \xi_2, \xi_3, \xi_4$ in the deployed configuration that correspond to four corner angles in the contracted pattern, we enforce

$$\xi_1 = \xi_2 = \xi_3 = \xi_4 = \frac{\pi}{2}. \tag{1.11}$$

The above constraints ensure that the deployed configuration corresponds to a rectangular generalized kirigami pattern. To further enforce the shape to be a square, the following constraints on boundary length are needed.

- *Equal boundary length* constraint: We further enforce the width and the height of the generalized kirigami pattern to be equal in length, thereby producing a square. Denote the edges in the deployed configuration that correspond to the top boundary edges in the contracted pattern by $\vec{e}_{T_i}, i = 1, \ldots, m$, and those corresponding to the right boundary edges by $\vec{e}_{R_j}, j = 1, \ldots, n$. We enforce the following constraint:

$$\|\vec{e}_{T_1}\| + \|\vec{e}_{T_2}\| + \cdots + \|\vec{e}_{T_m}\| = \|\vec{e}_{R_1}\| + \|\vec{e}_{R_2}\| + \cdots + \|\vec{e}_{R_n}\|. \tag{1.12}$$

Regular boundary angle sum constraints for generalized kagome kirigami patterns

For the generalized kagome kirigami patterns, we can enforce them to be a rectangle up to a small zig-zag effect on the left and the right boundaries, at which the angle sum is a multiple of $\pi/3$. This is achieved by impose the following *regular boundary angle sum* constraints. For each boundary node, we denote the number of faces adjacent to it by k and the angles by $\zeta_1, \zeta_2, \ldots, \zeta_k$. We enforce

$$\sum_{i=1}^{k} \zeta_i = \frac{k\pi}{3}. \tag{1.13}$$

More specifically, the angle sum at the top and the bottom boundary nodes is enforced to be $\frac{3\pi}{3} = \pi$ and hence the top and the bottom boundaries will form two straight lines. The angle sum at the left and the right boundary nodes will be either $2\pi/3$ or $4\pi/3$, the angle sum at the two corner nodes

on the left will be $2\pi/3$, and that at the two corner nodes on the right will be $\pi/3$. As a result, the pattern will be a rectangle up to a small zig-zag effect on the two sides.

Regular angle constraints for generalized hexagon kirigami patterns

For the generalized hexagon kirigami patterns, we can enforce the following *regular angle* constraints to further regularize the geometry. For every angle θ in the deployed configuration, we enforce

$$\theta = \frac{2\pi}{3}. \tag{1.14}$$

Here, the choice of $2\pi/3$ is compatible with the one-ring angle sum constraints (1.3). Therefore, even with such restrictions on all angles, we are still able to obtain valid generalized hexagon kirigami patterns that deploy and conform to a large variety of shapes.

Regular shape constraints for generalized multiple-cell Islamic kirigami patterns

For the generalized multiple-cell Islamic kirigami patterns, we can regularize their geometry by imposing a few extra constraints.

- *Non-self-intersecting* constraints: When compared to the triangle, quad and hexagon patterns, the two multiple-cell Islamic patterns involve polygonal tiles which are thinner and with a larger number of sides. To avoid those tiles from having self-intersection, we can enforce inequality constraints analogous to the non-overlap constraints at the angles of those tiles. More specifically, we form vectors using the nodes of those tiles and enforce that the cross product of the vectors is consistent with the face normal.
- *Regular angle* constraints:

– Note that the four-fold Islamic pattern (Figure 1.1**d**) contains four sharp corners for each I-shaped face. To avoid the corners from being squeezed in the generalized patterns, we can enforce the following constraint for each of such angles θ:

$$\theta = \frac{\pi}{4}. \tag{1.15}$$

– Note that for the hex Islamic pattern (Figure 1.1**e**), each of the longer sides consists of three nodes with angle sum being π. To preserve this property in the generalized patterns, we can simply enforce an additional angle sum constraint for those nodes. Also, we can regularize the boundary of the generalized hex Islamic patterns by enforcing that all the boundary angles remain unchanged in the generalized deployed configuration.

1.1.5 Constrained optimization

Any configuration that satisfies all the above constraints will yield a valid generalized kirigami pattern that conforms to the prescribed shape upon deployment. To search for such configurations, we set up a constrained optimization problem in the deployed space with the above conditions being the constraints.

Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ be the coordinates of the nodes in the deployed space. To produce a smooth generalized kirigami tessellation without large gradients in the shapes of the tiles, we consider minimizing the following objective function:

$$\frac{1}{M}\sum_{i=1}^{M}\left(\sum_{j}(\alpha_{i_j}-\beta_{i_j})^2+\sum_{k}(a_{i_k}-b_{i_k})^2\right) \tag{1.16}$$

where $\alpha_{i_j}, \beta_{i_j}$ are a pair of corresponding angles in two adjacent cells and a_{i_k}, b_{i_k} are corresponding edge lengths in two adjacent cells, and M is the total number of pairs of adjacent cells, subject to the constraints (1.1),(1.2),(1.8),(1.9), as well as any extra constraints described above if applicable. We remark that the above objective function can be replaced with

The above objective function and all constraints can be expressed solely in terms of the $2N$ coordinates of the nodes $\mathbf{x}_1 = (x_1, y_1), \mathbf{x}_2 = (x_2, y_2), \ldots, \mathbf{x}_N = (x_N, y_N)$. Furthermore, all the derivatives of them have a simple closed form in terms of the $2N$ variables $x_1, \ldots, x_N, y_1, \ldots, y_N$. Therefore, we can easily solve the constrained optimization problem using MATLAB's built-in optimization routine `fmincon`, with the derivatives of the objective function and all constraints supplied through the `SpecifyObjectiveGradient` and `SpecifyConstraintGradient` options.

1.1.6 Initial guess in the deployed space

To solve the constrained optimization problem, an initial guess of the configuration in the deployed space is needed. Theoretically, any approximation that preserves the number and connectedness of the input regular kirigami pattern can be used as an initial guess. Four choices that we have considered are listed below:

- **Standard deployed configuration**: The standard fully deployed configuration $\mathcal{D}$ of a regular kirigami pattern can be used as an initial guess. Note that it clearly satisfies the contractibility constraints, while the boundary shape is usually very different from the target boundary curve $\partial\mathcal{S}$.
- **Standard deployed configuration with rescaling**: One can also rescale $\mathcal{D}$ according to $\partial\mathcal{S}$ to reduce the boundary mismatch error and use it as an initial guess. Note that the rescaled

configuration again satisfies the contractibility constraints, but the boundary mismatch error is nonzero in general.

- **Conformal map**: Another initial guess can be obtained by applying the Schwarz-Christoffel map to produce a non-rigid transformation of $\mathcal{D}$ that matches its boundary matches $\partial\mathcal{S}$. As conformal maps preserve angles, the violation in the angle sum constraints is usually small. However, the edge length constraints are usually violated as lengths are not preserved under conformal maps.

- **Quasi-conformal map**: We combine recent conformal parameterization methods [26,92] to obtain a conformal map $g : \mathcal{S} \to \mathcal{R}$ from $\mathcal{S}$ to a rectangle $\mathcal{R}$, and then apply a rescaling transformation $h : \mathcal{R} \to \mathcal{D}$ to achieve the height and width of $\mathcal{D}$. The composition map $f = (h \circ g)^{-1} : \mathcal{D} \to \mathcal{S}$ is then a quasi-conformal map with $f|_{\partial\mathcal{D}} = \partial\mathcal{S}$. Because of the rescaling transformation, all angles will be distorted uniformly under the mapping and hence the angles sum constraints are violated. Nevertheless, the distortion in edge lengths is smaller than that of the conformal map in general.

1.1.7 Contraction

After solving the above constrained optimization problem, we obtain a valid deployed configuration of a generalized kirigami structure that yields a closed and compact tiling. To obtain the contracted configuration of the structure, note that there is a 1-1 correspondence between every tile in the pattern space and every tile in the deployed space, with each pair of corresponding tiles being identical up to translation and rotation. Therefore, we can start with one tile in the optimization result, and subsequently rotate and translate the adjacent tiles one by one to close the gaps between every pair of corresponding edges. After all tiles are rotated and translated, we obtain the contracted configuration of the generalized kirigami structure.

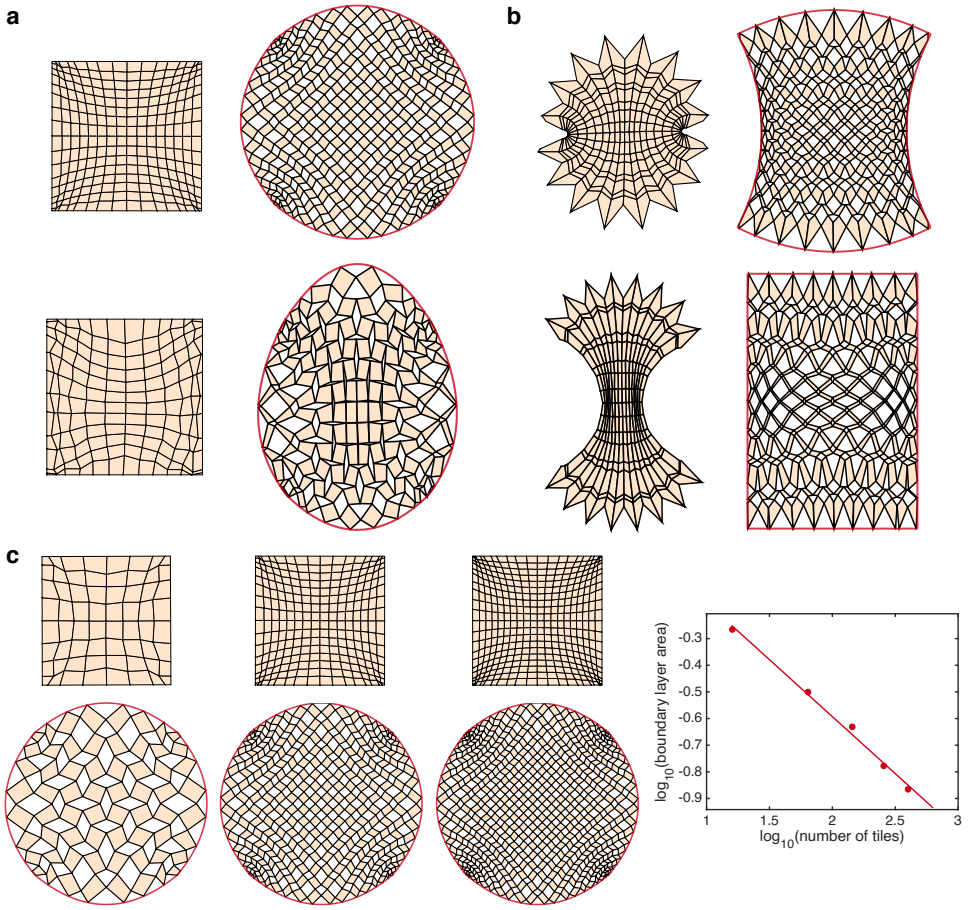

Figure 1.5: Generalized quad kirigami patterns. **a**, Examples of generalized quad kirigami patterns produced by our method for getting a circle or an egg shape from a square upon deployment. **b**, Examples of generalized kirigami patterns produced by our method for achieving target shapes with mixed curvature or zero curvature. Our method is capable of producing generalized kirigami patterns that matches boundary curves with different curvature properties when deployed. **c**, Examples of circling the square with different resolutions (number of tiles = $8 \times 8, 16 \times 16, 20 \times 20$), together with a log-log plot of the boundary layer area against the number of tiles and the least-square regression line.

1.2 Results

Using our proposed framework, we can obtain a large variety of generalized kirigami patterns. We start by showcasing various novel generalized quad kirigami patterns obtained by our method. As

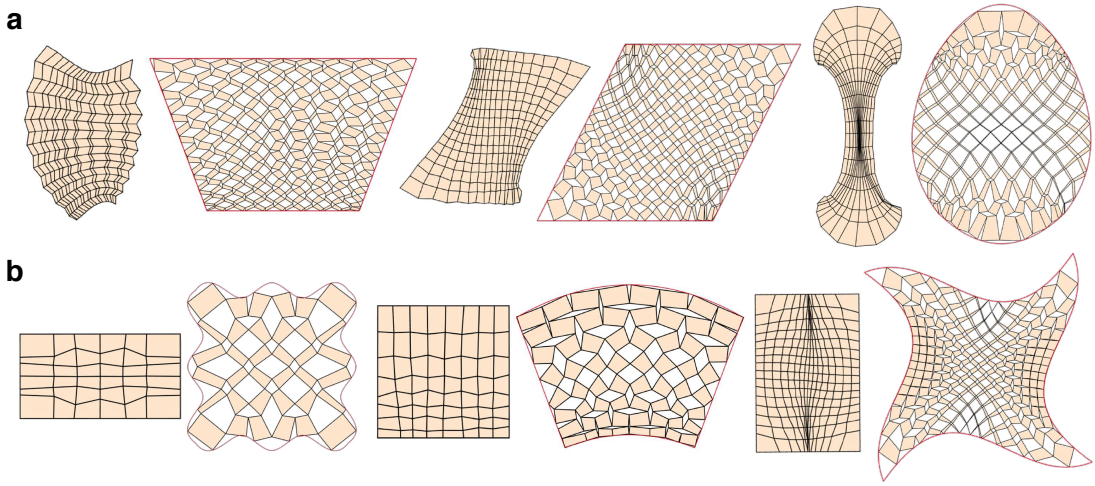

Figure 1.6: More generalized quad kirigami patterns. a, Generalized quad kirigami patterns with different target boundary shapes. **b**, Patterns obtained by imposing the additional rectangular or square boundary constraints described in Section 1.1.4.

shown in Figure 1.5**a**, by changing the cut geometry of a simple periodic quad kirigami tiling on a square, we can deploy a square and turn it into a circle or an egg. We can also use our method to create generalized kirigami patterns that approximate boundary shapes with mixed or zero curvature when deployed (Figure 1.5**b**). Note that the accuracy of the approximation can be improved by using a large number of smaller tiles, with an accuracy-effort trade-off in matching a prescribed shape. Figure 1.5**c** shows several generalized kirigami patterns of circling the square with different resolutions. It can be observed qualitatively that the boundary of the deployed pattern gets closer to a perfect circle as the number of tiles increases. To quantitatively assess the accuracy, we define the boundary layer area (denoted by A) of a generalized kirigami pattern by the total area of the gaps between the target boundary shape and the boundary of the deployed configuration. From the log-log plot, we observe that the boundary layer area decreases as the number of tiles (denoted by n^2)

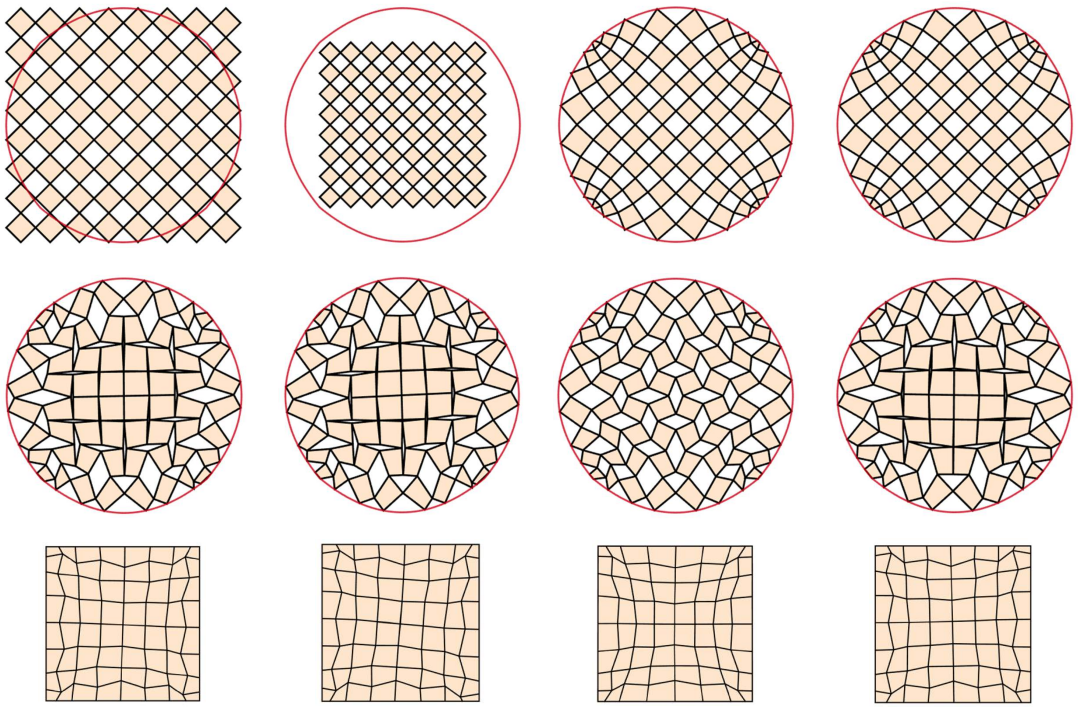

Figure 1.7: Generalized quad kirigami patterns obtained with different initial guesses. Each row shows the initial guess, the constrained optimization result, and the generalized kirigami pattern obtained. Left: The standard deployed configuration of the quad kirigami pattern. Middle left: The standard deployed configuration with rescaling. Middle right: Conformal map. Right: Quasi-conformal map.

increases, following the relation $A \propto (n^2)^{-1/2} = n^{-1}$. This can be explained by approximating every boundary gap by a triangle and measuring the change in the average triangle base length $\tilde{l}$ and average triangle height $\tilde{h}$ for different resolutions. Note that $\tilde{l} \propto n^{-1}$ and $\tilde{h} \propto n^{-1}$, and hence the average area of the triangles $\tilde{a} \propto n^{-2}$. As the total number of boundary gap triangles is approximately $4n$, we have $A \approx 4n\tilde{a} \propto n^{-1}$. More generalized quad kirigami patterns can be found in Figure 1.6.

It is noteworthy that the constrained optimization problem is in general underconstrained, and hence different initial guesses can possibly lead to different valid deployed configurations. Figure 1.7

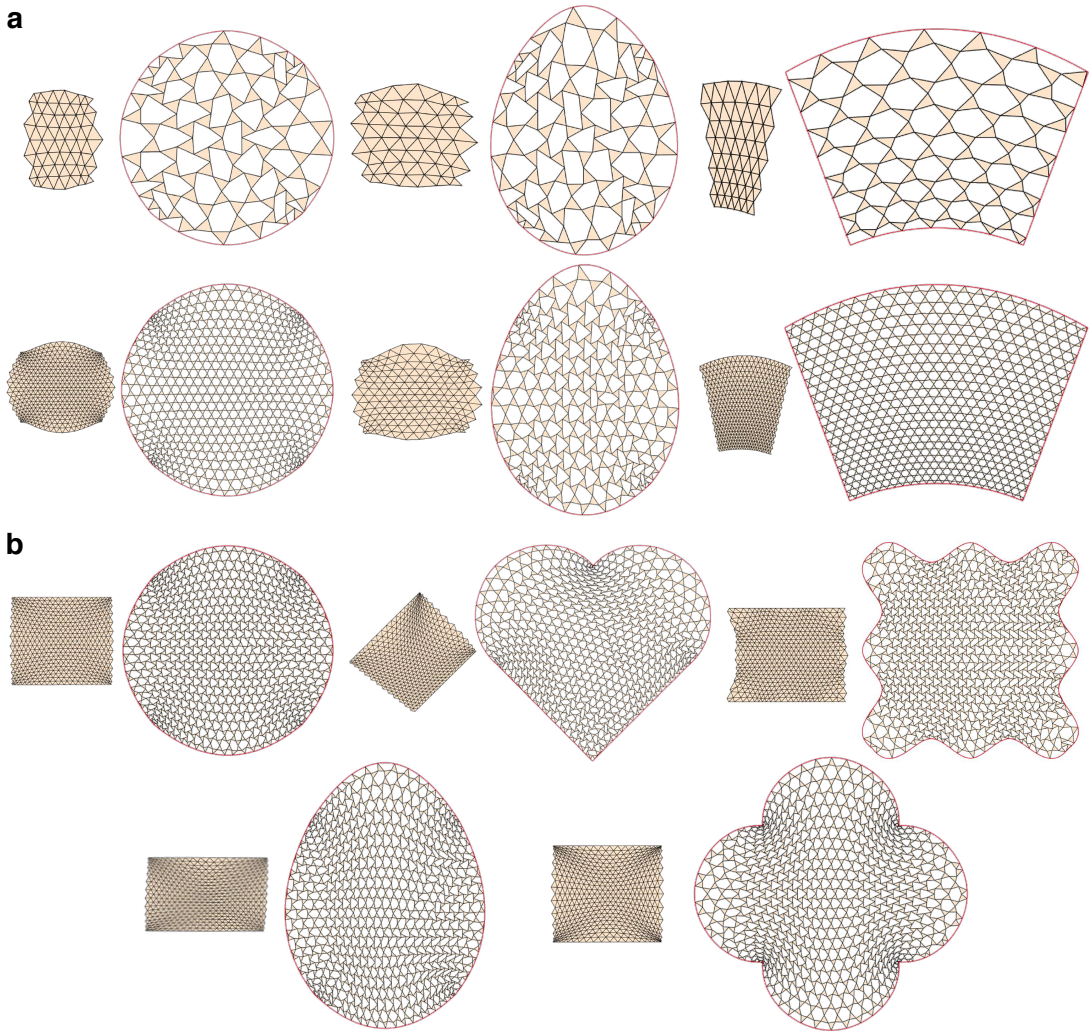

Figure 1.8: Generalized kagome kirigami patterns. a, Generalized kagome kirigami patterns with different target boundary shapes and different resolutions obtained by our approach. **b**, Patterns obtained by imposing the additional regular boundary angle sum constraints described in Section 1.1.4.

shows four generalized quad kirigami patterns of circling the square with the four different initial guesses described in Section 1.1.6. It can be observed that all four resulting generalized patterns are valid kirigami structures that can be deployed to approximate the same target circle, with a notable difference in their cut geometry.

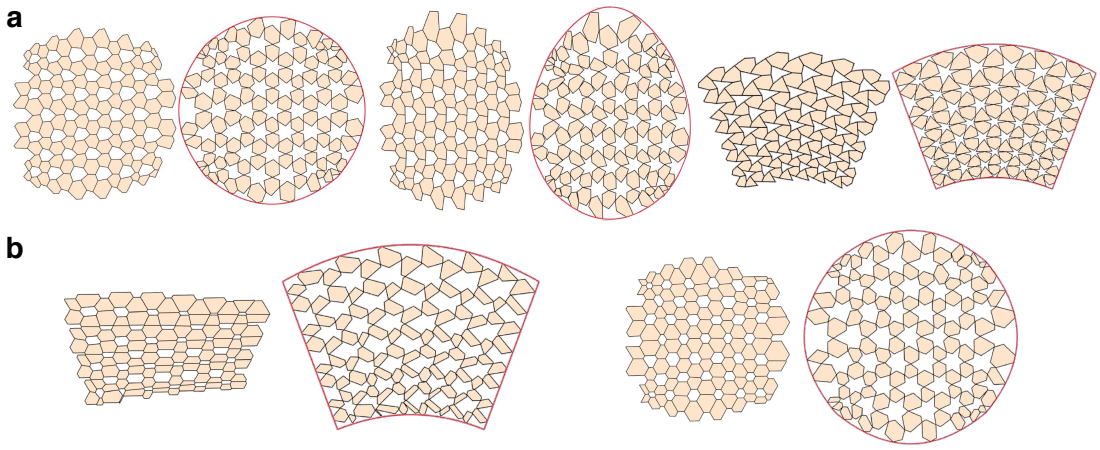

Figure 1.9: Generalized hexagon kirigami patterns. a, Generalized hexagon kirigami patterns with different target boundary shapes. **b**, Patterns obtained by imposing the additional regular angle constraints described in Section 1.1.4.

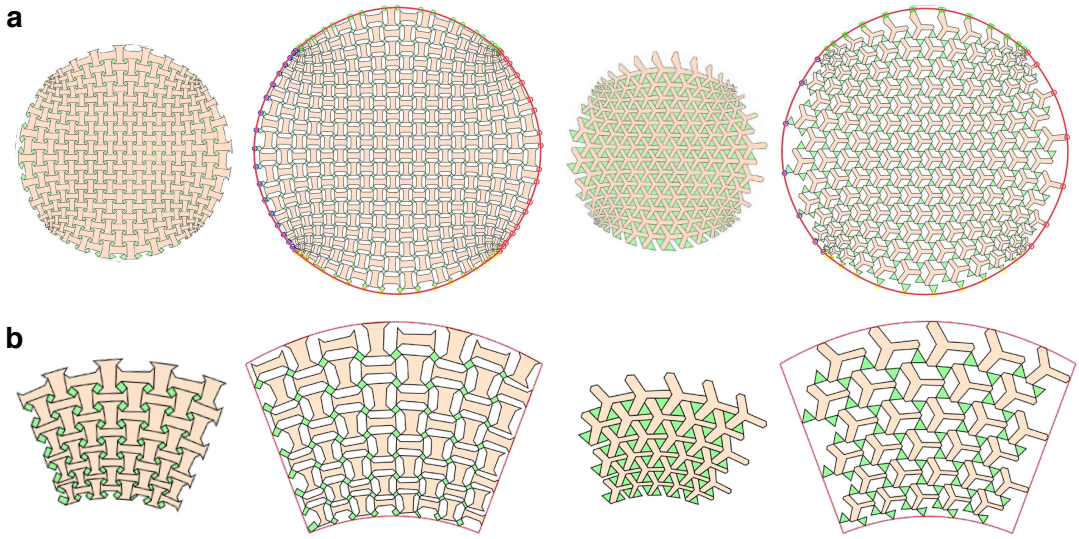

Figure 1.10: Generalized multiple-cell Islamic kirigami patterns. a, Generalized multiple-cell Islamic kirigami patterns whose deployed configurations approximate a circle. **b**, Patterns obtained by imposing the additional regular angle constraints described in Section 1.1.4.

Besides quads, our inverse design framework is also applicable to other kirigami base patterns. Figure 1.8, Figure 1.9, and Figure 1.10 show respectively various generalized kagome, hexagon, and

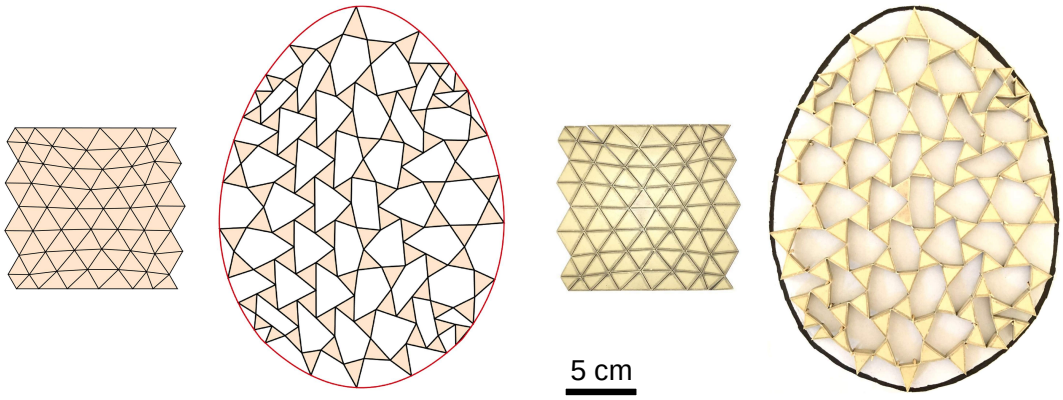

Figure 1.11: A generalized kirigami pattern for fitting an egg shape with a fabricated model. The two figures on the left show the undeployed and deployed configurations of the numerical optimization result obtained by our inverse design method. The two figures on the right show a fabricated model of the pattern and its deployed state. Pins are used to fix the position of the deployed fabricated model.

multiple-cell Islamic kirigami patterns that satisfy a large variety of target shapes when deployed. The additional constraints described in Section 1.1.4 can also be effectively imposed to produce more novel patterns.

For the realization of the generalized kirigami patterns obtained by our framework, we consider fabricating a physical model for the pattern shown in Figure 1.11 that transforms from a rectangular shape to an egg shape. The physical model was produced by laser cutting a thin sheet of highly-stretchable abrasion-resistant natural rubber. We pin the deployed state of the model and compare it with the pattern obtained by our framework. It can be observed that the physical model resembles the numerical optimization result very well.

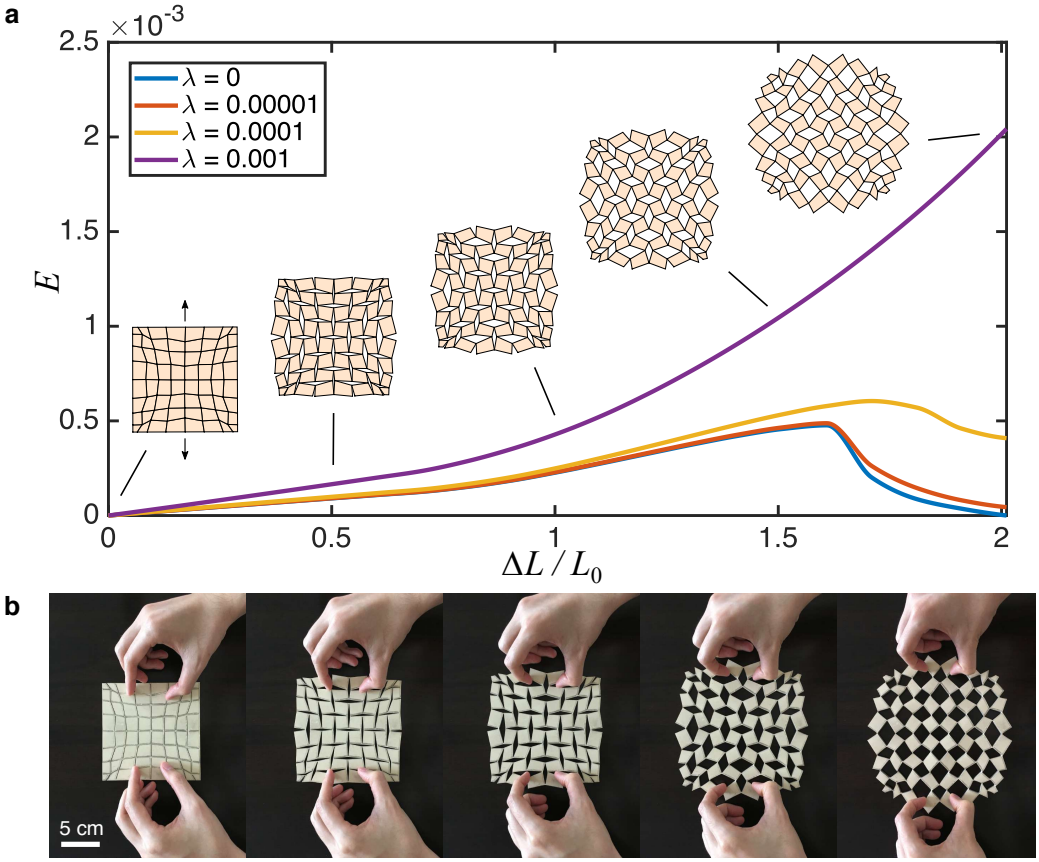

Figure 1.12: Planar deployment of generalized kirigami tessellation. a, Energetics of the deployment simulations of the square to circle example with different choices of λ. Here ΔL is the average displacement of the pulling points and L_0 is the average rest length of the extensional springs. The insets show the initial, intermediate and final configurations of the generalized kirigami pattern under deployment. **b**, Snapshots of the deployment of a monostable fabricated model.

1.3 Deployment energetic analysis

Our inverse design framework focuses on the contracted and deployed configurations of the generalized kirigami patterns but is agnostic to the *path* of the deployment. It is therefore natural to study the deployment process of the generalized kirigami patterns and the energetics.

Consider a linear spring model where linear springs are set along the edges and diagonals of the tiles of a generalized kirigami pattern, and simple torsional springs are set at the nodal hinges to model the ligaments that hold the structure together. The total mechanical energy of the system is given by

$$E(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N) = \frac{1}{N_s} \sum_{i,j} \left(\frac{\|\mathbf{x}_i - \mathbf{x}_j\| - l_{ij}}{l_{ij}} \right)^2 + \lambda \frac{1}{N_c} \sum_i \theta_i^2, \tag{1.17}$$

where $\mathbf{x}_i$ are the coordinates of the nodes, θ_i are the angles between every pair of edges created under the cuts, l_{ij} are the rest lengths of the extensional springs, N_s is the total number of extensional springs, N_c is the total number of torsional springs, and λ is the ratio of the torsional spring constant to the extensional spring constant. Note that a larger λ corresponds to a thicker ligament, which has a stronger tendency to close. We can then obtain a continuous deployment path of the system by iteratively moving the boundary nodes to the target boundary shape and solving for the intermediate deployed configurations, up to material deformations permitted by non-zero (1.17). Figure 1.12**a** shows the energetics of the deployment simulations with different λ. Note that if $\lambda \to 0$, we see the presence of bistability, while if $\lambda \not\to 0$, monostability or multistability can be observed. Figure 1.12**b** shows the deployment snapshots of a physical model fabricated by laser cutting a sheet of super-stretchable abrasion-resistant natural rubber. It can be observed that the simulated deployment path and real deployment have similar behaviors.

1.4 Inverse 3D kirigami design

While our inverse design approach has so far focused on approximating target planar shapes, it is in fact applicable for fitting surfaces in $\mathbb{R}^3$ as well. To achieve this, we first replace the boundary shape matching constraints (1.8) by the *surface matching* constraints so that every node $\mathbf{x}_i$ in the deployed configuration satisfies the condition

$$\|\mathbf{x}_i - \widetilde{\mathbf{x}}_i\|^2 = 0, \tag{1.18}$$

where $\widetilde{\mathbf{x}}_i$ is the projection of $\mathbf{x}_i$ onto the prescribed target surface S and $\|\cdot\|$ is the Euclidean 2-norm. Note that the *contractibility* constraints for surface fitting are the same as Eq. (1.1) and Eq. (1.2). The additional constraints introduced in Section 1.1.4 for further controlling the pattern geometry can also be directly extended to the three-dimensional case.

As for the *non-overlap* constraints, note that we have to prevent adjacent tiles in the deployed configuration from overlapping or intersecting with each other. Therefore, we replace the unit normal $\vec{n}$ in Eq. (1.9) with the normal computed using one of the two adjacent tiles. In other words, we enforce the following inequality constraints for every pair of adjacent tiles in the deployed configuration:

$$\langle (\mathbf{b} - \mathbf{a}) \times (\mathbf{c} - \mathbf{a}), (\mathbf{c} - \mathbf{a}) \times (\mathbf{d} - \mathbf{a}) \rangle \geq 0, \tag{1.19}$$

where $\mathbf{a}$, $\mathbf{b}$ are two nodes of a tile, $\mathbf{c}$, $\mathbf{a}$, $\mathbf{d}$ are three nodes of another tile, $(\mathbf{b}, \mathbf{a}, \mathbf{c})$ form a positive (right-hand ordered) angle between the two tiles, and $(\mathbf{c}, \mathbf{a}, \mathbf{d})$ also form a positive (right-hand ordered) angle.

Note that in the two-dimensional case, all tiles are always planar under the constrained optimization process. However, in the three-dimensional case, the tiles are not necessarily planar. Therefore, we need to enforce the following *planarity* constraints in the constrained optimization problem. For every face F in the deployed configuration, the volume of the polyhedron associated with its vertices should vanish:

$$\text{Volume}(F) = 0. \tag{1.20}$$

In particular, for generalized quad patterns, the above constraint becomes

$$\langle (\mathbf{b} - \mathbf{a}) \times (\mathbf{c} - \mathbf{a}), \mathbf{d} - \mathbf{a} \rangle = 0, \tag{1.21}$$

where $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, $\mathbf{d}$ are the four vertices of the quad F.

Finally, the objective function (1.16) and the contraction process can also be directly extended for three-dimensional surface fitting. We can then obtain generalized kirigami patterns that deploy to approximate a prescribed surface by solving a constrained optimization problem using `fmincon` in MATLAB.

Figure 1.13 shows several generalized kirigami patterns that deploy to fit surfaces of varying complexity. Additionally, just as for the two-dimensional problem, we can impose additional boundary angle constraints to produce different pattern design effects, such as using rectangular quad patterns to fit either a hyperbolic paraboloid (Figure 1.13**a**) or an elliptic paraboloid (Figure 1.13**b**), or an egg-carton shape (Figure 1.13**c**). Figure 1.13**d** shows an example of fitting a

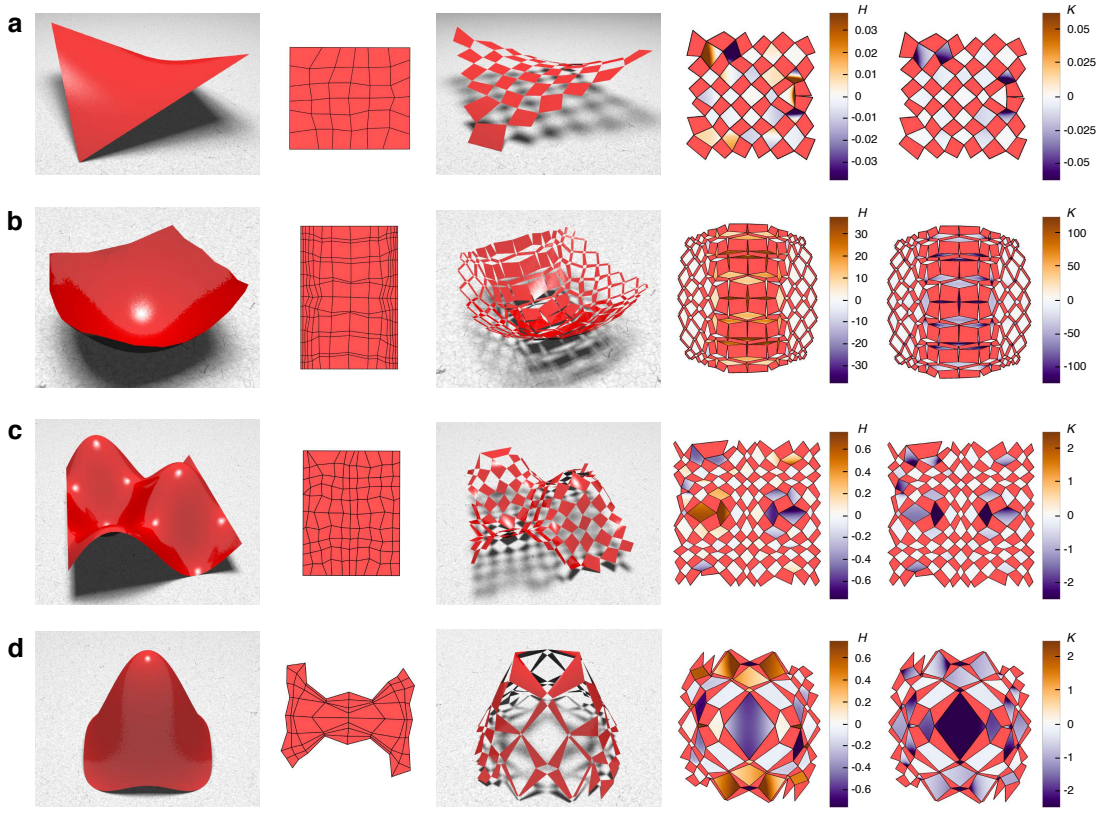

Figure 1.13: Generalized kirigami patterns for three-dimensional surface fitting. The target surfaces are **a**, a hyperbolic paraboloid (with negative curvature), **b**, a paraboloid (with positive curvature), **c**, a periodic patch of an egg-carton shape, and **d**, a bivariate Gaussian. Columns: The target surfaces (leftmost), the generalized kirigami patterns, the deployed configurations of the patterns that fit the target surfaces, the top views of the deployed patterns with the holes colored with the approximated mean curvature H, and the top views of the deployed patterns with the holes colored with the approximated Gauss curvature K (rightmost).

bivariate Gaussian with a more significant shape change upon deployment. It is noteworthy that all tiles in the deployed configurations are planar, while the surfaces are with non-zero curvature. This suggests that the curvature of the holes between the piece-wise planar tilings must be non-zero. To quantify this, we fit every hole by a bicubic Bézier surface and compute the mean curvature and the Gaussian curvature of it. It can be observed in Figure 1.13 that the holes between the planar tilings

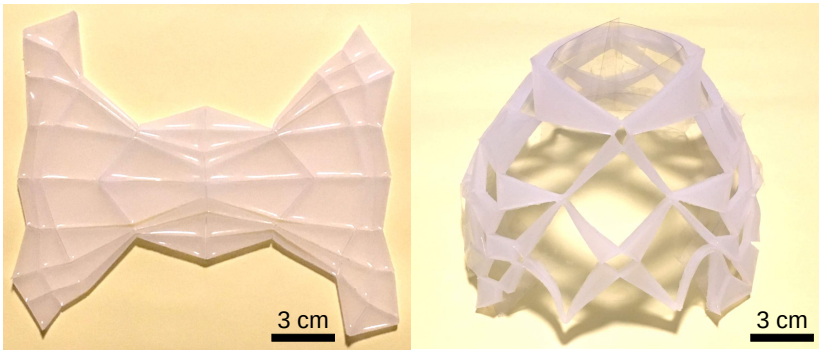

Figure 1.14: A physical model of a generalized quad pattern fabricated using PDMS. The model achieves a significant shape change and fits a hat-like surface (the underlying transparent sheet) upon deployment.

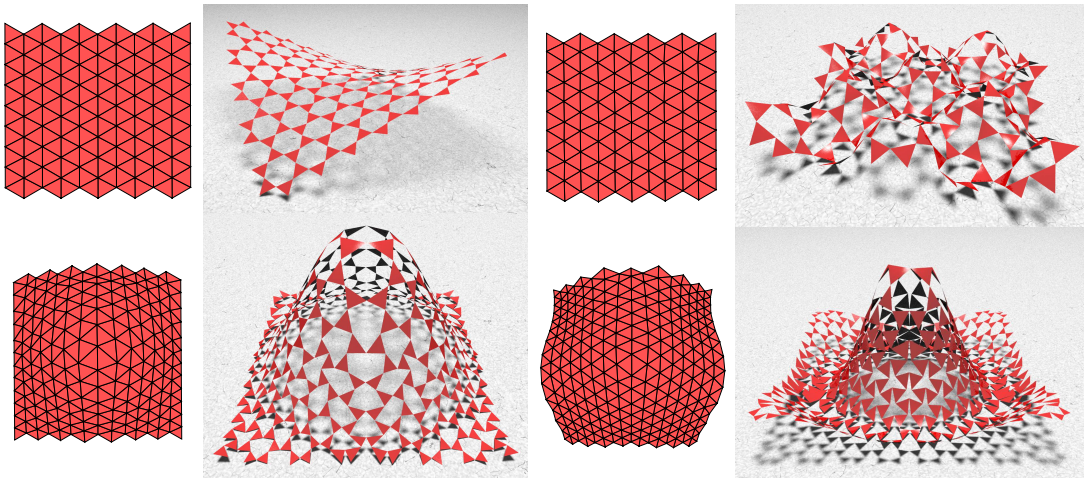

Figure 1.15: Generalized kagome kirigami patterns for surface fitting. The target surfaces are respectively a hyperbolic paraboloid, a landscape surface with multiple peaks, a bivariate Gaussian and a Mexican hat. For each target surface, the resulting generalized kirigami pattern and its deployed configuration are shown. It can be observed that our approach is capable of controlling the boundaries and the shape of the triangular faces of the generalized kirigami patterns for approximating different surfaces.

are indeed curved. Figure 1.14 shows a physical model of a generalized kirigami pattern, fabricated using Polydimethylsiloxane (PDMS). It can be observed that the deployed configuration of the fabricated model resembles our numerical optimization result very well. Figure 1.15 shows more generalized kirigami patterns for surface fitting obtained by our method, with the tiles being

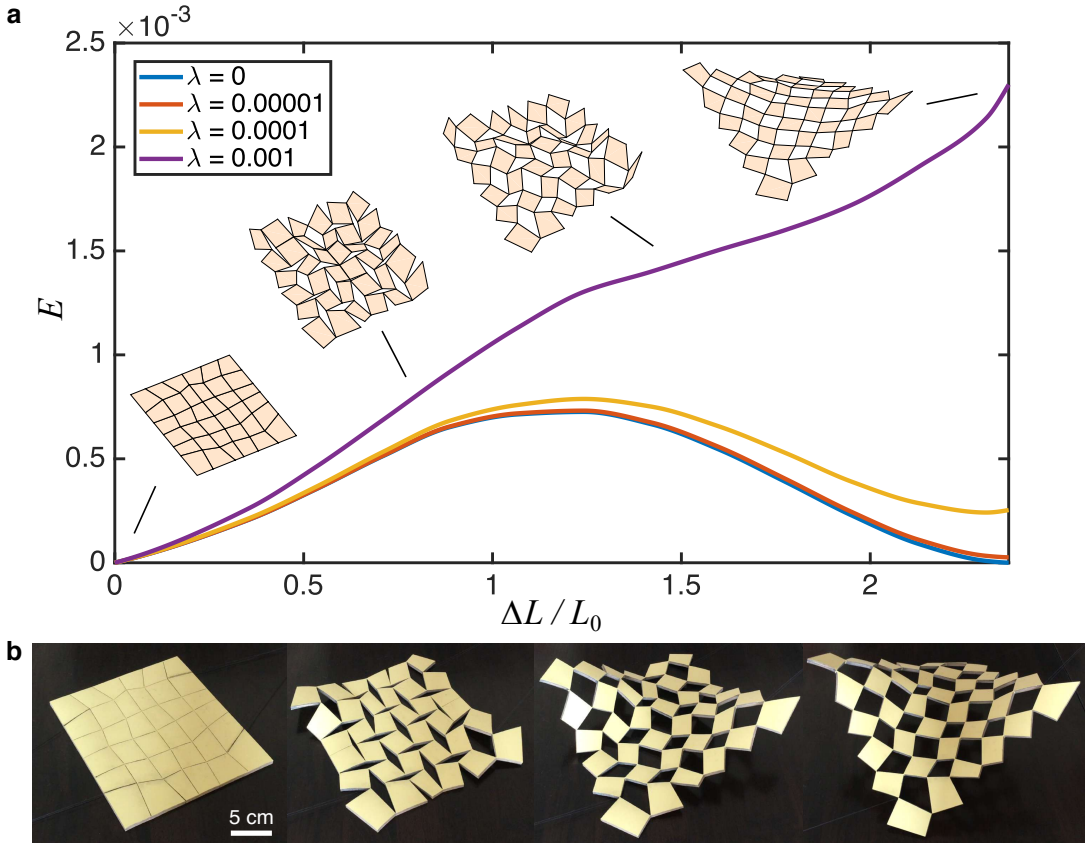

Figure 1.16: 3D deployment of generalized kirigami tessellation. a, Energetics of the 3D deployment simulations of the pattern in Figure 1.13**a** with different choice of λ. Here ΔL is the average displacement of the pulling points and L_0 is the average rest length of the extensional springs. The insets show the initial, intermediate and final configurations of the pattern under deployment. **b**, Snapshots of the deployment of a monostable fabricated model, with thin threads used for pulling the four sides. Both the numerical simulation and physical deployment results fit the hyperbolic paraboloid shape very well.

triangles instead of quads.

To study the deployment process in the three-dimensional case, we extend the planar energetic model to three dimensions, with an additional planarity constraint enforced to ensure that all tiles remain planar throughout the simulations. Figure 1.16**a** shows the deployment simulations with the four boundaries of a generalized kirigami pattern pulled towards the target positions for fitting a

hyperbolic paraboloid. While the intermediate states are warped, the final deployed configuration resembles the shape of a smooth hyperbolic paraboloid very well. As in the planar case, we see that the stability depends on λ. Figure 1.16**b** shows the deployment of a physical model fabricated using a thin sheet of natural rubber. It can be observed that the physical model also fits the target hyperbolic paraboloid very well.

1.5 Discussion

Our inverse design approach allows us to create generalized kirigami patterns that deploy and approximate any prescribed target shape in two- or three-dimensions. This is achieved by putting the essential constraints in lengths and angles together in a constrained optimization framework, with the flexibility of imposing additional constraints to further control the pattern geometry. More broadly, Our method provides a new way of engineering shape using geometry and computation.

I am interested in mathematics only as a creative art.

G. H. Hardy

2

Reconfigurable kirigami

In Chapter 1, we developed a novel inverse design framework for creating generalized kirigami patterns by identifying certain geometric constraints involving angles and edge lengths. More specifically, those patterns can be transformed from a closed and compact contracted shape to a deployed shape that approximates any prescribed target shape. Note that this process is reversible,

meaning that we can transform the deployed shape back to the original closed and compact contracted shape, as the conditions on the edge lengths and angles are satisfied in both the contracted and the deployed states.

A natural question that arises is as follows: Is it possible to design kirigami patterns that can be transformed from its deployed configuration to a closed and compact contracted shape different from the original one? In this chapter, we develop an inverse design framework for reconfigurable generalized kirigami patterns.

2.1 Constrained optimization for reconfigurable kirigami design

2.1.1 Formulation

To simplify our discussion, we focus on the quad kirigami patterns (Figure 2.1**a**). As discussed in Chapter 1, for guaranteeing that the deformed deployed configuration yields a valid kirigami pattern, we need to enforce the *contractibility constraints*, which consist of the edge length constraints (1.1) and the angle sum constraints (1.2).

To ensure that the deployed configuration can be contracted into another closed and compact contracted shape, we exploit the underlying duality of the kirigami pattern and formulate the following *reconfigurability constraints*:

- (*Dual edge length constraints*) For each pair of adjacent edges $\{\mathbf{e}_{i_2,1}, \mathbf{e}_{i_2,2}\}$ where $\mathbf{e}_{i_2,1}, \mathbf{e}_{i_2,2}$ belong to two different tiles and are not paired up in Eq. (1.1) (see the blue dotted lines in Figure 2.1**a**), we must have
$$l_{i_2,1} = l_{i_2,2}. \tag{2.1}$$

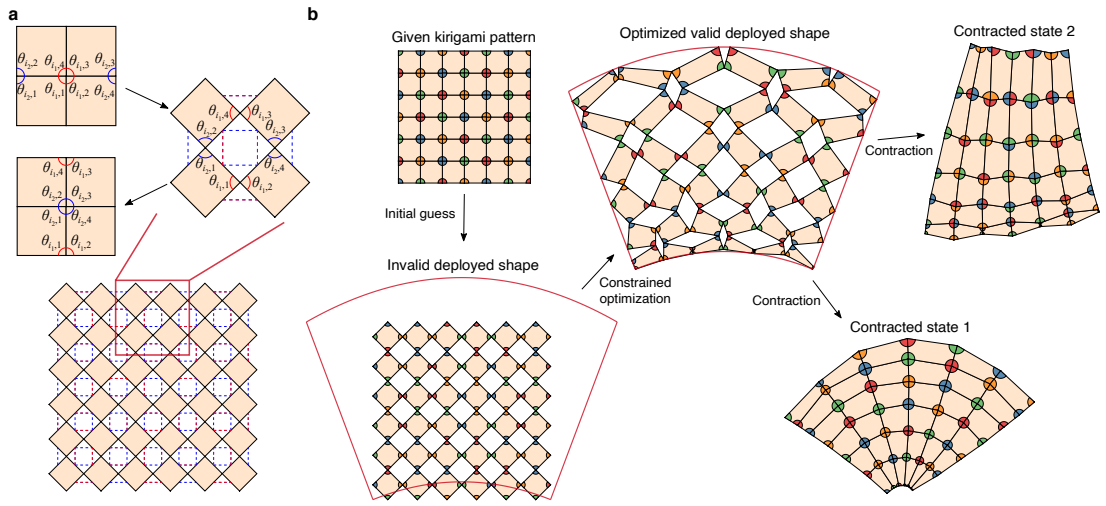

Figure 2.1: Reconfigurable kirigami design. a, An enlargement of the unit cell of a quad kirigami tessellation illustrating the constraints in edge lengths and angles to be satisfied. The red dotted lines indicate the ordinary edge pairs corresponding the same cuts, and the blue dotted lines indicate the dual edge pairs for getting the other contracted configuration. **b**, The inverse design framework for reconfigurable kirigami. Starting with a given kirigami pattern and a prescribed target shape, we construct an initial guess in the deployed space and solve a constrained optimization problem to obtain a valid deployed configuration that satisfies both the ordinary contractibility constraints and the new reconfigurability constraints and matches the target shape. We then contract the deployed configuration in two ways, one by following the cut edge pairs and one by following the dual edge pairs, and obtain two contracted states of it. The angles are colored based on the correspondence in the given kirigami pattern.

- (*Dual angle sum constraints*) For every set of angles $\{\theta_{i_2,k}\}_{k=1}^{n}$ which are dual to the set of angles $\{\theta_{i_1,k}\}_{k=1}^{n}$ mentioned in Eq. (1.2) inside a unit cell (see Figure 2.1**a**), we must have

$$\sum_{k=1}^{n} \theta_{i_2,k} = 2\pi. \tag{2.2}$$

The contractibility constraints and the reconfigurability constraints together enforce all edges around each hole in the deployed configuration to be equal in length. In other words, all holes in any reconfigurable generalized quad kirigami patterns must be rhombi. Also, the sum of all angles of the tiles at two opposite corners of each rhombus hole should be 2π.

Following the inverse design framework in Chapter 1, we obtain reconfigurable generalized

kirigami patterns by solving a constrained optimization problem (Figure 2.1**b**). Again, given any standard kirigami pattern and any target shape, we start by creating an initial guess in the deployed space. Then, we solve a constrained optimization problem in the deployed space to turn the initial guess into a valid deployed shape. This time, the constraints include the contractibility constraints (1.1),(1.2), the reconfigurability constraints (2.1),(2.2), as well as the shape matching constraints (1.8) and the non-overlap constraints (1.9). The objective function in Eq. (1.16) is used for regularizing the geometry of the entire pattern. We solve the problem numerically using the `fmincon` routine in MATLAB. Finally, different closed and compact contracted states can be obtained from the optimized deployed configuration, as both the contractibility constraints and the reconfigurability constraints are satisfied.

We remark that for symmetric target shapes, one can enforce an additional symmetry constraint on the coordinates of the nodes so that the deployed configuration is symmetric. This allows us to reduce the search space by half, with certain asymmetric admissible patterns neglected.

2.1.2 Results

Figure 2.2**a** shows several examples of reconfigurable kirigami patterns obtained by our method, where each of the kirigami patterns admits two distinct contracted states and the deployed configuration conforms to a prescribed target shape. As with the framework in Chapter 1, our method is capable of approximating target shapes with different curvature properties. Also, we can further control the boundary shape of a contracted state by introducing additional constraints on the boundary edge lengths and angles, yielding a reconfigurable kirigami pattern that deploys from a

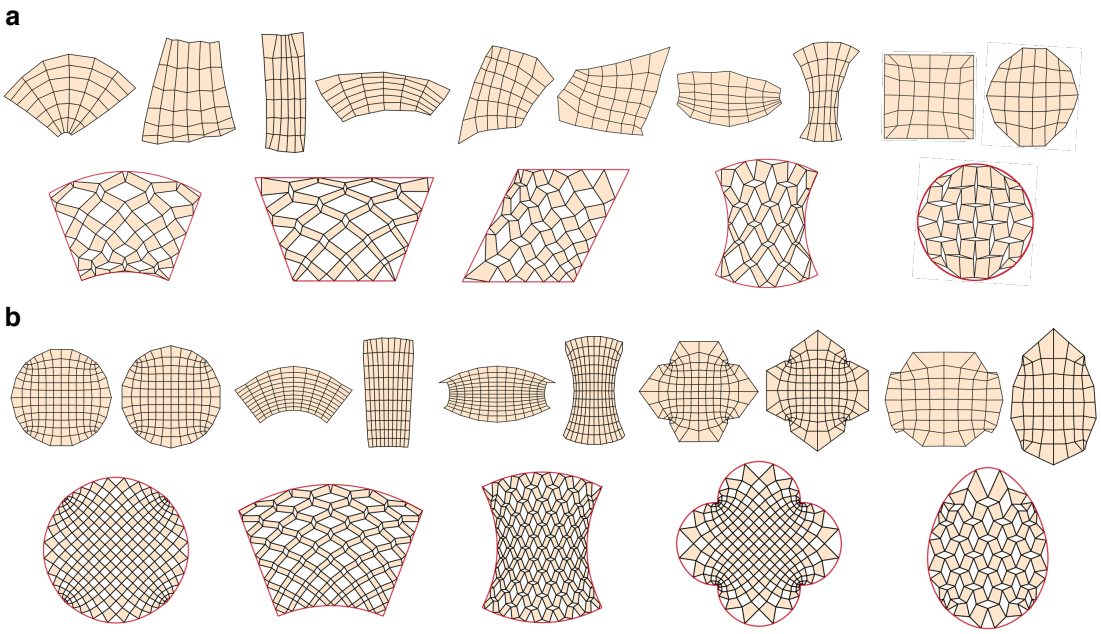

Figure 2.2: Reconfigurable generalized quad kirigami patterns obtained by our framework. a, Examples of reconfigurable generalized kirigami patterns that conform to prescribed target shapes with different curvature properties. For each example, the top row shows the two contracted states and the bottom row shows the deployed state. **b**, Results produced by further enforcing a symmetry constraint in the constrained optimization.

contracted rectangle to a circle and then contracts to another shape. As shown in Figure 2.2**b**, one can enforce a symmetry constraint to produce reconfigurable generalized kirigami patterns that are symmetric in the contracted and deployed states.

2.2 Enforcing rigid-deployability

In Chapter 1, we studied the energetics of generalized kirigami patterns and observed that in general the single degree-of-freedom of the generic quad kirigami pattern is lost after we change the cut geometry. Motivated by the geometric constraints for achieving reconfigurability, it is natural to ask whether rigid-deployability can be achieved by enforcing some other geometric constraints on the

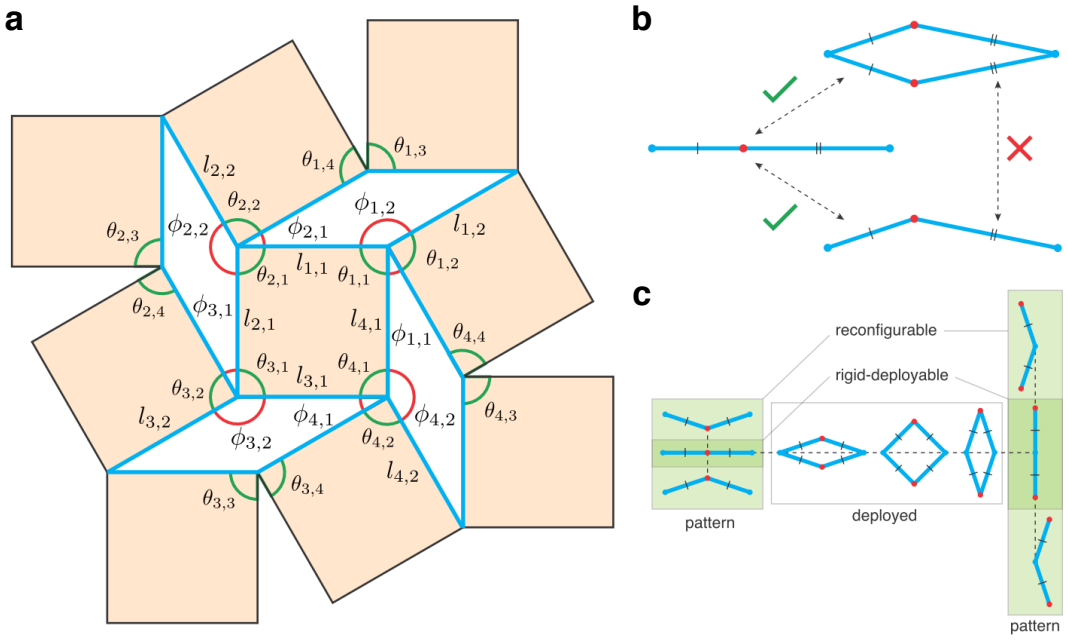

Figure 2.3: Rigid-deployability of kirigami patterns. a, Note that every negative space (blue) formed by a generic deployed quad kirigami structure (not necessarily reconfigurable) is a four-bar linkage with two pairs of adjacent edges having the same length. **b**, Such a linkage has two one-dimensional rigid deployments connected by a single branch point, the configuration with all edges collinear and an angle of π between overlapping edge pairs at the common hinge (red). The linkage can deploy rigidly from the branch point into either deployment paths, but cannot rigidly transform directly between points on the deployment paths while remaining embedded in two dimensions. **c**, If the reconfigurability constraints are enforced, all links in a four-bar linkage negative space have the same length. Such a linkage has three rigid deployments, one non-trivial path in which all angles between links are activated and two degenerate paths connected by branch points at the ends of the first path.

cut patterns.

2.2.1 Formulation

Here we propose a set of *rigid-deployability constraints* that further enforce the reconfigurable kirigami patterns are rigid-deployable, i.e. there exists a single continuous path from one contracted state to the deployed state and subsequently to another contracted state, such that none of the tiles

deform throughout the deployment process. Around every negative space, we enforce that

$$\theta_{i_1,1} + \theta_{i_1,2} = \theta_{i_2,1} + \theta_{i_2,2} = \pi, \tag{2.3}$$

where the design angles are as shown in Figure 2.1. Below, we prove that Eq. 2.3 is indeed necessary and sufficient.

Lemma. *Local rigid-deployability.* We first note that Eq. (2.3) ensures that each negative space forms a straight line in both contracted configurations. Taken in isolation, each negative space can be thought of as a four-bar linkage (see Figure 2.3**a**). A negative space from a generic quad kirigami pattern (i.e. not reconfigurable) has two unique edge lengths where edges with equal lengths are incident to each other (see Figure 2.3**b**). Such a four-bar linkage has two one-dimensional deployment paths in the plane connected to each other at two branch points, where the edges with equal lengths overlap each other and all edges are collinear. In the plane, the four-bar linkage cannot move from one deployment path to another except at a branch point. Thus, quad kirigami patterns which do not satisfy the rigid-deployability constraints (2.3) contain negative spaces which cannot pass from pattern to deployed states in the plane without changing the edge lengths. And, conversely, quad kirigami patterns which satisfy the rigid-deployability constraints (2.3) have only negative spaces which can rigidly deform from their straight-line pattern configurations to their solved, deployed configurations in the plane. Reconfigurable quad kirigami structures have negative spaces/four-bar linkages with all equal lengths. Such linkages have three one-dimensional deployment paths, one path in which all hinges are activated and the linkage forms a rhombus and

two degenerate paths in which two of the four hinges in the linkage are activated, each connected to the rhombus path at a respective branch point (see Figure 2.3**c**). Thus, reconfigurable quad kirigami patterns which satisfy the rigid-deployability constraints (2.3) have only negative spaces which can rigidly deform from their two straight-line pattern configurations to their solved, deployed configurations in the plane.

Theorem 2.1. *A reconfigurable kirigami pattern is globally rigid-deployable if and only if the constraints* (2.3) *are satisfied for all negative spaces.*

Proof. The above lemma provides local rigid-deployability if and only if Eq. (2.3) are satisfied for each negative space. To analyze global rigid-deployability, we construct a loop condition F around a single interior face in a generic (i.e. not necessarily reconfigurable) quad kirigami structure which must be identity at all points along a rigid-deployment. As shown in Figure 2.3**a**, let $\theta_{i,j}$ be design angles and $\varphi_{i,j}$ be deployment angles in a quad kirigami four-bar linkage negative space. Let f_i be the function that transfers a deployment angle $\varphi_{i,1}$ to the deployment angle $\varphi_{i+1,1}$ by composing angle-sum transfer h_i and four-bar kinematics transfer f_i such that

$$\varphi_{i+1,1} = f_i(\varphi_{i,1}) = g_i(h_i(\varphi_{i,1})) \tag{2.4}$$

$$\varphi_{i,2} = h_i(\varphi_{i,1}) = 2\pi - \varphi_{i,1} - \theta_{i,1} - \theta_{i,2} \tag{2.5}$$

$$\varphi_{i+1,1} = g_i(\varphi_{i,2}) = 2\sin^{-1}\left(\frac{l_{i,2}\sin\varphi_{i,2}}{\sqrt{l_{i,1}^2 + l_{i,2}^2 - 2l_{i,1}l_{i,2}\cos\varphi_{i,2}}}\right). \tag{2.6}$$

If the loop condition

$$F(\varphi_{1,1}) = f_4(f_3(f_2(f_1((\varphi_{1,1}))))) = \varphi_{1,1} \tag{2.7}$$

is satisfied for every value of $\varphi_{1,1} \in [0, 2\pi - \theta_{1,1} - \theta_{1,2}]$ for every interior quad, then the quad kirigami pattern is globally rigid-deployable. In a reconfigurable quad kirigami pattern, $\theta_{i,1} + \theta_{i,2} = \pi$ and $l_{i,1} = l_{i,2}$ and

$$\varphi_{i,2} = h_i(\varphi_{i,1}) = \pi - \varphi_{i,1} \tag{2.8}$$

$$\varphi_{i+1,1} = g_i(\varphi_{i,2}) = \pi - \varphi_{i,2} \tag{2.9}$$

$$\varphi_{i+1,1} = f_i(\varphi_{i,1}) = \varphi_{i,1}. \tag{2.10}$$

Hence, F is a composition of identity functions f_i and is itself identity. Therefore, reconfigurable quad kirigami patterns satisfying Eq. (2.3) are globally rigid-deployable. ■

Therefore, we can obtain reconfigurable generalized kirigami patterns which are rigid-deployable by simply adding Eq. (2.3) in the constrained optimization problem.

2.2.2 Results

Figure 2.4**a** shows examples of rigid-deployable, reconfigurable generalized kirigami patterns obtained by our method. It can be observed that our method is capable of producing a wide range of patterns to approximate different shapes even after enforcing the additional rigid-deployability constraints, and the accuracy of the approximation can be improved by increasing the resolution.

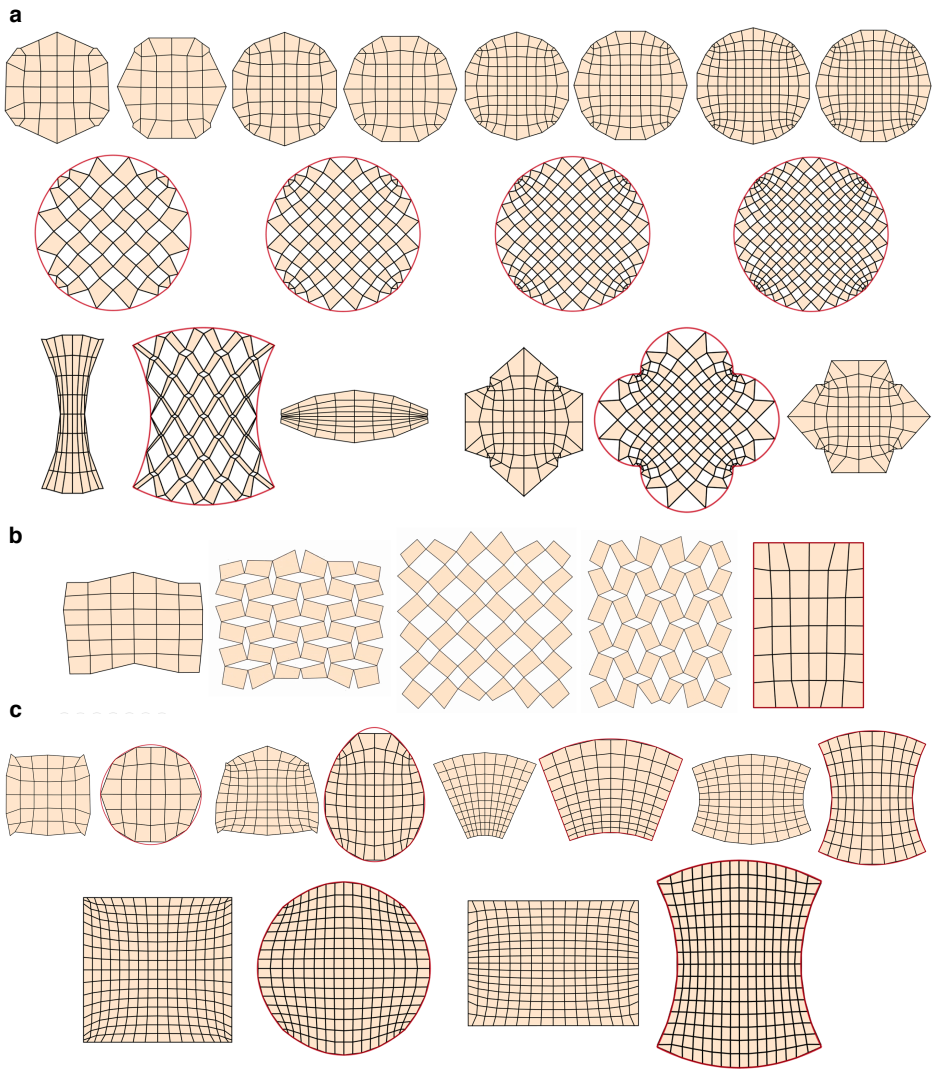

Figure 2.4: Rigid-deployable, reconfigurable generalized quad kirigami patterns. a, Examples of rigid-deployable, reconfigurable generalized quad kirigami patterns which conform approximately to a target shape when deployed. Note that the accuracy of the approximation is improved as the number of tiles increases. **b**, The simulated deployment of a pattern which is obtained by solving the constrained optimization problem directly on the contracted configurations without caring about the intermediate states. **c**, More rigid-deployable, reconfigurable generalized kirigami patterns obtained by this approach.

Moreover, in case the shape of the deployed configuration is not of our interest, it is possible for us to perform the constrained optimization directly on the two contracted configurations without

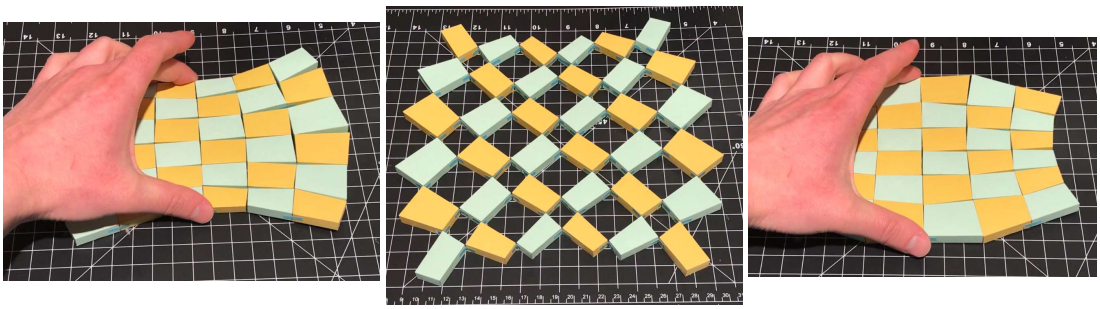

Figure 2.5: A physical model of a rigid-deployable, reconfigurable generalized kirigami pattern. It can be observed that the model can be contracted in two ways without any geometrical frustration.

caring about the intermediate states (Figure 2.4**b**). Figure 2.4**c** shows more rigid-deployable, reconfigurable generalized kirigami patterns obtained by this approach, including a pattern that can transform from a closed and compact square to a closed and compact circle. Figure 2.5 shows a physical model fabricated using cardboard papers with tape joints connecting the tiles. As the pattern is rigid-deployable, we can freely deploy and contract it without any geometrical frustration.

2.3 Discussion

We have demonstrated the effectiveness of our proposed inverse kirigami design framework by further incorporating additional geometric constraints that yield reconfigurability and rigid-deployability. While the focus of this chapter has been on the quad kirigami pattern only, our framework should also be applicable to other kirigami patterns. For instance, for the kagome kirigami pattern (Figure 1.1**a**), one can set up a set of reconfigurability constraints on top of the contractibility constraints. As shown in Figure 2.6, the two sets of edge length constraints together enforce that all edges of every hexagonal hole in a reconfigurable generalized kagome pattern should

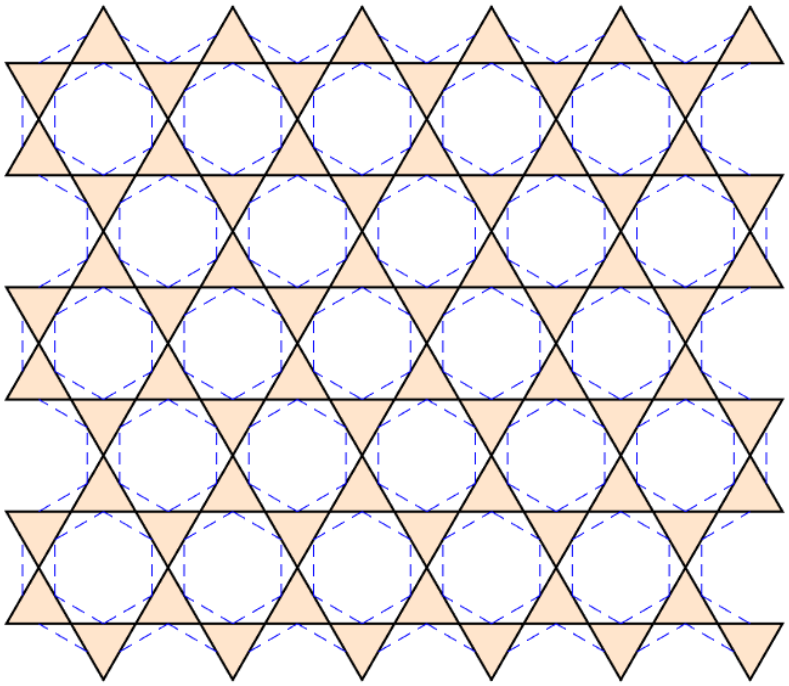

Figure 2.6: The edge length constraints for reconfigurable kagome patterns. At each negative space, the surrounding edges should all be equal in length.

be equal in length. Altogether, our study suggests that the cut geometry plays an important role in determining the structure and function of kirigami patterns.

Divide each difficulty into as many parts as is feasible

and necessary to resolve it.

René Descartes

3

Topological control of kirigami

Any structure is determined by its geometry and topology. In the previous two chapters, we have studied the geometric control of kirigami. Specifically, given a cut topology, we change the geometry of the tiles in order to achieve certain properties. In this chapter, we pose and solve a closely related problem: Given a cut geometry, how can we change the cut topology to

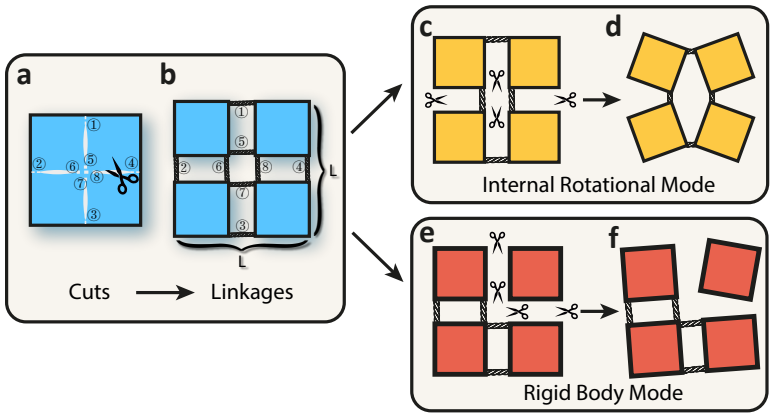

Figure 3.1: A kirigami system and two types of floppy modes. a-b, The cuts are along edges of square tiles except at the vertices, so that the pattern is equivalent to a linkage. **c-d**, Removing certain links may increase the DoF of the structure and add certain internal rotational mechanisms. **e-f**, Removing certain links may also increase the number of connected components (NCC) and add certain translational and rotational rigid body modes.

achieve certain prescribed properties?

Consider introducing cuts on a thin sheet of material with width and height both equal L to form a kirigami structure. To simplify our discussion, we assume that the material is cut using vertical and horizontal cuts along the grid lines with equal spacing 1, so that we have a rotating squares system with infinitesimally small corner hinges. Around the internal vertices, there are four small segments which we can independently decide on cutting or not. As shown in Figure 3.1**a**, all the black lines within this piece of paper have been cut, but near the vertices it is kept intact. This cut diagram can be transformed into a link diagram as shown in Figure 3.1**b**, where each small quad is separated, and each pair of neighbor nodes are connected via a "link".

Now, keeping the geometry of the square tiles fixed, we change the topology of the kirigami by determining how the cuts (links) are to be prescribed. More specifically, we would like to control the rigidity (Figure 3.1**c-d**) and connectivity (Figure 3.1**e-f**) of the kirigami system using the prescribed

cuts (links).

3.1 Rigidity control for 2D quad kirigami

To study the rigidity of the kirigami system, we first note that the infinitesimal degree-of-freedom (DoF) of a system is controlled by geometrical constraints associated with the tile geometry and the links. In particular, for every square tile $Q = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4\}$, we should have four *edge length constraints* and one *diagonal length constraint* that enforce the tile to be rigid:

$$\begin{cases} g_{\text{edge}}(\boldsymbol{x}_1, \boldsymbol{x}_2) = ||\boldsymbol{x}_1 - \boldsymbol{x}_2||^2 - l^2 = 0, \\ g_{\text{edge}}(\boldsymbol{x}_2, \boldsymbol{x}_3) = ||\boldsymbol{x}_2 - \boldsymbol{x}_3||^2 - l^2 = 0, \\ g_{\text{edge}}(\boldsymbol{x}_3, \boldsymbol{x}_4) = ||\boldsymbol{x}_3 - \boldsymbol{x}_4||^2 - l^2 = 0, \\ g_{\text{edge}}(\boldsymbol{x}_4, \boldsymbol{x}_1) = ||\boldsymbol{x}_4 - \boldsymbol{x}_1||^2 - l^2 = 0, \\ g_{\text{edge}}(\boldsymbol{x}_1, \boldsymbol{x}_3) = ||\boldsymbol{x}_1 - \boldsymbol{x}_3||^2 - 2l^2 = 0, \end{cases} \tag{3.1}$$

where l is the side length of the tile. Also, for each infinitesimal link connecting two nodes $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$, we should have two *link constraints*:

$$\begin{cases} g_{\text{link}_x}(\boldsymbol{x}_i, \boldsymbol{x}_j) = x_{i_1} - x_{j_1} = 0, \\ g_{\text{link}_y}(\boldsymbol{x}_i, \boldsymbol{x}_j) = x_{i_2} - x_{j_2} = 0, \end{cases} \tag{3.2}$$

where $\boldsymbol{x}_i = (x_{i_1}, x_{i_2})$ and $\boldsymbol{x_j} = (x_{j_1}, x_{j_2})$.

Therefore, for an $L \times L$ kirigami system with n links, there are in total $5L^2$ length constraints and

$2n$ link constraints. These constraints determine the range of motions associated with infinitesimal rigidity in terms of the rigidity matrix $\boldsymbol{A}$. Here, $\boldsymbol{A}$ is a $(5L^2 + 2n) \times 8L^2$ matrix with $A_{ij} = \partial g_i / \partial x_j$, where g_i is a length or link constraint ($i \in [1, 5L^2 + 2n]$), and j ranges from 1 to $8L^2$ (as there are in total $4L^2$ nodes in an $L \times L$ kirigami, and each node has two coordinates). Then, the infinitesimal DoF of the kirigami system can be computed by subtracting the number of independent constraints (i.e. the rank of $\boldsymbol{A}$) from $8L^2$ [61,84]:

$$\text{DoF} = 8L^2 - \text{rank}(A). \tag{3.3}$$

3.1.1 Minimum rigidifying link patterns (MRPs)

From the discussion above, as each link is associated with two link constraints, the decrease in the total DoF by adding one link must be either 0, 1 or 2. It is therefore natural to ask the following questions: What is the minimum number of links required for *rigidifying* an $L \times L$ kirigami, i.e. enforcing the system to have no extra DoF besides the global rigid body motion? How should these links be placed?

Define $\delta(L)$ as the minimum number of links for rigidifying an $L \times L$ kirigami, and a *minimum rigidifying link pattern* as a link pattern (a set of positions for links) that rigidifies the $L \times L$ kirigami system with exactly $\delta(L)$ links. It is easy to see that there are 3 DoF (2 translational and 1 rotational) if all possible links are added, and $3L^2$ DoF if none of them is added (as each tile has 3 DoF). Since each link reduces the DoF by at most 2, $\delta(L)$ links can at most reduce the DoF $2\delta(L)$. This implies

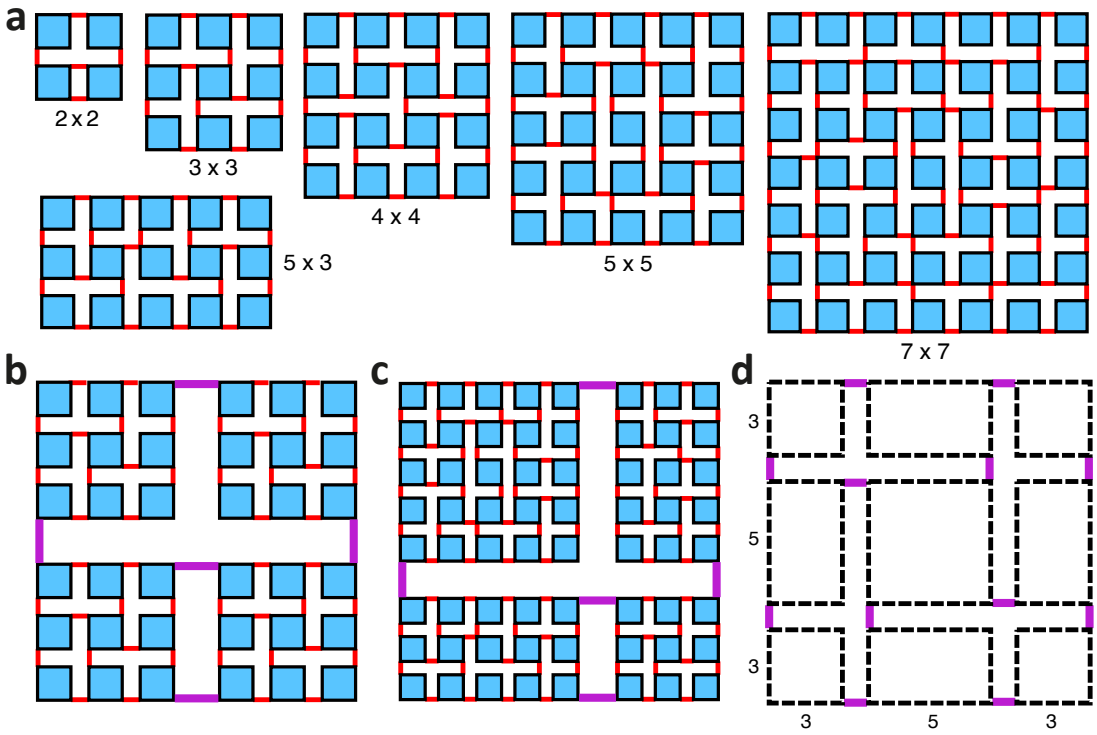

Figure 3.2: Minimum rigidifying link patterns (MRPs). **a**, Explicit examples of MRPs for $L = 2, 3, 4, 5, 7$, and for 3×5. **b**, An illustration of the hierarchical construction of MRPs. An MRP for $L = 6$ can be constructed by decomposing the 6×6 kirigami into four large blocks of size 3×3, and subsequently rigidifying every block using an MRP for $L = 3$ and then the four large rigid blocks using an MRP for $L = 2$. **c**. The hierarchical construction of an MRP for $L = 2^k, k \geq 3$ by decomposing the system into large blocks of size $3 \times 3, 5 \times 5$ and 3×5. **d**, The hierarchical construction of an MRP for odd primes $p \geq 11$.

that

$$3L^2 - 2\delta(L) \leq 3 \Rightarrow \delta(L) \geq \left\lceil \frac{3L^2 - 3}{2} \right\rceil . \tag{3.4}$$

In the following, we show that this lower bound for $\delta(L)$ is in fact optimal (achievable), and it is always possible to find a rigidifying link pattern with exactly $\left\lceil \frac{3L^2-3}{2} \right\rceil$ links for any L.

Theorem 3.1. *For all positive integer L,* $\delta(L) = \left\lceil \frac{3L^2-3}{2} \right\rceil$.

To prove Theorem 3.1, we first explicitly construct rigidifying link patterns with exactly $\left\lceil \frac{3L^2-3}{2} \right\rceil$

links for $L = 2, 3, 4, 5, 7$ (Figure 3.2**a**). The rigidity of these patterns is verified using the rigidity matrix computation described in Eq. (3.3). These explicit MRPs for small L form the building blocks for tackling the larger patterns.

Here we develop a *hierarchical construction* method for constructing MRPs for any system size L, where we combine the patterns for small L to construct the patterns for large L. For example, as illustrated in Figure 3.2**b**, we can decompose a 6×6 kirigami system into 4 large blocks of 3×3 tiles. We can then use an MRP with $\delta(3) = 12$ links to rigidify every 3×3 block, and then connect and rigidify the 4 large rigid blocks using an MRP for 2×2 with $\delta(2) = 5$ links. This results in a rigidifying link pattern for a 6×6 kirigami with the total number of links being

$$12 \times 4 + 5 = 53 = \left\lceil \frac{3(6^2) - 3}{2} \right\rceil . \tag{3.5}$$

More rigorously, the above hierarchical construction method suggests the following theorem:

Theorem 3.2. *For $L = 2^k \prod p_i^{n_i}$ where $k = 0, 1, 2$, p_i are odd primes that satisfy $\delta(p_i) = \left\lceil \frac{3p_i^2 - 3}{2} \right\rceil$, and n_i are nonnegative integers, we have $\delta(L) = \left\lceil \frac{3L^2 - 3}{2} \right\rceil$.*

Proof. For $k = 0$, we construct an MRP hierarchically as described below. Suppose L_1, L_2 are two odd numbers satisfying $\delta(L_1) = \left\lceil \frac{3L_1^2 - 3}{2} \right\rceil$ and $\delta(L_2) = \left\lceil \frac{3L_2^2 - 3}{2} \right\rceil$ (i.e. the lower bound for δ is optimal for L_1 and L_2). We construct a link pattern for $L = L_1 L_2$ by decomposing an $L_1L_2 \times L_1L_2$ kirigami system into $L_2 \times L_2$ large blocks of $L_1 \times L_1$ tiles. For every block of $L_1 \times L_1$ tiles, we use an MRP for L_1 to rigidify the block. Then, we connect and rigidify the $L_2 \times L_2$ large rigid blocks using an MRP for L_2. This results in a link pattern that rigidifies the entire $L_1L_2 \times L_1L_2$ kirigami, with the

total number of links being

$$\begin{aligned} L_2^2\delta(L_1)+\delta(L_2) &= L_2^2\left\lceil\frac{3L_1^2-3}{2}\right\rceil+\left\lceil\frac{3L_2^2-3}{2}\right\rceil \\ &= L_2^2\frac{3L_1^2-3}{2}+\frac{3L_2^2-3}{2} \\ &= \frac{3L_1^2L_2^2-3}{2}=\left\lceil\frac{3L_1^2L_2^2-3}{2}\right\rceil. \end{aligned} \tag{3.6}$$

This implies that $\delta(L_1L_2)=\left\lceil\frac{3L_1^2L_2^2-3}{2}\right\rceil$. By induction, The statement holds for any $L=\prod p_i^{n_i}$.

For $k=1$, we first use the above argument to construct an MRP for $\tilde{L}=\prod p_i^{n_i}$. Then, we decompose the $L\times L=2\tilde{L}\times 2\tilde{L}$ kirigami system into 4 large blocks of $\tilde{L}\times\tilde{L}$ tiles. Using the MRP for $\tilde{L}$ to rigidify each large block and an MRP for 2×2 to connect and rigidify all four of them, we obtain a rigidifying link pattern for the $L\times L$ kirigami with the total number of links being

$$\begin{aligned} 2^2\delta(\tilde{L})+\delta(2) &= 4\left\lceil\frac{3\tilde{L}^2-3}{2}\right\rceil+5 \\ &= 4\frac{3\tilde{L}^2-3}{2}+5 \\ &= \frac{3(2\tilde{L})^2-12+10}{2} \\ &= \frac{3(2\tilde{L})^2-2}{2}=\left\lceil\frac{3(2\tilde{L})^2-3}{2}\right\rceil. \end{aligned} \tag{3.7}$$

Therefore, the statement holds for any $L=2\prod p_i^{n_i}$.

Similarly, for $k=2$, we first construct an MRP for $\tilde{L}=\prod p_i^{n_i}$. Then, we decompose the $L\times L=4\tilde{L}\times 4\tilde{L}$ kirigami system into 16 large blocks of $\tilde{L}\times\tilde{L}$ tiles and rigidify each of them. Then we rigidity the entire structure using an MRP for a 4×4 kirigami, thereby producing a rigidifying link

pattern with the total number of links being

$$\begin{aligned} 4^2\delta(\tilde{L}) + \delta(4) &= 16\left\lceil \frac{3\tilde{L}^2-3}{2} \right\rceil + 23 \\ &= 16\frac{3\tilde{L}^2-3}{2} + 23 \\ &= \frac{3(4\tilde{L})^2 - 48 + 46}{2} \\ &= \frac{3(4\tilde{L})^2-2}{2} = \left\lceil \frac{3(4\tilde{L})^2-3}{2} \right\rceil. \end{aligned} \tag{3.8}$$

By induction, The statement holds for any $L = 4\prod p_i^{n_i}$. ■

Then, the corollary below follows immediately from the above theorem:

Corollary 3.1. *There exists infinitely many L such that* $\delta(L) = \left\lceil \frac{3L^2-3}{2} \right\rceil$.

We remark that the method used in the above proof cannot be directly extended for handling higher powers of 2. The reason is that the rounding error in the ceiling function may accumulate if the numerator is an odd number. To overcome this problem, we extend the definition of δ for general rectangular kirigami pattern by defining $\delta(M,N)$ as the minimum number of links required for rigidifying a $M \times N$ kirigami. It is easy to see that the lower bound for $\delta(M,N)$ is

$$\delta(M,N) \geq \left\lceil \frac{3MN-3}{2} \right\rceil. \tag{3.9}$$

By explicit construction, we obtained a rigidifying link pattern for a 3×5 kirigami system with 21 links (see Figure 3.2**a**). As $\left\lceil \frac{3(3\times5)-3}{2} \right\rceil = 21$, such a link pattern is an MRP for 3×5. We now prove the following theorem:

Theorem 3.3. *For any positive integer n, we have*

$$\delta(2^n) = \left\lceil \frac{3(2^n)^2 - 3}{2} \right\rceil . \tag{3.10}$$

Proof. The explicit construction in Figure 3.2**a** proves the statement for $n = 1, 2$. We prove the statement for the remaining n by induction. Suppose the statement is true for $n = k - 2$, i.e.

$$\delta(2^{k-2}) = \left\lceil \frac{3(2^{k-2})^2 - 3}{2} \right\rceil . \tag{3.11}$$

For $n = k$, we decompose the $2^k \times 2^k$ kirigami system into $2^{k-2} \times 2^{k-2}$ large blocks with size 3×3, 5×3, 3×5, and 5×5 (see Figure 3.2**c** for an illustration for $k = 3$). Using MRPs explicitly constructed for 3×3, 3×5 and 5×5, we rigidify every large block. Then, by the induction hypothesis, we have an MRP for $2^{k-2} \times 2^{k-2}$ that can be used for connecting and rigidifying all large blocks. This results in a rigidifying link pattern for the $2^k \times 2^k$ kirigami, with the total number of links being

$$\begin{aligned} &\frac{2^{k-2} \times 2^{k-2}}{4} \times (\delta(3) + \delta(5,3) + \delta(3,5) + \delta(5)) + \delta(2^{k-2}) \\ &= 2^{2k-6} \times (12 + 21 + 21 + 36) + \left\lceil \frac{3(2^{k-2})^2 - 3}{2} \right\rceil \\ &= 45(2^{2k-5}) + 3(2^{2k-5}) - 1 = 48(2^{2k-5}) - 1 = \frac{3(2^k)^2 - 2}{2} = \left\lceil \frac{3(2^k)^2 - 3}{2} \right\rceil . \end{aligned} \tag{3.12}$$

This implies that $\delta(2^k) = \left\lceil \frac{3(2^k)^2 - 3}{2} \right\rceil$. By induction, the statement holds for all n. ∎

Combining Theorem 3.2 and Theorem 3.3, it follows that $\delta(L) = \left\lceil \frac{3L^2 - 3}{2} \right\rceil$ for $L = \prod p_i^{n_i}$

where $p_i = 2, 3, 5, 7, \ldots$ are primes that satisfy $\delta(p_i) = \left\lceil \frac{3p_i^2-3}{2} \right\rceil$ and n_i are nonnegative integers. It remains to show that p_i can in fact be any prime. We make use of the following lemma:

Lemma 3.1. *Any odd number $L \geq 11$ can be written as*

$$L = 3m + 5n \tag{3.13}$$

where m and n are nonnegative integers.

Proof. Note that $11 = 3 + 3 + 5, 13 = 3 + 5 + 5, 15 = 5 + 5 + 5$ and $17 = 3 + 3 + 3 + 3 + 5$. For odd $L \geq 19$, we can express $L = (L - 8) + 3 + 5$. The result follows easily from induction. ■

The above lemma allows us to prove the following theorem:

Theorem 3.4. *For all primes $p \geq 11$,*

$$\delta(p) = \left\lceil \frac{3p^2 - 3}{2} \right\rceil. \tag{3.14}$$

Proof. Suppose the equality holds for all primes less than p. By Lemma 3.1, there exists nonnegative integers m, n such that $3m + 5n = p$. Since p is odd, $m + n$ is also odd. Also, since $m + n < 3m + 5n = p$, $m + n$ is either an odd prime or a product of odd primes which are smaller than p. Therefore, by the induction hypothesis as well as Theorem 3.2 and Theorem 3.3, we have

$$\delta(m + n) = \left\lceil \frac{3(m + n)^2 - 3}{2} \right\rceil. \tag{3.15}$$

Now, we decompose the $p \times p$ kirigami system into $(m+n) \times (m+n)$ large blocks with size $5 \times 5, 5 \times 3, 3 \times 5, 3 \times 3$ (see Figure 3.2**d** for an illustration for $p = 11$). Using MRPs explicitly constructed for 3×3, 3×5 and 5×5 and an MRP for $(m+n) \times (m+n)$ obtained above, we have a rigidifying link pattern for the entire $p \times p$ kirigami, with the total number of links being

$$\begin{aligned} & m^2\delta(3) + n^2\delta(5) + mn\delta(5,3) + mn\delta(3,5) + \delta(m+n) \\ =& 12m^2 + 36n^2 + 42mn + \left\lceil \frac{3(m+n)^2 - 3}{2} \right\rceil \\ =& 12m^2 + 36n^2 + 42mn + \frac{3(m+n)^2 - 3}{2} \\ =& \frac{3(9m^2 + 25n^2 + 30mn) - 3}{2} = \frac{3(3m+5n)^2 - 3}{2} = \left\lceil \frac{3p^2 - 3}{2} \right\rceil. \end{aligned} \tag{3.16}$$

This implies that $\delta(p) = \left\lceil \frac{3p^2-3}{2} \right\rceil$. By induction, the theorem holds for all primes $p \geq 11$. ■

Finally, using Theorem 3.2, Theorem 3.3, Theorem 3.4 and induction, we prove that $\delta(L) = \left\lceil \frac{3L^2-3}{2} \right\rceil$ for all L: If $L = 2^k \prod p_i^{n_i}$ where $k \leq 2$, by Theorem 3.2 we are done. If $k \geq 3$, we first construct an MRP for $\prod p_i^{n_i} \times \prod p_i^{n_i}$ and an MRP for $2^k \times 2^k$ using the three theorems above. Then, we decompose the $L \times L$ kirigami system into 2^{2k} large blocks of $\prod p_i^{n_i} \times \prod p_i^{n_i}$ tiles. Using the MRPs constructed for $\prod p_i^{n_i} \times \prod p_i^{n_i}$ $2^k \times 2^k$, we obtain a rigidifying link pattern for the entire

$L \times L$ system, with the total number of links being

$$\begin{aligned} 2^{2k}\delta\left(\prod p_i^{n_i}\right) + \delta(2^k) &= 2^{2k}\left\lceil \frac{3\left(\prod p_i^{n_i}\right)^2 - 3}{2} \right\rceil + \left\lceil \frac{3(2^k)^2 - 3}{2} \right\rceil \\ &= 2^{2k}\frac{3\left(\prod p_i^{n_i}\right)^2 - 3}{2} + \frac{3(2^k)^2 - 2}{2} \\ &= \frac{3\left(2^k \prod p_i^{n_i}\right)^2 - 3(2^{2k}) + 3(2^k)^2 - 2}{2} \\ &= \frac{3L^2 - 2}{2} = \left\lceil \frac{3L^2 - 3}{2} \right\rceil. \end{aligned} \tag{3.17}$$

This completes the proof of Theorem 3.1.

As a remark, we have

$$\lim_{L\to\infty} \frac{\delta(L)}{\text{Total number of links in an } L \times L \text{ quad kirigami}} = \lim_{L\to\infty} \frac{\left\lceil \frac{3L^2-3}{2} \right\rceil}{4L(L-1)} = \lim_{L\to\infty} \frac{3L^2/2}{4L^2} = \frac{3}{8}. \tag{3.18}$$

In other words, the MRPs for any $L \times L$ kirigami system require approximately $3/8$ of the total number of links as L is sufficiently large.

3.1.2 Algorithmic procedure of the hierarchical construction

Given any positive integer $L \geq 2$, the procedure for constructing an MRP for an $L \times L$ quad kirigami system is as follows:

- (Prime factorization) Compute the prime factorization $L = 2^k \prod_{i=1}^{m} p_i^{n_i}$ where $p_1, p_2, \ldots, p_m$ are distinct odd primes, $k \geq 0$ and $n_i \geq 1$ for all i (see Figure 3.3, top left).
- (MRPs for odd primes) For $p_i = 3, 5, 7$, take the explicitly constructed MRP for $p_i \times p_i$ given in Figure3.2**a**. For each $p_i \geq 11$, use the method in Theorem 3.4 to construct an MRP

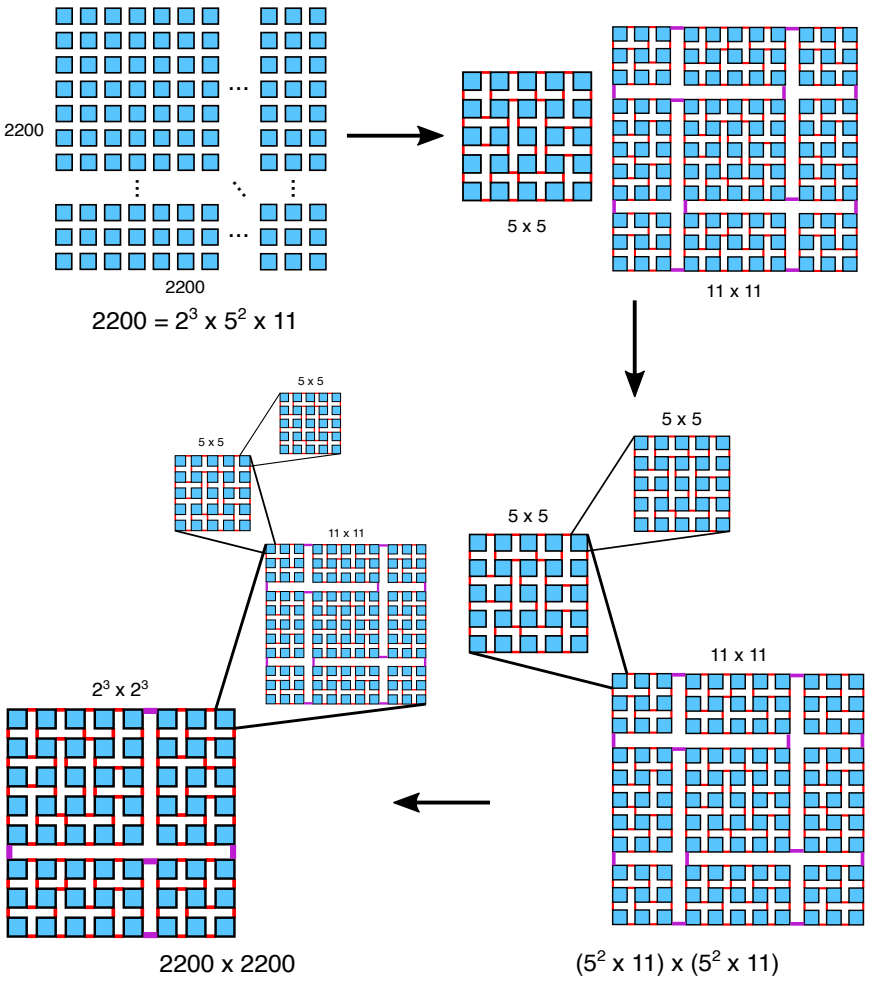

Figure 3.3: A flowchart of the hierarchical construction algorithm. To construct an MRP for an $L \times L = 2200 \times 2200$ quad kirigami, we first compute the prime factorization $2200 = 2^3 \times 5^2 \times 11$ (top left). Then, we take the explicitly constructed MRP for 5×5 given in Figure 3.2**a**, and construct an MRP for 11×11 using the method in the proof of Theorem 3.4 (top right). After getting MRPs for all prime factors, we construct an MRP for $(5^2 \times 11) \times (5^2 \times 11)$, i.e. the product of all odd prime powers of L, using the method in the proof of Theorem 3.2 (bottom right). Finally, we use the method in the proof of Theorem 3.3 to construct an MRP for $2^3 \times 2^3$, i.e. the largest even prime power of L, and subsequently apply the method in the proof of Theorem 3.2 again to construct an MRP for the entire $L \times L = 2200 \times 2200$ kirigami (bottom left).

for $p_i \times p_i$ by decomposing the $p_i \times p_i$ system into large blocks of size 3×3, 5×3, 3×5, and 5×5 (see Figure 3.3, top right).

- (MRP for the product of all odd prime powers) Use the approach in Theorem 3.2 to

construct an MRP for $p_i^{n_i} \times p_i^{n_i}$ for every i, and then construct an MRP for $\prod_{i=1}^m p_i^{n_i} \times \prod_{i=1}^m p_i^{n_i}$ using the hierarchical construction method (see Figure 3.3, bottom right).

- (MRP for the entire kirigami) If $k = 0$ we are done. If $k = 1, 2$, take the explicitly constructed MRPs for the 2×2 and 4×4 kirigami systems given in Figure 3.2**a**. If $k \geq 3$, use the approach in Theorem 3.3 to construct an MRP for $2^k \times 2^k$ by decomposing the $2^k \times 2^k$ system into large blocks of size 3×3, 5×3, 3×5, and 5×5. Finally, apply Theorem 3.2 again to construct an MRP for $L \times L$ by rigidifying the $2^k \times 2^k$ large blocks with size $\prod_{i=1}^m p_i^{n_i} \times \prod_{i=1}^m p_i^{n_i}$ (see Figure 3.3, bottom left).

We remark that the order of the operations described above is important. For instance, decomposing a 6×6 kirigami system into 4 large blocks of 3×3 tiles can yield an MRP, while decomposing it into 9 large blocks of 2×2 tiles cannot. The reason is that the order of such operations affects the parity of the numerator in the ceiling function in the calculation of the total number of links, which may make the resulting rigidifying link patterns suboptimal.

3.1.3 Enumeration of MRPs

Denote the number of MRPs for an $L \times L$ kirigami system by $n_r(L)$. Since the total number of possible links is $4L(L-1)$ and an MRP must have exactly $\delta(L) = \left\lceil \frac{3L^2-3}{2} \right\rceil$ links, there are in total $\binom{4L(L-1)}{\lceil (3L^2-3)/2 \rceil}$ possible combinations to examine for finding MRPs. For $L = 2$ and 3, it follows from a direct enumeration that there are $n_r(2) = 12$ and $n_r(3) = 140$ MRPs. Even for just $L = 4$ and 5, there are already $\binom{80}{36} \approx 3 \times 10^{13}$ and $\binom{80}{36} \approx 7 \times 10^{22}$ possibilities, making the enumeration impossible.

The computation can be simplified to a certain extent by assuming that all boundary links are connected. With this assumption, a direct enumeration shows that we have 4, 10 and 182280 MRPs

for $L = 2, 3, 4$ respectively. It can be observed that the number increases rapidly, and the enumeration again becomes impossible for larger L. Nevertheless, the hierarchical construction method provides us with a simple way to obtain a lower bound of $n_r(L)$ for composite L. For instance, as a 6×6 kirigami system can be decomposed into 4 large blocks of 3×3 tiles, the minimum number of MRPs for $L = 6$ is then $140^4 \times 12 \approx 4.6 \times 10^9$.

The above attempts for enumerating MRPs suggest that MRPs become extremely rare as L increases. In other words, it is nearly impossible to obtain an MRP simply by trial and error. This shows that the hierarchical construction is an effective method for obtaining MRPs.

3.1.4 Controlling rigidity using MRPs

Using the MRPs obtained by the hierarchical construction method, we can easily obtain kirigami systems with different rigidity properties. For instance, for odd L, since every link in an $L \times L$ MRP decreases the DoF of the kirigami system by exactly 2, we can obtain a system with DoF $= 2k + 3$ by removing exactly k links from an MRP. Also, by adding a link which reduces the DoF by 1 to such a kirigami system, we can obtain a system with DoF $= 2k + 2$. For even L, all but one links in an $L \times L$ MRP reduce the DoF of the system by 2 (except one that reduces the DoF by 1). By removing k links from an MRP, we can again obtain a kirigami with DoF $= 2k + 3$ or $2k + 2$. Therefore, any given DoF is achievable by suitably removing links from the MRPs.

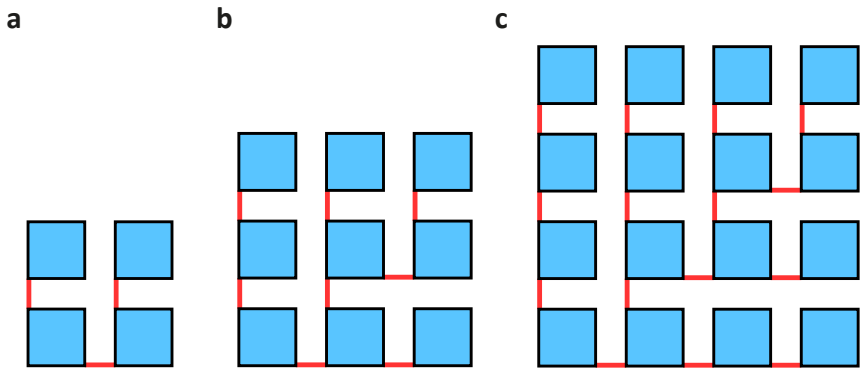

Figure 3.4: Minimum connecting link patterns (MCPs). Starting from an MCP for $L = 2$ (**a**), we add one link at each edge on the top and the right boundary. This produces an MCP for $L = 3$ (**b**). Repeating the same procedure, we obtain an MCP for $L = 4$ (**c**).

3.2 Connectivity control for 2D quad kirigami

We then proceed to consider how the number of prescribed links affects the connectivity of the kirigami system.

3.2.1 Minimum connecting link patterns (MCPs)

Analogous to the study of rigidity control, here we first consider the minimum number of prescribed links for making an $L \times L$ kirigami connected. Define $\gamma(L)$ as the minimum number of links required for connecting an $L \times L$ kirigami, and a *minimum connecting link pattern* (MCP) to be a link pattern with exactly $\gamma(L)$ links which connects the $L \times L$ kirigami. Clearly we have the following result:

Theorem 3.5. *For all positive integer* L, $\gamma(L) = L^2 - 1$.

A few examples of MLPs are given in Figure 3.4. Note that the hierarchical construction we

introduced for obtaining MRPs is also applicable for MCPs. Let m, n be two positive integers. Suppose we have an MCP for $m \times m$ and $n \times n$. If we decompose a $mn \times mn$ kirigami into $m \times m$ large blocks of $n \times n$ tiles, we can use the hierarchical construction method to obtain a connecting link pattern for the $mn \times mn$ kirigami, with the total number of links being

$$m^2\gamma(n) + \gamma(m) = m^2(n^2 - 1) + (m^2 - 1) = (mn)^2 - 1. \tag{3.19}$$

This shows that the constructed link pattern is an MCP for $mn \times mn$.

As a remark, by Theorem 3.5 we have

$$\lim_{L\to\infty} \frac{\gamma(L)}{\text{Total number of links in an } L \times L \text{ quad kirigami}} = \lim_{L\to\infty} \frac{L^2 - 1}{4L(L-1)} = \lim_{L\to\infty} \frac{L^2}{4L^2} = \frac{1}{4}. \tag{3.20}$$

This implies that any MCPs for an $L \times L$ quad kirigami require approximately $1/4$ of the total number of links as L is sufficiently large.

Combining Theorem 3.1 and Theorem 3.5, we have the following inequality for $L \geq 2$:

$$\delta(L) = \left\lceil \frac{3L^2 - 3}{2} \right\rceil > L^2 - 1 = \gamma(L). \tag{3.21}$$

This implies that there is no MRP which is also an MCP for any $L \geq 2$, and rigidifying a kirigami system requires more effort (links) compared to connecting it.

3.2.2 Enumeration of MCPs

Denote the number of MCPs in an $L \times L$ kirigami by $n_c(L)$. Using the Kirchhoff's matrix tree theorem, we can obtain the exact value of $n_c(L)$. Suppose we construct the Laplacian matrix of the $L \times L$ kirigami system by treating the L^2 tiles as vertices and all the $4L(L-1)$ possible links as edges. Then, by the Kirchhoff's matrix tree theorem, the total number of MCPs is

$$n_c(L) = \frac{1}{L^2} \prod \lambda_i, \tag{3.22}$$

where λ_i are the non-zero eigenvalues of the Laplacian matrix. Analogous to MRPs, it can be observed that the ratio of $n_c(L)$ to the number of all possible link patterns with exactly $L^2 - 1$ links becomes extremely small as L increases. The hierarchical construction provides us with a simple way for explicitly constructing MCPs for large L.

3.2.3 Controlling connectivity using MCPs

The MCPs are useful for controlling the connectivity of a kirigami system. Note that every link in an MCP decreases the number of connected components (NCC) by exactly 1. Therefore, by removing any k links from an MCP, we obtain a kirigami system with exactly $k + 1$ connected components.

3.3 Simultaneous control of rigidity and connectivity

More interestingly, we can achieve a certain level of control in both rigidity and connectivity using the MRPs and MCPs.

3.3.1 Simultaneous control of NCC and DoF using MRPs

Note that for any $L \times L$ MRP obtained by the hierarchical construction, adding or removing links that connect the rigid sub-blocks does not change the NCC or DoF within the sub-blocks themselves. Therefore, if d is a factor of L, it is possible for us to reverse the process of the hierarchical construction and only remove certain "key links" (i.e. those connecting the rigid sub-blocks) from the MRPs to control both the NCC and DoF precisely. More specifically, we can achieve NCC = $1, 2, \ldots, d^2$, and at the same time DoF can go from 3 to $3d^2$.

For instance, consider an MRP for an 18×18 kirigami system constructed by combining $\delta(2) = 5$ key links with the MRPs for the four sub-blocks of 9×9 tiles. Removing one of the five key links increases the DoF by 1 or 2 while preserving the NCC. Removing two of the five key links increases the DoF by 3 or 4, while the NCC either remains unchanged or increases by 1. As the process continues, all the five key links are eventually removed and the DoF of each 9×9 sub-block is 3, and hence the system is with NCC $= 4$ and DoF $= 3 \times 2^2 = 12$. To summarize, we have the following possible combinations of NCC and DoF:

- NCC $= 1$, DoF $= 3$ (the original MRP), 4, 5 (1 key link removed), 6 (2 key links removed);
- NCC $= 2$, DoF $= 7$ (2 key links removed), 8 (3 key links removed);

- NCC $=$ 3, DoF $=$ 9 (3 key links removed), 10 (4 key links removed);
- NCC $=$ 4, DoF $=$ 12 (5 key links removed).

In other words, by controlling only 5 links out of the $\partial(18) = 485$ links in an MRP for a 18×18 kirigami system, we can already achieve a large number of combinations of NCC and DoF. Moreover, note that the above process can be repeated for each of the sub-blocks by manipulating the key links in the MRPs. This shows that our hierarchical construction method for MRPs is capable of simultaneously controlling the NCC and DoF of kirigami systems.

It is also possible to change the DoF while keeping the NCC as small as possible using the MRPs. Note that in our hierarchical construction method, we always rigidify sub-blocks with odd size, in which each link changes the DoF by exactly 2. Hence, by removing any link from any rigid sub-block in an MRP, we can increase the DoF of the system by exactly 2 while keeping the NCC unchanged. This process can be continued until some tiles become disconnected. In other words, we can simultaneously achieve NCC $=$ 1 and DoF $=$ $2k + 3$ by removing k links, with k being sufficiently small.

3.3.2 Simultaneous control of NCC and DoF using MCPs

For any MCP of an $L \times L$ kirigami, clearly we have NCC $=$ 1. Also, since there are $L^2 - 1$ links in the MCP and each link decreases the DoF by 2, we have DoF $=$ $3L^2 - 2(L^2 - 1) = L^2 + 2$. By removing any link from the MCP, we can increase the DoF and the NCC by 1 and 2 respectively. Therefore, we achieve a kirigami system with NCC $= k + 1$ and DoF $= L^2 + 2k + 2$ by removing any k links.

Note that adding any link to an MCP will not change the NCC, while the DoF will decrease by 2. The process can be continued until some of the links become redundant. In other words, we can achieve NCC $= 1$ and DoF $= L^2 - 2k + 2$ simultaneously by adding k links to an MCP, where k is sufficiently small.

3.4 Extension to 2D kagome kirigami

While so far we have focused on the quad kirigami system only, the study of rigidity and connectivity can in fact be extended to kagome kirigami systems, for which the tiles are regular triangles instead of squares. Again, to simplify our discussion, we fix the geometry of the tiles and consider changing the links connecting them.

As shown in the previous sections, the MRPs and MCPs play an important role in controlling the rigidity and connectivity of kirigami. It turns out that Theorem 3.1 and Theorem 3.5 also hold for kagome kirigami systems.

To prove Theorem 3.1 for kagome kirigami systems, we start by constructing explicit examples of MRPs for 2×2, 3×3, 4×4, 5×5, 7×7, 3×5 and 5×3 (Figure 3.5**a**-**g**). Note that here we need two MRPs for 3×5 and 5×3, while only one MRP is needed for the case of quad kirigami. Using these MRPs, we can follow the proofs of Theorem 3.2, Theorem 3.3 and Theorem 3.4 to prove that the same lower bound is achievable for all L in the case of kagome kirigami (see Figure 3.5**h** for an illustration of the hierarchical construction for kagome kirigami).

Similarly, it is easy to see that Theorem 3.5 holds as each link reduces the NCC of a kagome

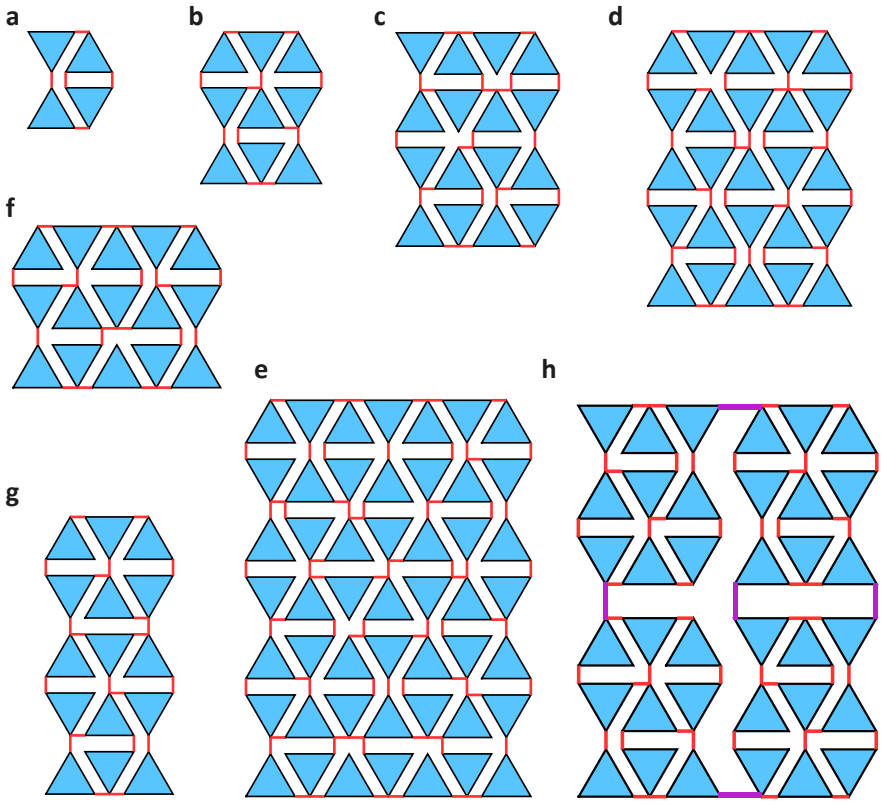

Figure 3.5: Minimum rigidifying link patterns (MRPs) for kagome kirigami. a-g, Explicit construction of MRPs for $L \times L$ kagome kirigami systems with $L = 2$ (**a**), $L = 3$ (**b**), $L = 4$ (**c**), $L = 5$ (**d**), $L = 7$ (**e**), and $M \times N$ kagome kirigami systems with $(M, N) = (3, 5)$ (**f**) and $(M, N) = (5, 3)$ (**g**). **h**, An illustration of the hierarchical construction for kagome kirigami. By decomposing a 6×6 kagome kirigami system into four large blocks of 3×3 triangles, we can rigidify each block using an MRP for $L = 3$ and the four large blocks using an MRP for $L = 2$. This results in an MRP for $L = 6$.

kirigami system by at most 1 (see Figure 3.6 for examples of MCPs for kagome kirigami).

Thus, the analysis of the rigidity and connectivity control in the previous sections can be extended to kagome kirigami.

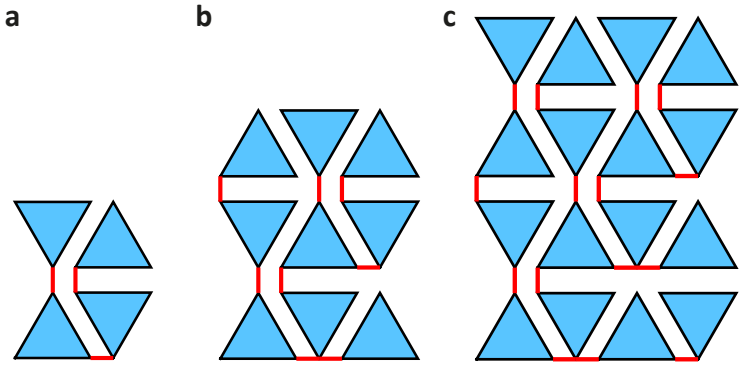

Figure 3.6: Minimum connecting link patterns (MCPs) for kagome kirigami. Starting from an MCP for $L = 2$ (**a**), we add one link at each edge on the top and the right boundary. This produces an MCP for $L = 3$ (**a**). Repeating the same procedure, we obtain an MCP for $L = 4$ (**c**).

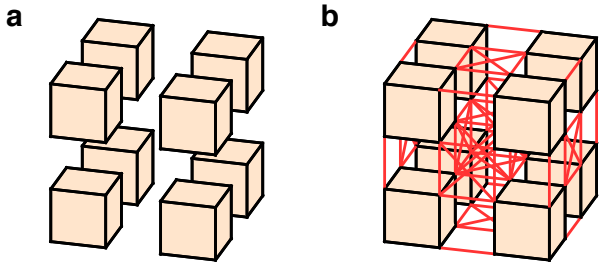

Figure 3.7: Topological control of 3D prismatic assembly as a linkage problem. a, We decompose a solid into $L \times M \times N$ cubes (here $L = M = N = 2$). **b**, All possible links for connecting neighboring cubes in a $2 \times 2 \times 2$ system.

3.5 Extension to 3D prismatic assemblies

We further extend the study of the topological control of kirigami to 3D. Given a 3D solid prismatic assembly formed by introducing vertical and horizontal cuts on a volumetric object (Figure 3.7), we study the minimum number of links needed for rigidifying or connecting the polyhedral elements, and how we can a target DoF or NCC using prescribed links. To simplify our discussion, we focus on 3D rectangular prismatic assemblies consisting of identical cubic elements.

3.5.1 MRPs for 3D rectangular prismatic assemblies

We start by extending the rigidity matrix rank computation to 3D. Recall that the rigidity matrix in the 2D case involves the edge length constraints and the diagonal length constraints for each square tile. In the 3D case, by the Dehn's rigidity theorem [34], any closed convex polyhedron with infinitesimally rigid faces is infinitesimally rigid. Therefore, for every cube with side length l, there are 12 edge length constraints in the form of

$$g_{\text{edge}}(\boldsymbol{v}_i, \boldsymbol{v}_j) = \|\boldsymbol{v}_i - \boldsymbol{v}_j\|^2 - l^2 = 0, \tag{3.23}$$

where $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ are adjacent vertices, and 6 *diagonal length constraints* for the 6 faces of the cube:

$$g_{\text{diagonal}}(\boldsymbol{v}_i, \boldsymbol{v}_j) = \|\boldsymbol{v}_i - \boldsymbol{v}_j\|^2 - 2l^2 = 0, \tag{3.24}$$

where $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ are opposite vertices in a face.

As for the link constraints, note that adding a link between two vertices $\boldsymbol{v}_i = (x_{3i-2}, x_{3i-1}, x_{3i})$ and $\boldsymbol{v}_j = (x_{3j-2}, x_{3j-1}, x_{3j})$ imposes three link constraints in the 3D case:

$$\begin{cases} g_{\text{link}_x}(\boldsymbol{v}_i, \boldsymbol{v}_j) = x_{3i-2} - x_{3j-2} = 0, \\ g_{\text{link}_y}(\boldsymbol{v}_i, \boldsymbol{v}_j) = x_{3i-1} - x_{3j-1} = 0, \\ g_{\text{link}_z}(\boldsymbol{v}_i, \boldsymbol{v}_j) = x_{3i} - x_{3j} = 0. \end{cases} \tag{3.25}$$

Again, the above length constraints and link constraints are not necessarily independent. Therefore,

we determine the DoF of the 3D rectangular prismatic assembly using the rigidity matrix A formed by the above constraints. Since there are 8 vertices per cube and 3 coordinates per vertex, the DoF is given by

$$d = 24LMN - \text{rank}(A). \tag{3.26}$$

Denote $\delta_{3D}(L, M, N)$ as the minimum number of links for rigidifying an $L \times M \times N$ 3D rectangular prismatic assembly. Since every cube has 3 translational DoF and 3 rotational DoF, the maximum and minimum values of the total DoF are respectively $d = 6LMN$ and $d = 6$. It follows that

$$6LMN - 3\delta_{3D}(L, M, N) \leq 6 \Rightarrow \delta_{3D}(L, M, N) \geq \frac{6LMN - 6}{3} = 2LMN - 2. \tag{3.27}$$

Denote a link patterns with exactly $2LMN - 2$ links which can rigidify an $L \times M \times N$ rectangular prismatic assembly as a *minimum rigidifying link pattern (MRP)*.

Here we show that the hierarchical construction also works for the 3D case. We first consider the case where $L = M = N$ and simplify the notation $\delta_{3D}(L, M, N)$ as $\delta_{3D}(L)$. Suppose the above lower bound holds for $L = l_1$ and $L = l_2$, i.e. $\delta_{3D}(l_1) = 2l_1^3 - 2$ and $\delta_{3D}(l_2) = 2l_2^3 - 2$. By decomposing an $l_1l_2 \times l_1l_2 \times l_1l_2$ rectangular prismatic assembly into $l_2 \times l_2 \times l_2$ large blocks with size $l_1 \times l_1 \times l_1$, we can rigidify each large block using an MRP for $l_1 \times l_1 \times l_1$ (with exactly $\delta_{3D}(l_1)$ links), and then connect and rigidity the large blocks using an MRP for $l_2 \times l_2 \times l_2$ (with exactly $\delta_{3D}(l_2)$ links). This results in a rigidifying link pattern for the $l_1l_2 \times l_1l_2 \times l_1l_2$ rectangular prismatic

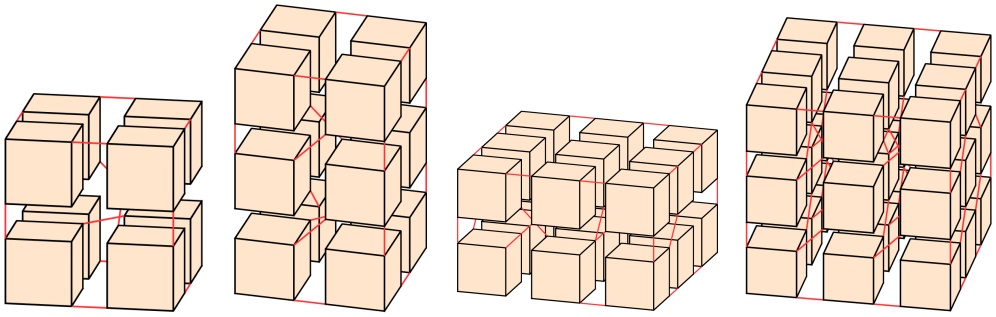

Figure 3.8: Minimum rigidifying link patterns (MRPs) for 3D rectangular prismatic assemblies. Left to right: MRPs for 3D rectangular prismatic assemblies with size $2 \times 2 \times 2$ (with exactly $2 \times 2^3 - 2 = 14$ links), $2 \times 2 \times 3$ (with exactly $2 \times 2 \times 2 \times 3 - 2 = 22$ links), $2 \times 3 \times 3$ (with exactly $2 \times 2 \times 3 \times 3 - 2 = 34$ links), and $3 \times 3 \times 3$ (with exactly $2 \times 3 \times 3 \times 3 - 2 = 52$ links).

assembly, with the total number of links being

$$l_2^3 \delta_{3D}(l_1) + \delta_{3D}(l_2) = l_2^3(2l_1^3 - 2) + (2l_2^3 - 2) = 2(l_1 l_2)^3 - 2. \tag{3.28}$$

This implies that $\delta_{3D}(l_1 l_2) = 2(l_1 l_2)^3 - 2$. Similar to the 2D case, we have the following theorem:

Theorem 3.6. *For all positive integer $L \geq 2$, we have*

$$\delta_{3D}(L) = 2L^3 - 2. \tag{3.29}$$

Proof. By explicitly constructing MRPs for $L \times M \times N = 2 \times 2 \times 2, 3 \times 3 \times 3, 2 \times 2 \times 3$, $2 \times 3 \times 3$ with exactly $2LMN - 2$ links (see Figure 3.8), we have shown that the statement is true for $L = 2, 3$.

For $L \geq 4$, we prove the statement by induction. Suppose it is true for all positive integers less

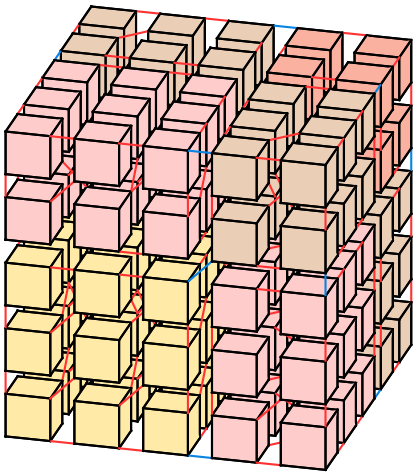

Figure 3.9: The hierarchical construction method for 3D rectangular prismatic assemblies. To construct an MRP for a $5 \times 5 \times 5$ 3D rectangular prismatic assembly, we decompose the assembly into blocks with size $2 \times 2 \times 2$, $2 \times 2 \times 3$, $2 \times 3 \times 3$, and $3 \times 3 \times 3$ (with different colors). We rigidify each block using an MRP shown in Figure 3.8 (the red links) and then the entire structure using an MRP for $2 \times 2 \times 2$ (the blue links).

than L. Note that for $L \geq 4$, there exists nonnegative integers a, b with $a + b \geq 2$ such that

$$L = 2a + 3b. \tag{3.30}$$

More specifically, we have the following cases:

- If $L \equiv 0 \pmod 3$, we can express L as $L = 2 \times 0 + 3 \times \frac{L}{3}$.
- If $L \equiv 1 \pmod 3$, we can express L as $L = 2 \times 2 + 3 \times \frac{L-4}{3}$.
- If $L \equiv 2 \pmod 3$, we can express L as $L = 2 \times 1 + 3 \times \frac{L-2}{3}$.

Now, we decompose the $L \times L \times L$ rectangular prismatic assembly into $(a+b) \times (a+b) \times (a+b)$ large blocks with size $2 \times 2 \times 2$, $2 \times 2 \times 3$, $2 \times 3 \times 3$, and $3 \times 3 \times 3$ (Figure 3.9). Since $2 \leq a + b < L$,

by the induction hypothesis and the hierarchical construction we have

$$\delta_{3D}(a+b) = 2(a+b)^3 - 2. \tag{3.31}$$

Therefore, by rigidifying the large blocks using MRPs and then the entire structure using an MRP for $(a+b) \times (a+b) \times (a+b)$, we obtain a rigidifying link pattern for the $L \times L \times L$ rectangular prismatic assembly, with the total number of links

$$\begin{aligned}
&a^3\delta_{3D}(2) + b^3\delta_{3D}(3) + 3a^2b\delta_{3D}(2,2,3) + 3ab^2\delta_{3D}(2,3,3) + \delta_{3D}(a+b) \\
=&14a^3 + 52b^3 + 66a^2b + 102ab^2 + 2(a+b)^3 - 2 \\
=&16a^3 + 54b^3 + 72a^2b + 108ab^2 - 2 \\
=&2(2a+3b)^3 - 2 \\
=&2L^3 - 2.
\end{aligned} \tag{3.32}$$

This implies that $\delta_{3D}(L) = 2L^3 - 2$. The result then follows from induction. ■

We further prove the following result:

Theorem 3.7. *For infinitely many positive integers L, M, N which are not all identical, we have*

$$\delta_{3D}(L,M,N) = 2LMN - 2. \tag{3.33}$$

Proof. Take any set of nonnegative integers $a_l, b_l, a_m, b_m, a_n, b_n$ such that $a_l + b_l = a_m + b_m = a_n + b_n \geq 2$, and $L = 2a_l + 3b_l, M = 2a_m + 3b_m, N = 2a_n + 3b_n$ are not all identical (e.g.

$(a_l, b_l, a_m, b_m, a_n, b_n) = (1, 6, 2, 5, 3, 4)$, with $(L, M, N) = (20, 19, 18)$). We decompose the $L \times M \times N$ 3D rectangular prismatic assembly into $(a_l + b_l) \times (a_m + b_m) \times (a_n + b_n)$ sub-blocks of size $2 \times 2 \times 2$, $2 \times 2 \times 3$, $2 \times 3 \times 3$, and $3 \times 3 \times 3$. Since $a_l + b_l = a_m + b_m = a_n + b_n$, we follow the proof of Theorem 3.6 and rigidify each sub-block and subsequently the entire structure using MRPs, forming a rigidifying link pattern with the total number of links being

$$\begin{aligned}
&a_l a_m a_n \delta_{3D}(2) + b_l b_m b_n \delta_{3D}(3) + (a_l a_m b_n + a_l a_n b_m + a_m a_n b_l)\delta_{3D}(2,2,3) \\
&+ (a_l b_m b_n + a_m b_l b_n + a_n b_l b_m)\delta_{3D}(2,3,3) + \delta_{3D}(a_l + b_l) \\
=&14 a_l a_m a_n + 52 b_l b_m b_n + 22(a_l a_m b_n + a_l a_n b_m + a_m a_n b_l) + 34(a_l b_m b_n + a_m b_l b_n + a_n b_l b_m) \\
&+ 2(a_l + b_l)(a_m + b_m)(a_n + b_n) - 2 \\
=&2(2a_l + 3b_l)(2a_m + 3b_m)(2a_n + 3b_n) - 2 = 2LMN - 2.
\end{aligned} \tag{3.34}$$

This implies that $\delta_{3D}(L, M, N) = 2LMN - 2$. ■

It is noteworthy that the above technique can be further exploited for constructing more MRPs. For instance, using $2 + 0 = 1 + 1 = 0 + 2 = 2$, we first construct an MRP for a $4 \times 5 \times 6$ 3D rectangular prismatic assembly (Figure 3.10**a**). Then, for any nonnegative integers $a_l, b_l, a_m, b_m, a_n, b_n$ with $a_l + b_l = 4, a_m + b_m = 5, a_n + b_n = 6$, the same technique can be used for constructing an MRP for a $(2a_l + 3b_l) \times (2a_m + 3b_m) \times (2a_n + 3b_n)$ rectangular prismatic assembly (Figure 3.10**b**).

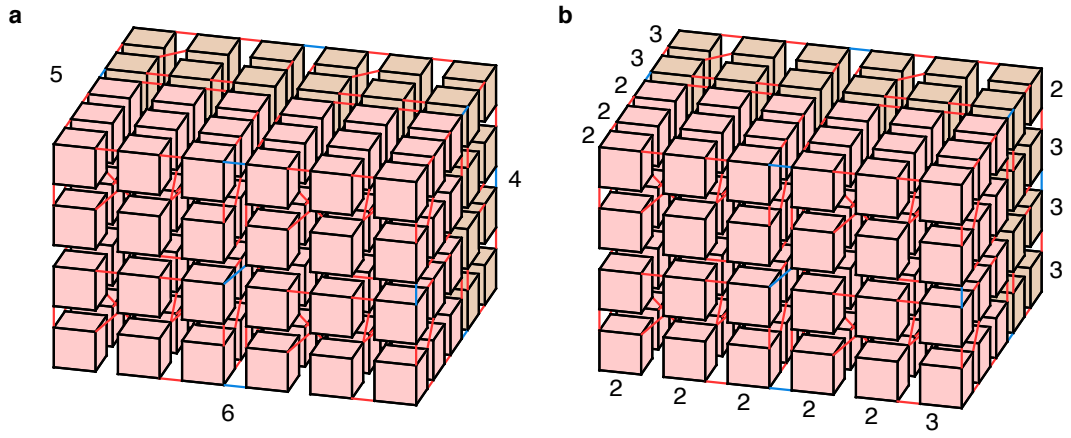

Figure 3.10: More examples of minimum rigidifying link patterns (MRPs) for 3D rectangular prismatic assemblies obtained using the hierarchical construction. a, An MRP for $4 \times 5 \times 6$ constructed using four MRPs for $3 \times 3 \times 2$ (the pink cubes and the associated red links) and four MRPs for $3 \times 2 \times 2$ (the pale brown cubes and the associated red links), together with an MRP for $2 \times 2 \times 2$ (the blue links). **b**, An MRP for $13 \times 12 \times 11$ constructed using the result in **a**, with each cube replaced with a system with size $2 \times 2 \times 2$, $2 \times 2 \times 3$, $2 \times 3 \times 2$, or $3 \times 3 \times 3$.

3.5.2 MCPs for 3D rectangular prismatic assemblies

The study of MCPs can be easily extended to the 3D case. Denote $\gamma_{3D}(L, M, N)$ as the minimum number of links needed for connecting a $L \times M \times N$ 3D rectangular prismatic assembly. Since each link reduces the NCC by at most 1, we have

$$\gamma_{3D}(L, M, N) = LMN - 1. \tag{3.35}$$

MCPs can be explicitly constructed using an approach analogous to the 2D construction (Figure 3.11). Using MCPs for small systems, we can hierarchically construct MCPs for larger systems.

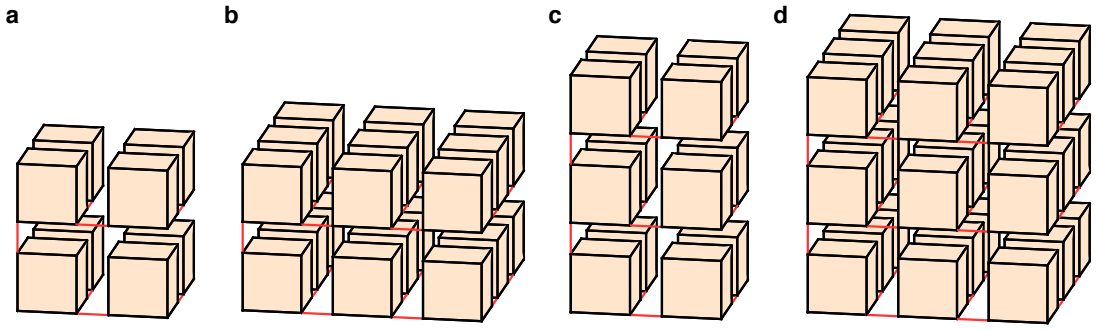

Figure 3.11: Minimum connecting link patterns (MCPs) for 3D rectangular prismatic assemblies. Left to right: MCPs for $2 \times 2 \times 2, 2 \times 3 \times 3, 2 \times 2 \times 3$, and $3 \times 3 \times 3$.

Interestingly, $\delta_{3D}(L)$ and $\gamma_{3D}(L)$ are related by a simple formula:

$$\delta_{3D}(L) = 2L^3 - 2 = 2(L^3 - 1) = 2\gamma_{3D}(L). \tag{3.36}$$

In other words, the minimum number of links for rigidifying any $L \times L \times L$ 3D rectangular prismatic assembly is exactly twice of the minimum number of links for connecting it.

3.5.3 Controlling rigidity and connectivity using MRPs and MCPs

Similar to the 2D case, we can easily control the rigidity and connectivity of 3D rectangular prismatic assemblies by manipulating the links in the MRPs and MCPs.

For instance, note that for any MRP of an $L \times L \times L$ 3D rectangular prismatic assembly, we have DoF = 6 and NCC = 1. Since every link in the MRP is non-redundant, we can achieve DoF = $6 + 3k$ by removing any k links from it. Note that NCC will remain unchanged for sufficiently small L. Similarly, for any MCP of an $L \times L \times L$ 3D cubic kirigami system, we have NCC = 1 and DoF

$= 6L^3 - 3\gamma_{3D}(L) = 3L^3 + 3$. By removing any k links from it, we achieve a rectangular prismatic assembly with NCC = $k + 1$ and DoF $3L^3 + 3k + 3$. The other approaches for simultaneously controlling the DoF and NCC discussed in the 2D case can also be suitably extended for the 3D case.

3.5.4 Triangular prismatic assemblies

Our topological control framework does not only work for rectangular prismatic assemblies but also structural assemblies formed by other space-filling prisms, such as the triangular prisms.

More specifically, for triangular prismatic assemblies, we again consider the length and link constraints as discussed above. This time, the DoF of an $L \times M \times N$ triangular prismatic assembly is given by

$$d = 18LMN - \text{rank}(A), \tag{3.37}$$

where A is the rigidity matrix. Since each prism has six vertices and each of them has three coordinates, we have the factor 18 (instead of 24) for the first term.

Following the approach for rectangular prismatic assemblies, we construct several MRPS as the building blocks for the hierarchical construction of MRPs for larger triangular prismatic assemblies (Figure 3.12). Using these MRPs, we can show that Theorem 3.6 and Theorem 3.7 hold for triangular prismatic assemblies. The connectivity control can be achieved similarly.

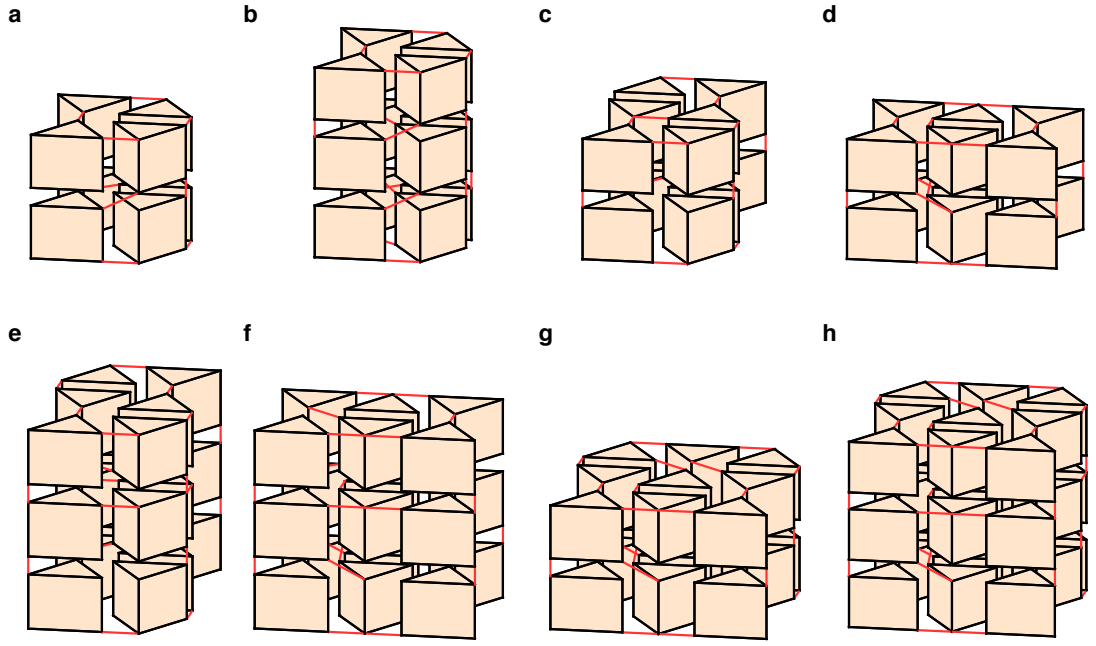

Figure 3.12: Minimum rigidifying link patterns (MRPs) for triangular prismatic assemblies. **a**, An MRP for $2 \times 2 \times 2$, with $2 \times 2 \times 2 \times 2 - 2 = 14$ links. **b**, An MRP for $2 \times 2 \times 3$, with $2 \times 2 \times 2 \times 3 - 2 = 22$ links. **c**, An MRP for $2 \times 3 \times 2$, with $2 \times 2 \times 3 \times 2 - 2 = 22$ links. **d**, An MRP for $3 \times 2 \times 2$, with $2 \times 3 \times 2 \times 2 - 2 = 22$ links. **e**, An MRP for $2 \times 3 \times 3$, with $2 \times 2 \times 3 \times 3 - 2 = 34$ links. **f**, An MRP for $3 \times 2 \times 3$, with $2 \times 3 \times 2 \times 3 - 2 = 34$ links. **g**, An MRP for $3 \times 3 \times 2$, with $2 \times 3 \times 3 \times 2 - 2 = 34$ links. **h**, An MRP for $3 \times 3 \times 3$, with $2 \times 3 \times 3 \times 3 - 2 = 52$ links.

3.6 Discussion

We have explored the control of rigidity and connectivity of changing the cut topology of kirigami. This complements our study of the geometric control of kirigami in the previous two chapters. More broadly, our studies provide guidelines on the geometric and topological consideration for designing mechanical metamaterials.

The essence of mathematics lies entirely in its freedom.

Georg Cantor

4

Additive origami and kirigami

Motivated by the recent advances in additive manufacturing, we consider an alternative approach for metamaterial design in this chapter. In the previous three chapters, we have focused on the geometric and topological control of kirigami in a global perspective. While we can effectively achieve prescribed target shapes, rigidity and connectivity using our proposed methods,

the global design problems involve the consideration of the constraints in all nodes in a kirigami pattern, making the computation time-consuming.

Here, we consider an additive design approach that creates a metamaterial structure in a layer-by-layer manner. By identifying the constraints in lengths and angles at the growth front of a partially built structure, we can simplify the global design problem as a series of local design problems involving the nodes at the growth front only.

4.1 Additive origami design

We start by considering the problem of *additive origami design*, i.e. the design of origami structures in an additive manner. Suppose we are given a quad origami surface, such as the well-known Miura-ori pattern as shown in Figure 4.1**a**. Our goal is to grow the origami surface by suitably adding new quad strips along the boundary of the surface.

Note that one has to ensure that every quad in a newly added quad strip is compatible with the existing pattern. This requires the consideration of all angles at the growth front. Nevertheless, we can largely simplify the problem using the following result:

Theorem 4.1. *The space of new interior edge directions along the entire growth front in a quad origami is one-dimensional, i.e. uniquely determined by a single angle.*

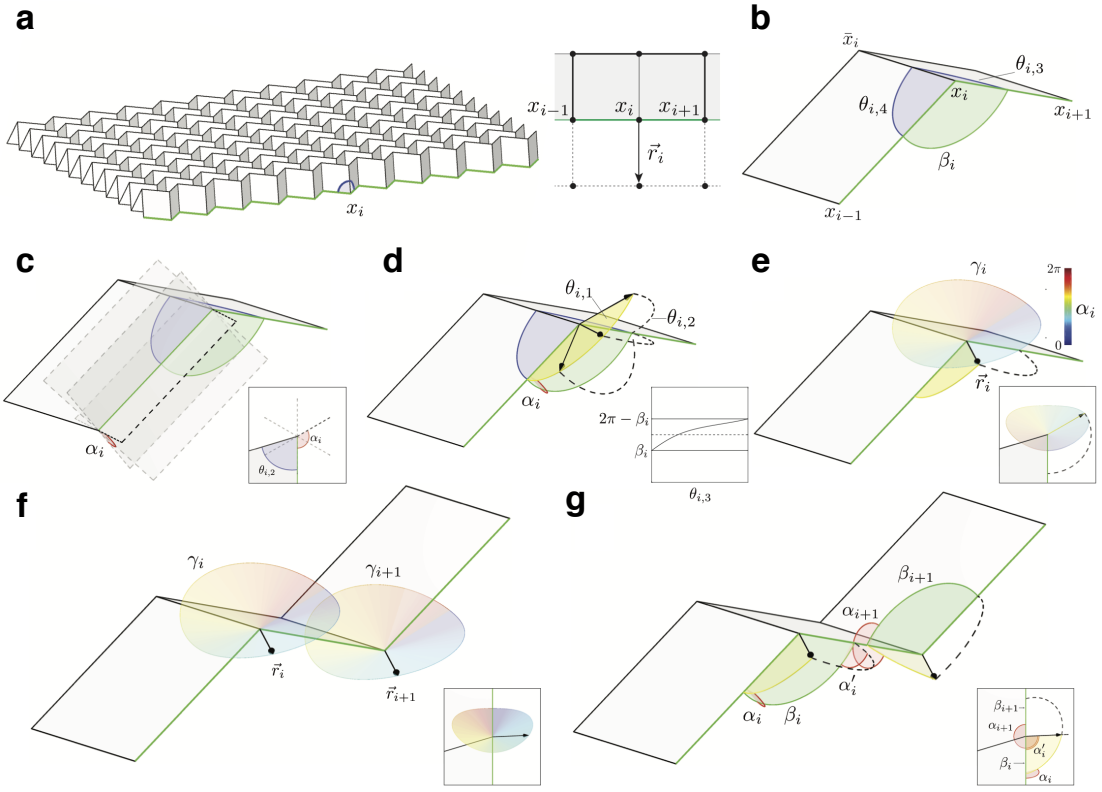

Figure 4.1: Additive origami design. **a**-**b**, An existing quad origami surface with a growth front (green), and a zoom-in of a single boundary vertex x_i. **c**, An illustration of the flap angle α_i (red), which sweeps from the β_i face counter-clockwise about $\vec{e}_i$. Inset view is along $\vec{e}_i$. **d**, The new design angle $\theta_{i,1}$ sweeps through the plane determined by α_i, giving the possible new edge directions $\vec{r}_i$. The other new design angle $\theta_{i,2}$ (dashed), between $\vec{r}_i$ and $\vec{e}_{i+1}$, is determined by $\theta_{i,1}$. **e**, An illustration of the choices of $\theta_{i,1}$ that satisfy the developability constraint. **f**, Note that two adjacent growth directions $\vec{r}_i$ and $\vec{r}_{i+1}$ must be coplanar, and hence $\vec{r}_{i+1}$ is determined by the intersection of this plane and γ_{i+1}. **g**, The secondary flap angle α'_i at x_i, which sweeps from the β_i face counter-clockwise about $\vec{e}_{i+1}$, is determined by α_i. Consequently, the flap angle α_{i+1} at x_{i+1} is also determined by α_i.

4.1.1 Single vertex construction

To prove Theorem 4.1, we start by considering the problem at a single boundary vertex x_i along the growth front (Figure 4.1**a**), with the two adjacent growth front vertices denoted by x_{i-1}, x_{i+1}, the two boundary design angles incident to x_i in the existing surface denoted by $\theta_{i,3}$ and $\theta_{i,4}$, and the

angle in space at x_i along the growth front denoted by $\beta_i = \angle\{-\vec{e}_i, \vec{e}_{i+1}\} \in (0, \pi)$ (Figure 4.1**b**), where $\vec{e}_i = x_i - x_{i-1}$ and $\vec{e}_{i+1} = x_{i+1} - x_i$.

To obtain a new edge direction vector $\vec{r}_i$ that gives the direction of an interior edge x_i, x_i' in the augmented quad origami surface, let $\alpha_i \in [0, 2\pi)$ be the left-hand oriented flap angle about $\vec{e}_i$ from the β_i plane to the plane of the new quad containing $\vec{r}_i$ and $\vec{e}_i$ (Figure 4.1**c**). Note that the single vertex origami at x_i should satisfy the local angle sum developability constraint

$$\sum_{j=1}^{4} \theta_{i,j} = 2\pi, \tag{4.1}$$

where $\theta_{i,1} = \cos^{-1}(-\vec{r}_i \cdot \vec{e}_i)$ and $\theta_{i,2} = \cos^{-1}(\vec{r}_i \cdot \vec{e}_{i+1})$ are two new design angles implied by $\vec{r}_i$ (Figure 4.1**d**). Since $\theta_{i,1}$, $\theta_{i,2}$ and β_i form a spherical triangle with α_i being an interior spherical angle opposite $\theta_{i,2}$, it follows from the spherical law of cosines that

$$\cos\theta_{i,2} = \cos\theta_{i,1}\cos\beta_i + \sin\theta_{i,1}\sin\beta_i\cos\alpha_i. \tag{4.2}$$

Solving Eq. (4.1) and Eq. (4.2) for $\theta_{i,1}$ and $\theta_{i,2}$, we are then able to create a single vertex origami at x_i:

$$\begin{cases} \theta_{i,1} = \tan^{-1}\frac{\cos(\theta_{i,3}+\theta_{i,4})-\cos\beta_i}{\sin(\theta_{i,3}+\theta_{i,4})+\sin\beta_i\cos\alpha_i}, \\ \theta_{i,2} = 2\pi - \theta_{i,1} - \theta_{i,3} - \theta_{i,4}. \end{cases} \tag{4.3}$$

One can further prove that the solutions $\theta_{i,1}, \theta_{i,2}$ to Eq. (4.1) and Eq. (4.2) exist and are unique for any given $\theta_{i,3}, \theta_{i,4}, \beta_i$ (angles intrinsic to the existing origami) and α_i (the flap angle), modulo a finite

number of singular configurations. The edge direction $\vec{r}_i$ can then be obtained from the two new design angles $\theta_{i,1}, \theta_{i,2}$ (Figure 4.1**e**), which are uniquely determined by α_i.

4.1.2 Adjacent vertices

After understanding the geometry of a single vertex origami, we proceed to consider the relationship between adjacent vertices at the growth front. More specifically, we show that the new edge directions $\vec{r}_{i+1}, \vec{r}_{i-1}$ at the adjacent vertices x_{i+1}, x_{i-1} are also uniquely determined by α_i.

Without loss of generality, we consider the new edge direction $\vec{r}_{i+1}$ (Figure 4.1**f**) and denote α'_i as the left-hand oriented angle about $\vec{e}_{i+1}$ from the β_i plane to the plane of the new quad containing $\theta_{i,2}$. Applying the spherical laws of sines and cosines on the spherical triangle formed by $\theta_{i,1}$, $\theta_{i,2}$ and β_i, we have

$$\begin{cases} \sin\alpha'_i = \frac{\sin\theta_{i,1}(\alpha_i)}{\sin\theta_{i,2}(\alpha_i)}\sin\alpha_i, \\ \cos\alpha'_i = \frac{\cos\theta_{i,1}(\alpha_i) - \cos\theta_{i,2}(\alpha_i)\cos\beta_i}{\sin\theta_{i,2}(\alpha_i)\sin\beta_i}. \end{cases} \tag{4.4}$$

The two equations above yield a unique solution $\alpha'_i \in [0, 2\pi)$. Since $\theta_{i,1}$ and $\theta_{i,2}$ are functions of α_i, α'_i is also a function of α_i. Now, note that α'_i and α_{i+1} are measured about a common axis. Hence, they are thus related by a shift of the left-hand oriented angle φ_i from the β_i face to the β_{i+1} face. This gives the flap angle transfer function $g_i : [0, 2\pi) \to [0, 2\pi)$:

$$\alpha_{i+1} = g_i(\alpha_i) = \mathrm{mod}(\alpha'_i(\alpha_i) - \varphi_i, 2\pi) \tag{4.5}$$

as measured left-hand oriented about $\vec{e}_{i+1}$ starting at the β_i plane (Figure 4.1**g**). It is easy to see that

g is bijective. Therefore, the new edge direction $\vec{r}_{i+1}$ is uniquely determined by α_i. Similarly, $\vec{r}_{i-1}$ is also uniquely determined by α_i.

4.1.3 The entire growth front

It remains to establish a bijection between the flap angles α_i and α_j at arbitrary i, j with $i < j$. Consider the following composition of the transfer functions:

$$\alpha_j = g_j(g_{j-1}(g_{j-2}(\cdots g_i(\alpha_i)))). \tag{4.6}$$

Since each of the transfer functions $g_i, g_{i+1}, \ldots, g_j$ is bijective, the composition of them is also bijective. It follows that all new interior edge directions along the entire growth front are parameterized by a single angle α_i.

With the results in the three subsections above, we have completed the proof of Theorem 4.1.

4.1.4 Creating origami structures using the additive design

To apply Theorem 4.1 for creating an origami surface additively, we start by selecting a growth front at an existing origami surface (Figure 4.2**a**). Suppose the growth front consists of $m + 1$ vertices. By choosing an arbitrary flap angle along one of the edges at the growth front, we determine the orientation of one plane in the new strip. By Theorem 4.1, this choice propagates along the entire growth front, thereby uniquely determining the angles of all m planes in the new strip, except for the two boundary design angles (Figure 4.2**b**). The lengths of the $m + 1$ new edges can then be set

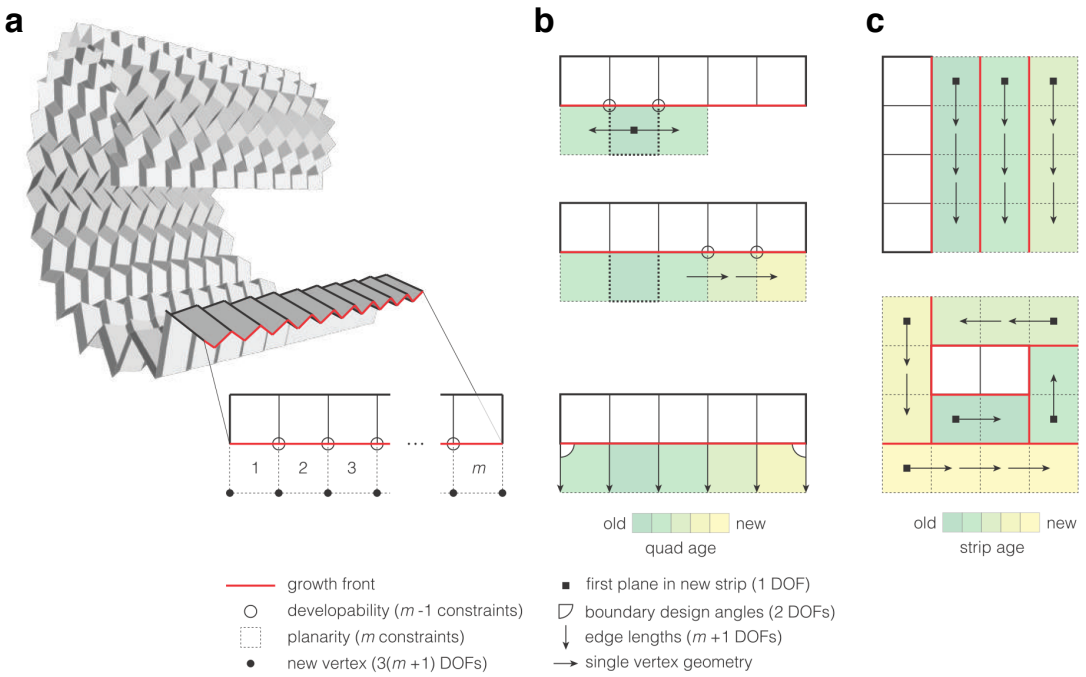

Figure 4.2: Construction of origami surfaces via additive origami design. a, An existing origami surface with a growth front consisting of $m + 1$ vertices (red). **b**, Starting from any edge along the growth front, we can choose the flap angle for the new quad. The choice of the flap angle will then propagate along the entire growth front, thereby uniquely determining all angles in the new strip (except for the two boundary ones). **c**, Any boundary of the origami surface can be chosen as the growth front to extend the surface.

freely, with the condition that none of them intersects with each other. This determines the geometry of the newly added strip. It is noteworthy that the process can be continued with any boundary of the origami surface being the growth front (Figure 4.2**c**), which makes the design largely flexible.

To demonstrate the effectiveness of our additive design approach, we apply it for creating origami structures that approximate any prescribed surface in $\mathbb{R}^3$. We first construct an upper surface and a lower surface of the prescribed surface[40]. Then, for every newly added strip, the angles and the edge lengths are determined by enforcing adjacent vertices to lie on the upper and lower surfaces

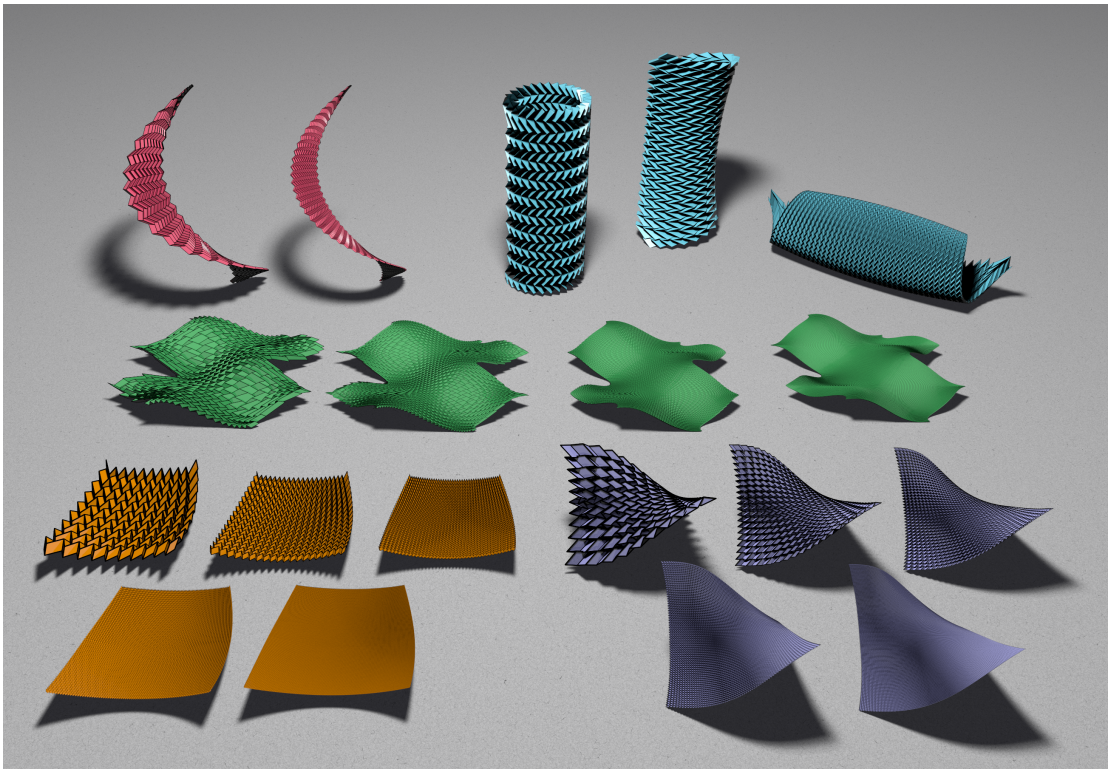

Figure 4.3: A gallery of surface fitting results obtained by our additive origami design. We apply our method to create origami surfaces that approximate a helicoid at two different resolutions (top left), cylinders with different Gaussian curvatures (zero/negative/positive as shown in top right), a landscape surface with mixed curvature at four different resolutions (the middle row), a paraboloid at five different resolutions (bottom left), and a hyperbolic paraboloid at five different resolutions (bottom right).

alternately. As shown in Figure 4.3, our additive design approach allows for the construction of origami surfaces that approximate a large variety of surfaces with different curvature properties. Furthermore, unlike the global inverse design approach[40], the additive design approach only requires solving a constrained optimization problem for each newly added strip, which is more computationally feasible.

Besides the generalized Miura-ori patterns, our method is capable of producing curved fold models with the global mountain-valley patterns set to alternate in the direction transverse that of

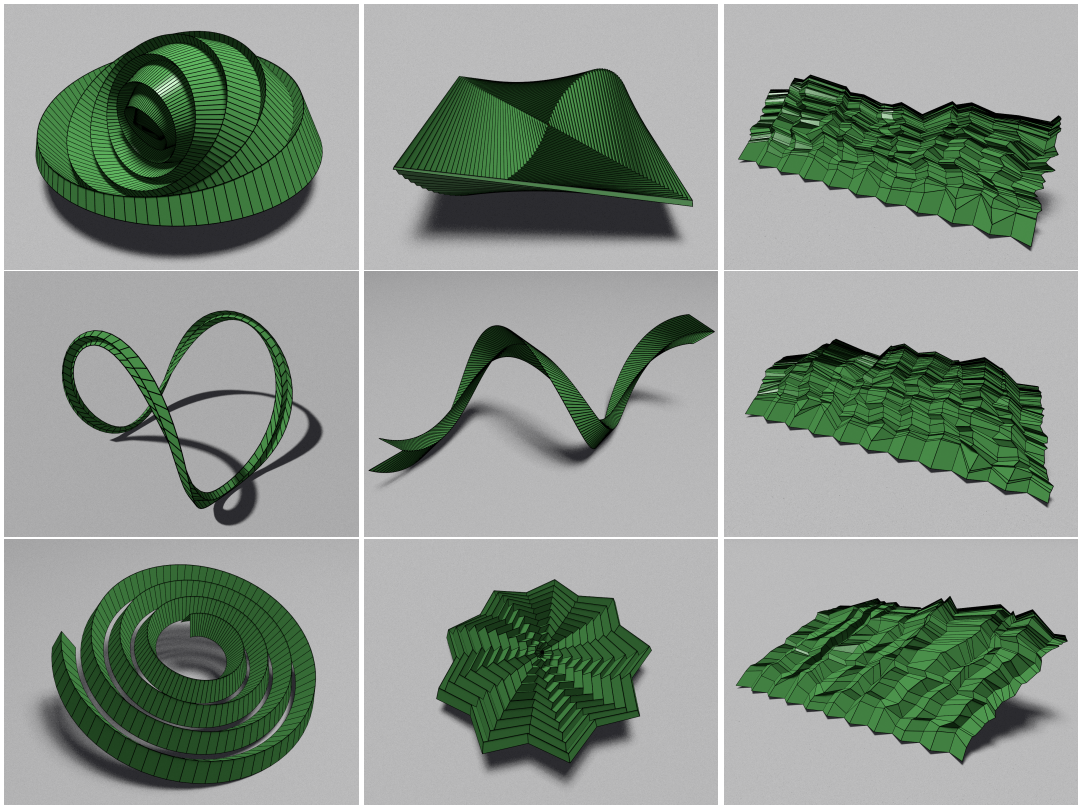

Figure 4.4: More origami models produced by our additive design approach. Our method allows for the design of curved fold models (left and middle) as well as disordered crumpled sheet models (right).

the curved folds, as well as disordered crumpled sheet models without any systematic mountain-valley patterns (Figure 4.4). These examples again demonstrate the effectiveness of our additive approach for origami design.

4.2 Additive kirigami design

The idea of additive design is also applicable to kirigami. Below, we outline two possible approaches for the additive design of kirigami.

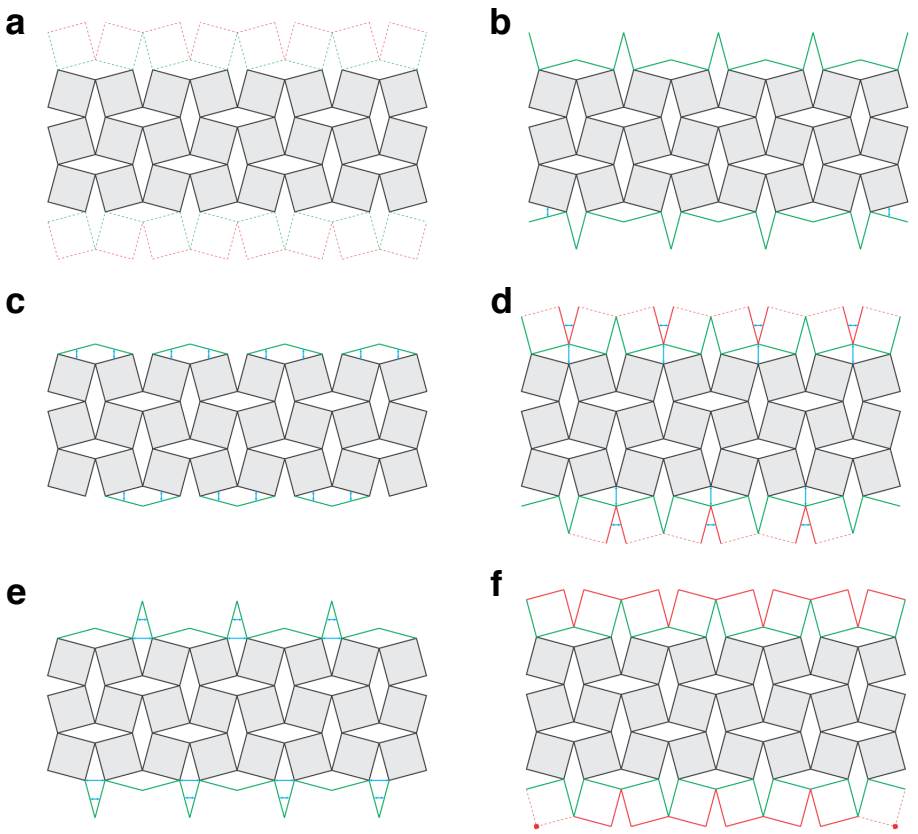

Figure 4.5: Forward additive kirigami design. **a**, To augment a kirigami structure by adding two new layers, we first determine all edges in the new quads incident to the existing pattern (green) and subsequently all edges not incident to the existing structure (red). **b**, Note that some of the green edges are uniquely determined by their length correspondence (blue) with edges in the existing pattern (DoF = 0). **c**, The other green edges at the interior of the new strip are also uniquely determined by the angle sum constraint at the corresponding four-bar linkage (DoF = 0). **d**, The positions of the two boundary vertices at the top strip can be chosen freely (DoF = 2×2), while the those at the bottom strip are determined by the choice of the two deployment angles (DoF = 2). **e**, For each rigid wedge (solid red), 1 length and 1 orientation can be chosen (DoF = $2 \times \#$wedges). The angle between the edges in each wedge is uniquely determined by the angle sum constraint, and the remaining dashed red edges are also uniquely determined by the above choices. **f**, The top strip has been completed determined, while the bottom strip has two boundary vertices that can be placed freely (DoF = 2×2).

4.2.1 Forward additive design

Analogous to the study of additive origami design, we explore the flexibility of controlling the geometry of the newly added strip. Consider adding two layers of quads to an existing quad

kirigami system (Figure 4.5**a**). The process can be decomposed into two steps: First, we determine the edges in the new quads incident to the existing pattern. Then, we determine the edges not incident to the existing pattern. We note that all edges incident to the existing pattern are either uniquely determined by their length correspondence with edges in the existing pattern (Figure 4.5**b**) or by the angle sum constraints (Figure 4.5**c**), except for the boundary points (Figure 4.5**d**). For the edges not incident to the existing pattern, note that we have local control of their lengths and orientations via the design of the wedges (Figure 4.5**e**). Finally, we have full control of certain boundary points (Figure 4.5**f**).

In other words, if the growth front consists of $2k$ quads, then the DoF in determining the geometry of the newly added strip is either $2 \times 2 + 2 \times k$ (for the top strip in Figure 4.5) or $2 + 2 \times (k-1) + 2 \times 2$ (for the bottom strip in Figure 4.5). Note that in both cases, the total DoF equals $4 + 2k$. We have thereby characterized the full design space for kirigami design. By changing the $4 + 2k$ variables in each newly added strip, we can easily create large-scale kirigami patterns.

4.2.2 Inverse additive design

One can also perform the inverse design of kirigami in Chapter 1 additively. As illustrated in Figure 4.6, given a target shape $S \subset \mathbb{R}^2$, we consider a series of nested shapes $\{S_i\}_{i=1}^N$ where $S_1 \subset S_2 \subset \cdots \subset S_N = S$.

We first solve a simple constrained optimization problem to approximate S_1 using a few kirigami tiles. Then, for $i = 2, 3, \ldots, N$, we can subsequently approximate S_i by keeping the result for S_{i-1} and solving the constrained optimization problem for the tiles inside $S_i \setminus S_{i-1}$. Here, the constraints

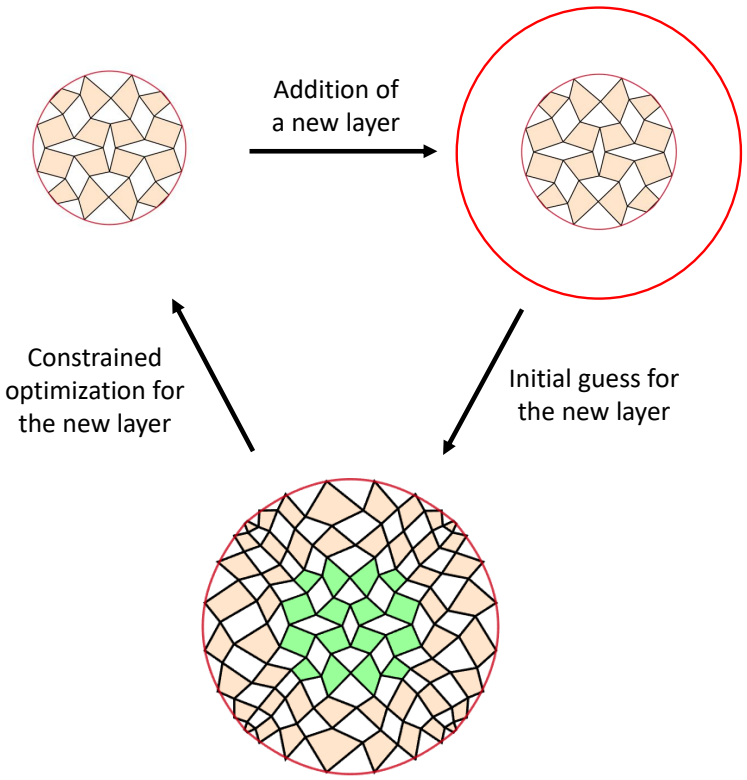

Figure 4.6: Inverse additive kirigami design. We start by solving the inverse design problem for a small pattern. Then, we add a new layer at the boundary of the existing kirigami structure and construct an initial guess of the geometry of the new layer in the deployed space. Keeping the existing optimization result (green) fixed, we solve a new constrained optimization problem on the exterior region and obtain a valid kirigami structure. This process can be continued until a target resolution is reached.

and objective function are the same as those introduced in Chapter 1, while the dimension of the search space is largely reduced as the tiles inside S_{i-1} are fixed. As for the initial guess, the quasi-conformal mapping method[92] can be used for enforcing all deformed tiles to lie on $S_i \setminus S_{i-1}$.

It is natural to ask about the limit of the inverse additive design approach, i.e. to what extent we can control the geometry of the newly added tiles in this approach. Consider adding a new layer with width 1 to an existing $n \times n$ quad kirigami structure (Figure 4.7, left). The total DoF in the

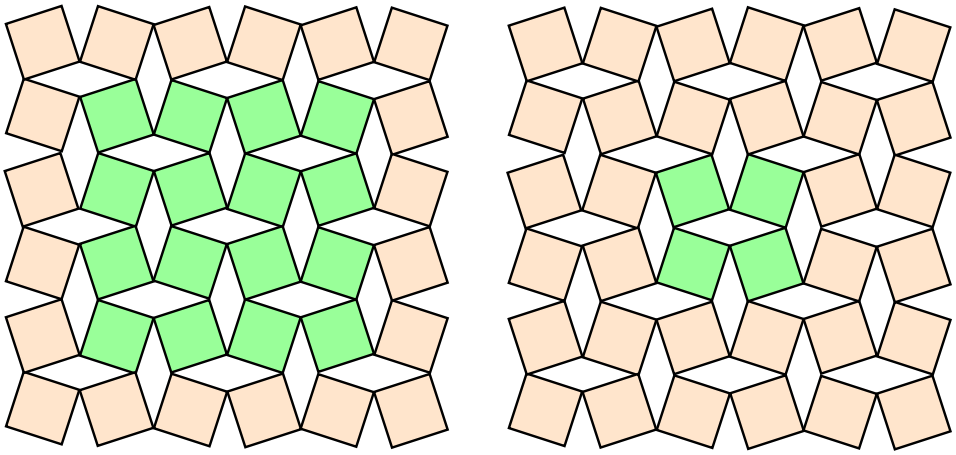

Figure 4.7: The limit of inverse additive kirigami design. Here we consider adding a new layer with width 1 (left) and 2 (right) to an existing kirigami structure. Note that the total number of variables in the new layer determines the possibility of imposing shape constraints in the constrained optimization framework.

newly added layer is $2(2n+2)+4+8 = 4n+16$. Here, the first term is from the $2n+2$ 'V' shapes at the boundary, the second term is from the four deployment angles at the four corners of the existing structure, and the third term is from the four corner vertices of the new layer. Now, to enforce that all boundary vertices of the augmented kirigami structure to lie on a target shape, we have in total $2(2n+2)+4 = 4n+8$ constraints. Since $4n+16 > 4n+8$, it is possible to impose such boundary shape matching constraints to control the shape of the augmented structure.

However, in case we would like to enforce the contracted configuration of the augmented structure to be a rectangle, we have $2(2n+2)+4 = 4n+8$ additional rectangular boundary constraints. As $4n+16 < 4n+8+4n+8$, it is in general impossible for us to achieve this requirement.

Nevertheless, if we consider adding a new layer with width 2 to an existing $n \times n$ quad kirigami structure (Figure 4.7, right), a similar calculation suggests that the total DoF in the newly added

layer is $4n + 16 + 4(n + 2) + 16 = 8n + 40$. The number of the boundary shape matching constraints is $4(n + 2) + 8 = 4n + 16$, and the number of the rectangular boundary constraints is also $4(n + 2) + 8 = 4n + 16$. Since $8n + 40 > 4n + 16 + 4n + 16$, adding a new layer with width 2 is sufficient for enabling the control of both the contracted and the deployed boundary shapes.

4.3 Discussion

The additive design framework is advantageous over the global design approaches as it allows for the flexibility of controlling the local geometry of the patterns without going through the global optimization process. The reduction of computational complexity makes the additive design a promising way for creating mechanical metamaterials at arbitrary scale.

Cell and tissue, shell and bone, leaf and flower, are so many portions of matter, and it is in obedience to the laws of physics that their particles have been moved, moulded and conformed. They are no exceptions to the rule that God always geometrizes.

D'Arcy Thompson

5

Insect wing morphometry

Biological form both constrains and enables biological function. To quantify, compare and classify biological shapes, morphometric tools are necessary. In geometric morphometrics, landmark coordinates are widely used for shape quantification [149]. Note that a planar shape may be quantified by its boundary, which determines its overall geometry.

Additionally, specific features in its interior may be used as landmarks to capture its structural features. It is therefore important to develop a method that takes both the overall shape and the interior landmarks into consideration for quantifying shape variation.

In this work, we develop a landmark-matching, curvature-guided Teichmüller mapping for planar morphometrics using quasi-conformal theory [51] and functional data analysis [127]. Our method allows for the exact matching of boundaries as well as the prescribed landmarks, thereby overcoming all the above-mentioned problems. We deploy our method for quantifying the shape variation of insect wings across developmental and evolutionary time scales.

5.1 Landmark-matching, curvature guided Teichmüller maps

Consider two simply-connected closed regions $S_1, S_2 \subset \mathbb{C}$ as two planar shapes that we want to compare (Figure 5.1**a**). Let $\{l_k^{int_1}\}_{k=1}^m$ and $\{l_k^{int_2}\}_{k=1}^m$ be two sets of landmarks at the interior of S_1 and S_2 respectively, and $\{l_k^{bdy_1}\}_{k=1}^n$ and $\{l_k^{bdy_2}\}_{k=1}^n$ be two sets of landmarks on the boundaries ∂S_1 and ∂S_2. Our goal is to find a bijective map $f : S_1 \to S_2$ satisfying

$$f(l_k^{int_1}) = l_k^{int_2}, k = 1, 2, ..., m, \tag{5.1}$$

and

$$f(l_k^{bdy_1}) = l_k^{bdy_2}, k = 1, 2, ..., n, \tag{5.2}$$

with the shape difference between S_1 and S_2 captured by f.

5.1.1 Boundary matching based on curvature

To compute the map f, we start by determining the boundary correspondence $\phi : \partial S_1 \to \partial S_2$. A natural way to match the boundaries of S_1 and S_2 is to use their curvatures. More specifically, the highly curved parts of ∂S_1 should correspond to the highly curved parts of ∂S_2, while the relatively flat parts of ∂S_1 should correspond to the relatively flat parts of ∂S_2. Define the accumulated arc length and the accumulated curvature of a planar curve C by $x_t = \sum_{i=0}^{t} l(i)$ and $y_t = \sum_{i=0}^{t} \kappa(i)$. Here, $l(i)$ and $\kappa(i)$ are respectively the arc length and the curvature approximated at the i-th point on C. Then, the function ψ defined by $\psi(x_t) = y_t$ for all t encodes the curvature distribution of C. Using this idea, we obtain two functions ψ_1, ψ_2 that represent the curvature variations of ∂S_1 and ∂S_2, with a reparameterization of the domains of ψ_1, ψ_2 to be $[0,1]$ and consider $\psi_1, \psi_2 : [0,1] \to \mathbb{R}$.

Now, we match the curvature variations of ∂S_1 and ∂S_2 by aligning ψ_1 and ψ_2. To do this in a reparameterization independent way, we use the square root velocity function (SRVF) dynamic warping method [128,79] that considers a bijection from ψ_1, ψ_2 to the square root velocity functions (SRVFs) $q_1, q_2 : [0,1] \to \mathbb{R}$ defined by

$$q_1 = \operatorname{sgn}(\dot{\psi}_1)\sqrt{|\dot{\psi}_1|} \quad \text{and} \quad q_2 = \operatorname{sgn}(\dot{\psi}_2)\sqrt{|\dot{\psi}_2|}. \tag{5.3}$$

Instead of aligning ψ_1, ψ_2 directly, the SRVF dynamic warping method finds the optimal alignment

of ψ_1, ψ_2 by aligning the SRVFs q_1, q_2 using a warping function

$$\gamma^* \in \Gamma := \{\gamma : [0,1] \to [0,1] | \gamma(0) = 0, \gamma(1) = 1, \gamma \text{ is a diffeomorphism}\}. \tag{5.4}$$

To find the optimal γ^*, we solve the following minimization problem using dynamic programming:

$$\gamma^* = \text{argmin}_{\gamma \in \Gamma} \| q_1 - (q_2 \circ \gamma)\sqrt{\dot{\gamma}} \|_2, \tag{5.5}$$

where $\| \cdot \|_2$ denotes the Euclidean 2-norm. We obtain a curvature-guided correspondence

$$\psi_1(t) \leftrightarrow \psi_2(\gamma(t)), t \in [0,1]. \tag{5.6}$$

This gives us the desired curvature-guided map $\phi : \partial S_1 \to \partial S_2$.

In case there are boundary landmark constraints as specified in Equation (5.2), one can partition ∂S_1 and ∂S_2 according to the boundary landmarks $\{l_k^{bdy_1}\}_{k=1}^n$ and $\{l_k^{bdy_2}\}_{k=1}^n$ and n pairs of corresponding boundary segments. For each pair of segments, we deploy the above procedures and match them based on their curvature variations. Ultimately, this results in a curvature-guided boundary map $\phi : \partial S_1 \to \partial S_2$ satisfying

$$\phi(l_k^{bdy_1}) = l_k^{bdy_2}, k = 1, 2, ..., n. \tag{5.7}$$

5.1.2 Quasi-conformal theory and Teichmüller maps

Since conformal maps preserve angles, infinitesimal circles are mapped to infinitesimal circles under any conformal maps. This condition is relaxed under quasi-conformal maps. Mathematically, a quasi-conformal map $f : D \subset \mathbb{C} \to \mathbb{C}$ is a homeomorphism satisfying the Beltrami equation $\frac{\partial f}{\partial \overline{z}} = \mu_f(z)\frac{\partial f}{\partial z}$ for some complex-valued function $\mu_f(z)$ with $\|\mu_f(z)\|_\infty < 1$. Here, μ_f is called the Beltrami coefficient of f. Intuitively, quasi-conformal maps send infinitesimal circles to infinitesimal ellipses with bounded eccentricity. To see this, let z_0 be a point in D. The first order approximation of f around z_0 is given by

$$\begin{aligned} f(z) &\approx f(z_0) + f_z(z_0)(z - z_0) + f_{\overline{z}}(z_0)\overline{z - z_0} \\ &= f(z_0) + f_z(z_0)\left(z - z_0 + \mu_f(z_0)\overline{z - z_0}\right). \end{aligned} \tag{5.8}$$

In other words, infinitesimal circles are mapped to infinitesimal ellipses with the maximum magnification $|f_z|(1 + |\mu_f|)$ and the maximum shrinkage $|f_z|(1 - |\mu_f|)$, so that the aspect ratio of the ellipses, the dilatation, is $\frac{1 + |\mu_f|}{1 - |\mu_f|}$. The maximal dilatation of the quasi-conformal map f is then defined by $K_f = \frac{1 + \|\mu_f\|_\infty}{1 - \|\mu_f\|_\infty}$. Among all quasi-conformal maps, Teichmüller maps achieve a constant $|\mu_f(z)|$ over the entire domain D, so that every infinitesimal circle on D is mapped to an infinitesimal ellipse with a constant aspect ratio [51].

Teichmüller maps are advantageous for morphometrics for the following reasons. First, given any boundary correspondence $\phi : \partial S_1 \to \partial S_2$ and landmark constraints $\{l_i^1\} \subset S_1 \leftrightarrow \{l_i^2\} \subset S_2$, there exists a unique landmark-matching Teichmüller map f that achieves the minimum maximal

dilatation over the space of all landmark-matching quasi-conformal maps[131], i.e.

$$f = \text{argmin}_{h:h|_{\partial S_1}=\phi, h(l_i^1)=l_i^2} K_h. \tag{5.9}$$

Second, the bijectivity of Teichmüller maps is guaranteed[51]. This gives a 1-1 correspondence between every part of two shapes, thereby facilitating the pairwise comparison between shapes. Third, as the norm of the Beltrami coefficient μ_f is constant over the entire domain, we have a natural a measure of the dissimilarity between two shapes. Note that $|\mu_f|$ always lies within $[0, 1)$, and equals 0 if and only if f is conformal. Hence, if $1 - |\mu_f| = 1$, then the two shapes are identical up to conformal maps. If $1 - |\mu_f| \ll 1$, there is a large local quasi-conformal dissimilarity between the two shapes. Besides, if we denote $w = f(z)$, we have

$$0 = \mu_{f^{-1}\circ f} = \frac{(f^{-1}\circ f)_{\bar{z}}}{(f^{-1}\circ f)_z} = \frac{((f^{-1})_w \circ f)f_{\bar{z}} + ((f^{-1})_{\bar{w}} \circ f)\bar{f}_{\bar{z}}}{((f^{-1})_w \circ f)f_z + ((f^{-1})_{\bar{w}} \circ f)\bar{f}_z} = \frac{\mu_f + (\mu_{f^{-1}} \circ f)\frac{\bar{f}_{\bar{z}}}{f_z}}{1 + (\mu_{f^{-1}} \circ f)\frac{\bar{f}_z}{f_z}}. \tag{5.10}$$

This implies that $|\mu_f(z)| = |\mu_{f^{-1}}(f(z))|$ for all z. Therefore, $1 - |\mu_f|$ provides an inverse consistent measurement of dissimilarity between any two planar shapes with prescribed landmarks.

For the computation of discrete Teichmüller maps, we make use of the QC Iteration algorithm[86,92], which is an efficient iterative algorithm with guaranteed convergence[85]. Using the boundary correspondence $f|_{\partial S_1} = \phi$, along with the QC Iteration algorithm, we can obtain a curvature-guided Teichmüller map $f : S_1 \to S_2$ satisfying the interior landmark constraints in Equation (5.1). Since f is Teichmüller, the norm of the associated Beltrami coefficient $|\mu_f|$ is a

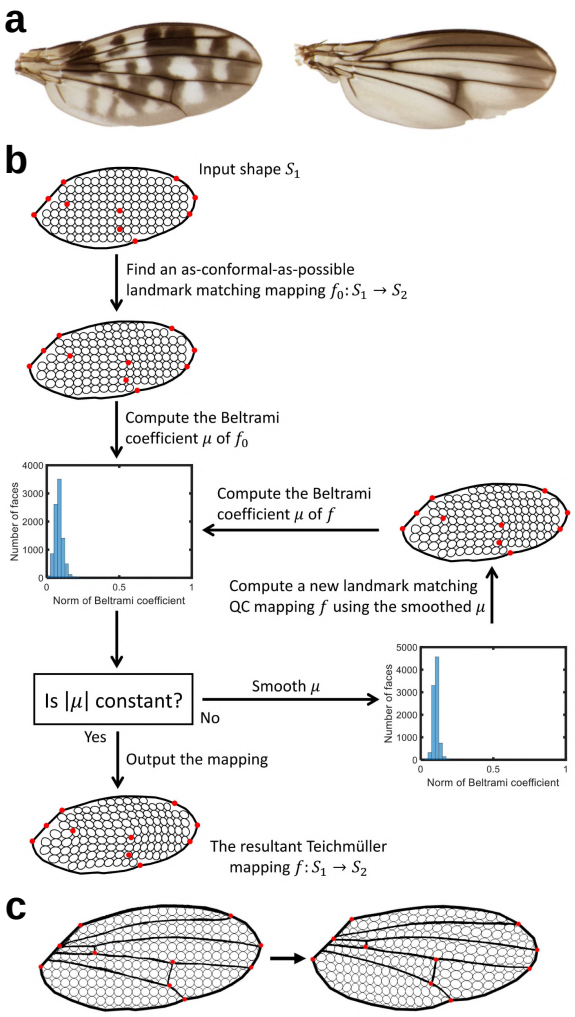

Figure 5.1: An illustration of the landmark-matching Teichmüller mapping algorithm. a, Two planar insect wing shapes to be compared. **b**, A flow chart describing the approach following the QC Iteration algorithm [86,92] for computing discrete landmark-matching Teichmüller maps. We first discretize the insect wing images and use a curvature-guided boundary correspondence to obtain a landmark-matching initial map. Then we iteratively solve for a landmark-matching Teichmüller map. **c**, The resulting landmark-matching Teichmüller map, with small circles in the left shape mapped to small ellipses with a uniform aspect ratio in the right shape. The red landmark points of vein intersections on the source wing shape are exactly mapped to the corresponding landmark points on the target wing shape. The wing images are adapted from the Hawaiian Drosophila Wing Database [42].

constant, and the quantity $1 - |\mu_f|$ is a measure of quasi-conformal similarity between S_1 and S_2.

The procedure is summarized as Algorithm 5.1. An illustration of the algorithm is given in

Figure 5.1.

Algorithm 5.1: Landmark-matching curvature-guided Teichmüller map for planar shapes

Input: Two planar shapes S_1, S_2, with interior landmarks $\{l_k^{int_1}\}_{k=1}^m$, $\{l_k^{int_2}\}_{k=1}^m$, boundary landmarks $\{l_k^{bdy_1}\}_{k=1}^n$, $\{l_k^{bdy_2}\}_{k=1}^n$.

Output: A landmark-matching, curvature-guided Teichmüller mapping $f : S_1 \to S_2$, a similarity score s.

1 Compute a curvature-guided boundary mapping $\phi : \partial S_1 \to \partial S_2$ satisfying $\phi(l_k^{bdy_1}) = l_k^{bdy_2}$ for all $k = 1, 2, \ldots, n$;

2 With the boundary correspondence $\phi : \partial S_1 \to \partial S_2$, compute a landmark-matching Teichmüller mapping $f : S_1 \to S_2$ satisfying $f|_{\partial S_1} = \phi$ and $f(l_k^{int_1}) = l_k^{int_2}$ for all $k = 1, 2, \ldots, m$;

3 Compute the Beltrami coefficient μ_f of the mapping f. The score s is given by $1 - |\mu_f|$;

5.2 Statistical analysis of the quasi-conformal similarity matrix

Suppose we are given a set of planar shapes $\{S_i\}_{i=1}^p$. Using the above mapping method, we can construct a $p \times p$ similarity matrix M, where the (i, j)-th entry of M represents a measure of similarity between S_i and S_j defined by $1 - |\mu_{f_{ij}}|$, with $f_{ij} : S_i \to S_j$ being the desired landmark-matching, curvature-guided Teichmüller map between S_i and S_j. The values of all entries of M are within the range $[0, 1]$, where a larger value indicates a higher level of similarity. We can then use this similarity matrix M to perform a statistical analysis and cluster the set of shapes.

5.2.1 Adaptive thresholding

Note that the similarity matrix M is dense with $p^2 - p$ nonzero entries in general. To highlight the important information in it, we propose an adaptive thresholding algorithm that iteratively modifies

and sparsifies M.

Consider M as the adjacency matrix of a weighted directed graph with p vertices, where every vertex represents a shape in the set $\{S_i\}_{i=1}^{p}$, and every pair of vertices are connected by two directed weighted edges. The weight of the directed edge $[i, j]$ is given by M_{ij}. Intuitively, from the perspective of one of the shapes S_i, a larger M_{ij} among all entries $\{M_{i1}, M_{i2}, \ldots, M_{ip}\}$ indicates a higher level of similarity between S_i and S_j as compared to other shapes. Consider a weighted directed graph with M being the weighted adjacency matrix. In other words, the weight of the directed edge $[i, j]$ from vertex i to vertex j is given by M_{ij}. Given a thresholding parameter λ, to determine the importance of the directed edge $[i, j]$ from vertex i, we consider the quantity $\nu_i = \overline{M_i} + \lambda\sigma_i$, where $\overline{M_i}$ and σ_i are respectively the mean and the standard deviation of $\{M_{ik}\}_{k=1}^{p}$. If $M_{ij} > \nu_i$ we set $M_{ij} = 1$. If not, we neglect the edge by setting $M_{ij} = 0$. For a pair of directed edges $[i, j]$ and $[j, i]$, there are exactly three possibilities: (i) $M_{ij} = M_{ji} = 1$, i.e. S_i and S_j are similar to each other, (ii) $M_{ij} = M_{ji} = 0$, i.e. S_i, S_j are dissimilar, and (iii) $M_{ij} = 1, M_{ji} = 0$ or $M_{ij} = 0, M_{ji} = 1$, i.e. it is not clear that whether S_i, S_j are sufficiently similar to be grouped in one community. To better represent this, we symmetrize M by taking $M \longleftarrow \dfrac{M + M^T}{2}$, so that $M_{ij} \in \{0, \frac{1}{2}, 1\}$ indicates the relationship of S_i, S_j for all $i, j = 1, \ldots, p$. We repeat the thresholding and symmetrizing steps on M until the result converges. The algorithm is summarized as Algorithm 5.2.

Now, we prove that Algorithm 5.2 converges for any M and any λ.

Theorem 5.1. *Algorithm 5.2 converges for any similarity matrix M and for any thresholding*

Algorithm 5.2: Adaptive thresholding

Input: A $n \times n$ similarity matrix M, a thresholding parameter λ.
Output: A thresholded matrix where all entries are 0, $\frac{1}{2}$ or 1.

Set $M^0 = M$;
Set $k = 0$;
repeat
  Update k by $k + 1$;
  For each row i, denote $v_i^k = \overline{M_i^k} + \lambda\sigma_i^k$, where $\overline{M_i^k}$ and σ_i^k are respectively the mean and the standard deviation of $\{M_{it}^k\}_{t=1}^n$. Set $M_{ij}^k = \begin{cases} 1 & \text{if } M_{ij}^{k-1} \geq v_i^{k-1}, \\ 0 & \text{otherwise.} \end{cases}$;
  Update M^k by $\frac{M^k + (M^k)^T}{2}$;
until $M^k = M^{k-1}$;

parameter λ.

Proof. Note that at each iteration, we always have $1 \to 1$, $0 \to 0$, and $\frac{1}{2} \to (1 \text{or} \frac{1}{2} \text{or} 0)$. Let n_k be the number of $\frac{1}{2}$ in the matrix M^k. It is easy to see that the sequence $\{n_k\}_{k=1}^{\infty}$ is non-increasing. Also, note that $0 \leq n_k \leq n_1 \leq |M|$ where $|M|$ denotes the total number of entries in M. By Monotone Convergence Theorem, $\{n_k\}_{k=1}^{\infty}$ converges and $0 \leq \lim_{k\to\infty} n_k \leq n_1$.

Now, it is easy to see that if $n_{K+1} = n_K$ then $M^{K+1} = M^K$, which indicates the convergence. By the symmetry of M^k, if $n_{K+1} < n_K$, then $n_K - n_{K+1}$ must be a multiple of 2. Hence, the maximum number of iterations needed for achieving convergence is bounded above by $n_1/2$. ■

5.2.2 Clustering and community detection

Finally, to cluster all shapes into several communities based on the thresholded matrix M, we apply a recent community detection method[97] that accounts for the non-locality and asymmetry of the

connections between edges[96]. The method is briefly described below.

Denote the set of shapes $\{S_i\}_{i=1}^p$ by $\mathcal{S}$. The similarity between two communities $C_I, C_J \subset \mathcal{S}$ is given by $sim_{IJ} = sim^{in} - sim^{out}$, where $sim^{in} = \frac{1}{\|C_I \cup C_J\|^2} \sum_{i,j \in C_I \cup C_J} g_{ij}$ represents the average similarity score inside the communities, and $sim^{out} = \frac{1}{\|C_I \cup C_J\| \cdot \|\mathcal{S} \setminus C_I \cup C_J\|} \sum_{i \in \mathcal{S} \setminus C_I \cup C_J} \sum_{j \in C_I \cup C_J} g_{ij}$ represents the average similarity score outside the communities. At each iteration, the communities with a high sim_{IJ} are combined until the grouping result stabilizes. The final communities formed represent the clustering result based on our Teichmüller morphometric method. The procedure is summarized as Algorithm 5.3.

Algorithm 5.3: Cluster analysis of planar shapes via landmark-matching curvature-guided Teichmüller maps

Input: A set of planar shapes $\{S_i\}_{i=1}^p$ with prescribed landmark correspondences.
Output: Community labels $\{l_i\}_{i=1}^p$.

1. Apply Algorithm 5.1 for all pairs of shapes $(S_i, S_j), 1 \leq i, j \leq p$. Denote the similarity score between them by s_{ij};
2. Construct a $p \times p$ similarity matrix $M = (s_{ij})$;
3. Apply Algorithm 5.2 on M and obtain the thresholded matrix;
4. Apply the community detection method[97] with the thresholded matrix and obtain the community labels $\{l_i\}_{i=1}^p$;

This completes our Teichmüller morphometric framework for planar shapes.

5.3 Quantifying wing shape in evolution and development

5.3.1 Phenotypic variation of Hawaiian Drosophila wings

We deploy our Teichmüller-map morphometric framework for studying the *Drosophila* wings in the Hawaiian Drosophila Wing Database[42], with a total of 128 wings drawn from four "picture wing" phylogenetic groups, including the *adiastola* group, the *planitibia* group, the *glabriapex* group and the *grimshawi* group (Table 5.1). We discretize every wing image from the different species using a triangle mesh with approximately 9000 triangle elements. As shown in Figure 5.2, seven points are manually chosen as boundary landmarks for each wing, including the intersections between the longitudinal veins L2, L3, L4, L5 and the wing boundary. Also, three intersections between the veins are manually chosen as interior landmarks, including the intersections between L4 and the anterior cross-vein (ACV) and the posterior cross-vein (PCV), and the intersection between L5 and PCV. Figure 5.3 shows the performance of our landmark-matching Teichmüller mapping method and some prior methods for comparing a pair of wings from the *D. punalua* and the *D. silvestris* species. It can be observed that our approach is capable of matching the boundary shapes and interior landmarks of the two wings accurately, while the prior methods are unable to match all landmarks and the wing boundaries exactly.

After demonstrating the effectiveness of our method for pairwise comparison, we compute the 128×128 similarity matrix M with $M_{ij} = 1 - |\mu_f(i,j)|$ for the entire set of shapes (Figure 5.4). We then apply Algorithm 5.2 with the thresholding parameter $\lambda = 1$ and the community detection

Species	Phylogenetic group	Specimen number
clavisetae	*adiastola*	1–5
ornata	*adiastola*	6–7
setosimentum	*adiastola*	8–10
adiastola	*adiastola*	11–14
cilifera	*adiastola*	15–20
hamifera	*adiastola*	21–26
spectabilis	*adiastola*	27–32
heteroneura	*planitibia*	33–37
planitibia	*planitibia*	38
silvestris	*planitibia*	39–40
nigribasis	*planitibia*	41–42
cyrtoloma	*planitibia*	43–46
melanocephala	*planitibia*	47–48
neoperkinsi	*planitibia*	49–51
neopicta	*planitibia*	52–55
oahuensis	*planitibia*	56–57
obscuripes	*planitibia*	58–60
hemipeza	*planitibia*	61–63
picticornis	*planitibia*	64–65
aglaia	*glabriapex*	66–67
basisetae	*glabriapex*	68–70
digressa	*glabriapex*	71–72
discreta	*glabriapex*	73–74
fasciculisetae	*glabriapex*	75
glabriapex	*glabriapex*	76–77
macrothrix	*glabriapex*	78–80
montgomeryi	*glabriapex*	81–83
punalua	*glabriapex*	84–87
affinidisjuncta	*grimshawi*	88–89
balioptera	*grimshawi*	90–91
bostrycha	*grimshawi*	92–93
craddockae	*grimshawi*	94
crucigera	*grimshawi*	95–98
disjuncta	*grimshawi*	99–103
grimshawi	*grimshawi*	104
heedi	*grimshawi*	105–106
silvarentis	*grimshawi*	107–108
limitata	*grimshawi*	109–112
engyochracea	*grimshawi*	113–115
hawaiiensis	*grimshawi*	116–118
murphyi	*grimshawi*	119–120
orphnopeza	*grimshawi*	121–122
orthofascia	*grimshawi*	123
recticilia	*grimshawi*	124–125
sproati	*grimshawi*	126–127
villosipedis	*grimshawi*	128

Table 5.1: The list of wing specimens adapted from the Hawaiian Drosophila Wing Database [42]. The specimen numbers represent the row/column numbers corresponding to the specimens in the similarity matrix.

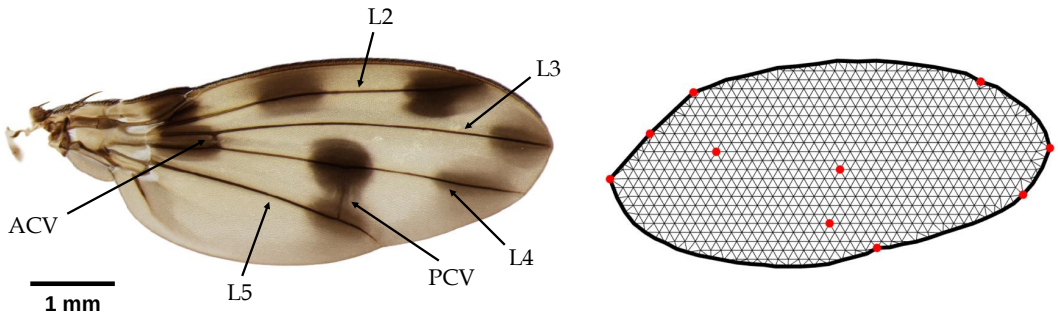

Figure 5.2: A Hawaiian *Drosophila* wing and the finite element discretization of it. Red: Landmark points of the intersections between the longitudinal veins L2, L3, L4, L5, the anterior cross-vein (ACV), the posterior cross-vein (PCV) and the boundary. The wing image is adapted from the Hawaiian Drosophila Wing Database [42].

method [97] on the thresholded similarity matrix for clustering the wing shapes (Algorithm 5.3). To visualize the result, we apply the multidimensional scaling (MDS) method to project the information of the thresholded matrix onto the Euclidean plane, with the similarity information preserved as distances between nodes. Every specimen in our dataset is visualized as a node on the plane. The clustering result produced by our framework is shown in Figure 5.5. It can be observed that the nodes are clustered into three groups, with the species *D. glabriapex* (denoted by Community 1), *D. planitibia* (denoted by Community 2) and *D. grimshawi* (denoted by Community 3) being a representative in each of them.

Of the 22 specimens in the *glabriapex* phylogenetic group, 20 (91%) of them are clustered into Community 1. Of the 33 specimens in the *planitibia* phylogenetic group, 28 (85%) of them are clustered into Community 2. Of the 41 specimens in the *grimshawi* phylogenetic group, 21 (51%) of them are clustered into Community 3 and 18 (44%) are clustered into Community 1. Of the 32 specimens in the *adiastola* phylogenetic group, 18 (56%) of them are clustered into Community 3 and 12 (38%) are clustered into Community 1. From the above, it can be observed that Community

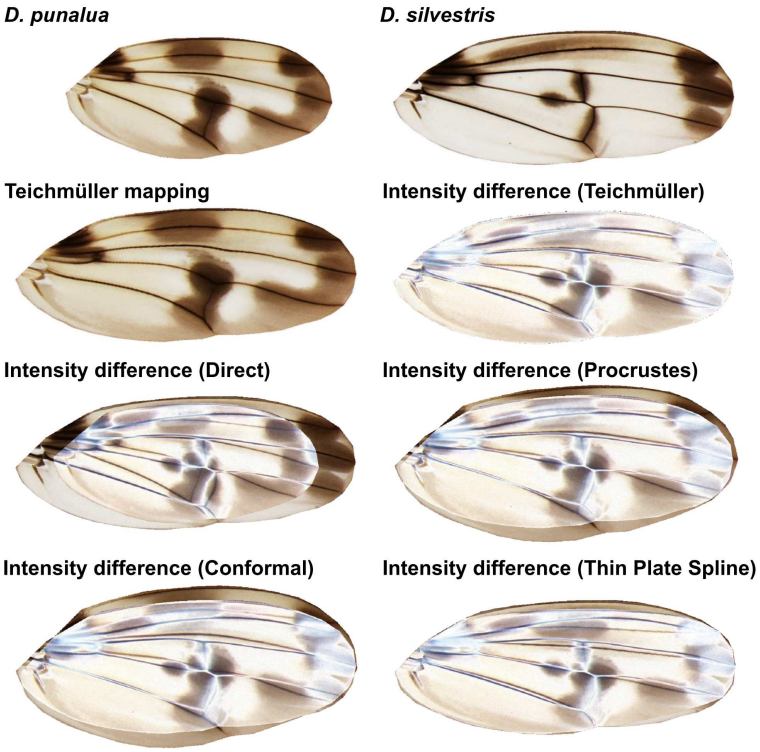

Figure 5.3: A comparison between our proposed method and four prior morphometric approaches. First row: The *D. punalua* and *D. silvestris* wings. Second row: The landmark-matching Teichmüller map of *D. punalua* onto *D. silvestris*, and the intensity difference between the mapping result and the *D. silvestris* wing. The third and fourth rows: The intensity differences computed using direct mapping, Procrustes superimposition [55], least-square conformal mapping and Thin Plate Spline [11]. The *D. punalua* and *D. silvestris* wing images are adapted from the Hawaiian Drosophila Wing Database [42].

2 primarily consists of wings from the *planitibia* phylogenetic group but not the other three groups. This suggests that the wings in the *planitibia* phylogenetic group share highly similar phenotypic features and are very different from the wings in all other phylogenetic groups. Noticing the high percentage of specimens in the *glabriapex* phylogenetic group classified into Community 1, we deduce that there is also a high level of similarity among the wings in the *glabriapex* phylogenetic group. By contrast, the *adiastola* phylogenetic group and the *grimshawi* phylogenetic group

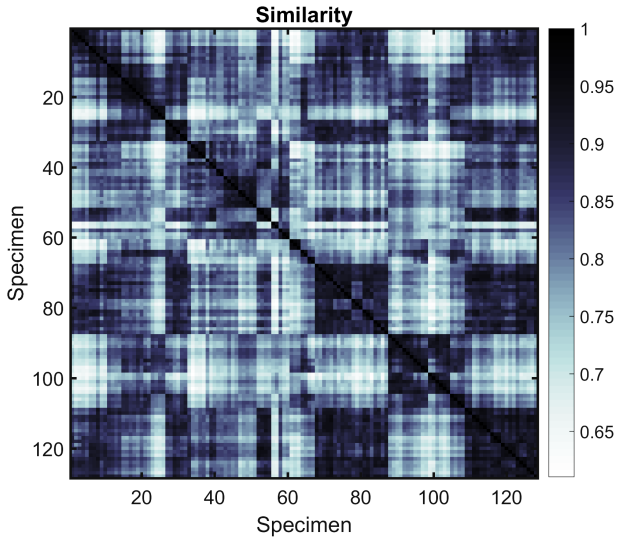

Figure 5.4: The similarity matrix for the 128 wings in the Hawaiian Drosophila Wing Database [42] obtained by our landmark-matching, curvature-guided Teichmüller mapping method.

demonstrate a higher level of shape diversity as both of the two groups are primarily clustered into two communities.

Now, we further analyze the phenotypic features of the three communities qualitatively to understand the community detection results. The images of the wings in Community 1, Community 2 and Community 3 are respectively shown in Figure 5.6, Figure 5.7 and Figure 5.8. There is a notable difference in the wing geometries of the three communities in terms of the wing shapes and the relative locations of the landmarks. More specifically, for the wings in Community 1, a relatively round shape at the bottom of the wing boundary and a relatively sharp wing tip can be observed in general, and the intersection between L5 and PCV is relatively far away from the wing boundary. For the wings in Community 2, they are in general with an elongated shape, and the intersection between L5 and PCV is relatively close to the wing boundary. For the wings in

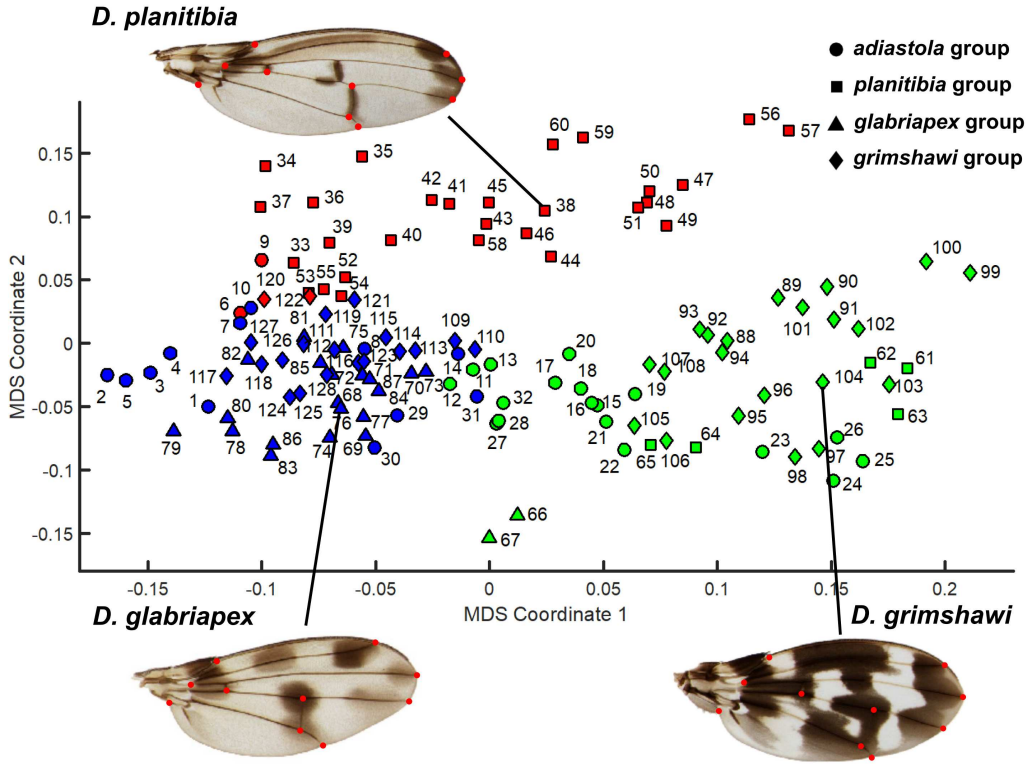

Figure 5.5: The community detection result obtained by our proposed framework visualized on the multidimensional scaling (MDS) coordinate plane. Every specimen is represented as a node on the plane constructed by MDS. The nodes are color-coded by the community labels obtained by our framework. Blue: Community 1. Red: Community 2. Green: Community 3. The shapes of the nodes represent their phylogenetic groupings. Circle: The *adiastola* group. Square: The *planitibia* group. Triangle: The *glabriapex* group. Diamond: The *grimshawi* group. The number beside each node represents the specimen number specified in Table 5.1. The wing images shown are adapted from the Hawaiian Drosophila Wing Database[42].

Community 3, we observe a round silhouette in general, and the intersection between L4 and ACV is relatively distal when compared with that in the other two communities.

There are a few interesting exceptions in the community detection result for the *planitibia* phylogenetic group. For the two species *D. hemipeza* and *D. picticornis*, which belong to the *planitibia* phylogenetic group but are clustered into Community 3, it can be observed that their

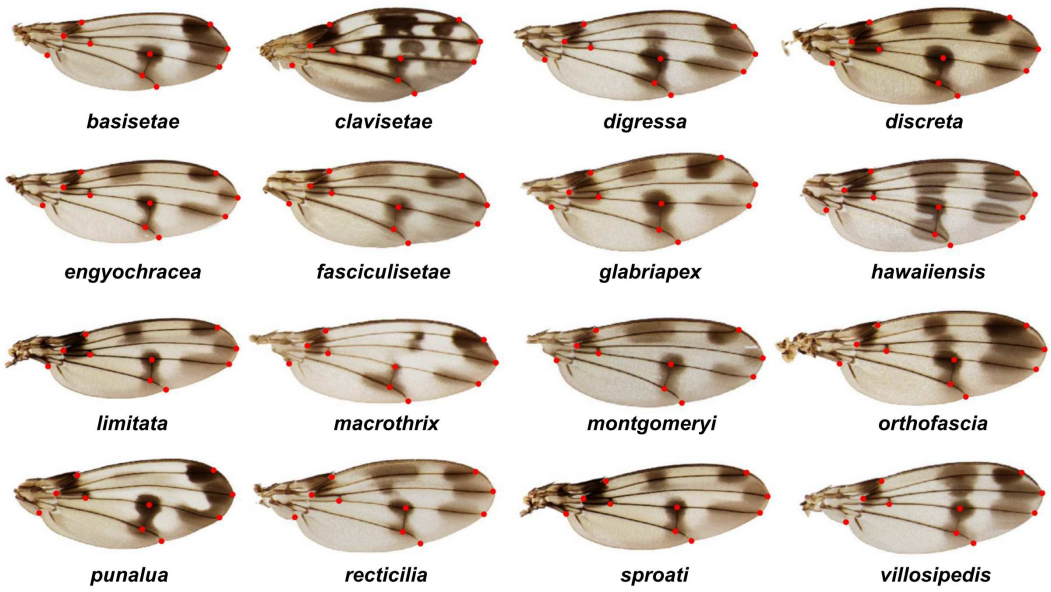

Figure 5.6: The species clustered as Community 1 by our framework. The images are adapted from the Hawaiian Drosophila Wing Database[42] (not to scale).

wings are different from the other wings in the *planitibia* phylogenetic group. More specifically, the *D. picticornis* wing has a relatively round shape and is more pigmented than all other wings in the *planitibia* phylogenetic group. This can possibly be explained by the early division in the phylogeny of the *planitibia* phylogenetic group[9] that separates *D. picticornis* from the other species. For *D. hemipeza*, we observe that the intersection between L4 and ACV is distal relative to the same landmark for the other wings in the *planitibia* phylogenetic group.

While we have been focusing on landmarks based on wing venation network motifs, it is natural to ask if our method might shed light on the pigmentation patterns, a trait that is controlled by just a few genes[104] and likely to be more labile. We observe three different pigment patterns in the three communities we obtained. More specifically, the majority of Community 1 possesses a moderate

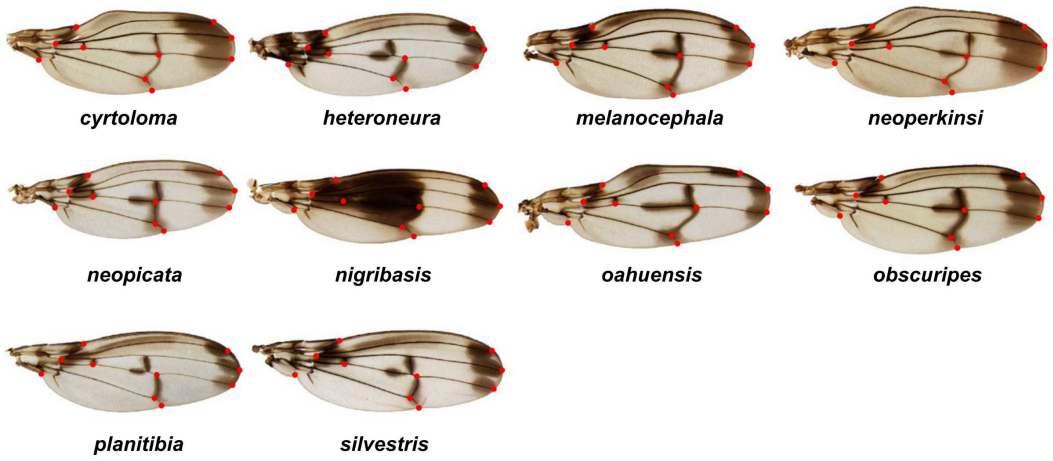

Figure 5.7: The species clustered as Community 2 by our framework. The images are adapted from the Hawaiian Drosophila Wing Database[42] (not to scale).

number of pigment spots, which occur near the top part of the wing, the intersection between L4 and PCV, and the intersection between L5 and PCV. The wings in Community 2 possess a small number of pigment spots, which occur near the distal tips of L2, L3 and L4. For Community 3, most wings in the community possess a high level of pigmentation. As our Teichmüller-based classification method is solely based on wing boundary shape and venation landmark positions, it is surprising that the communities formed show a clustering effect with respect to their pigment patterns. This suggests that there may be a crosstalk between the genes encoding for both phenotypes, which is worth exploring further using phylogenetic approaches in future works.

5.3.2 Temporal development of Lepidoptera wings

Besides studying the phenotypic variation of *Drosophila* wings, we can also apply our proposed Teichmüller morphometric method for analyzing the temporal development of Lepidoptera wings.

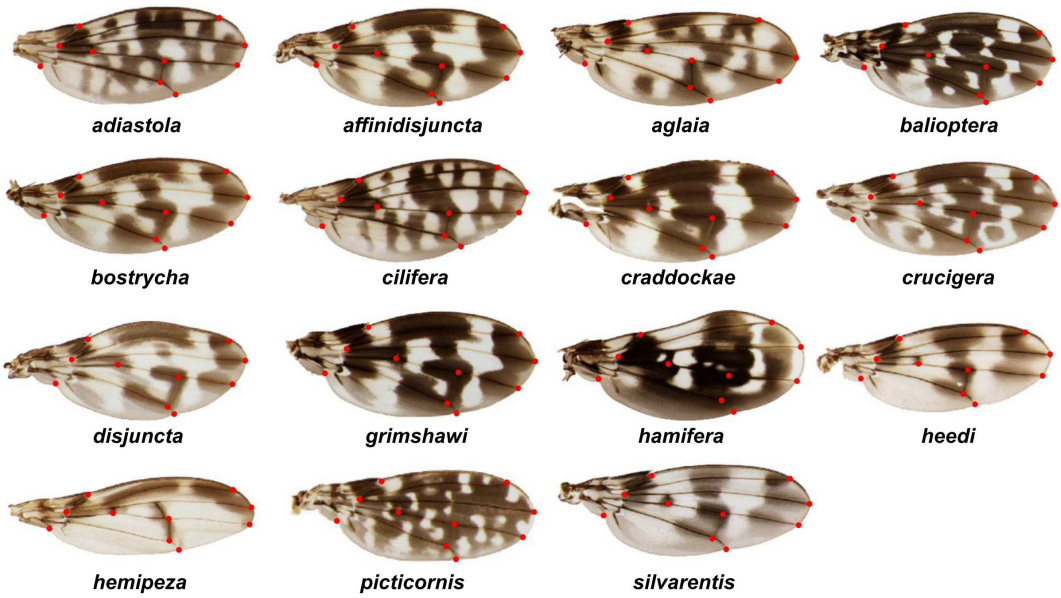

Figure 5.8: The species clustered as Community 3 by our framework. The images are adapted from the Hawaiian Drosophila Wing Database [42] (not to scale).

We deploy our method on the forewings of the species *Manduca sexta* and *Junonia coenia* [101] at the larval, prepupal, pupal and adult stages. Between every two successive developmental stages, we compute a Teichmüller map. The constant norm of the associated Beltrami coefficient between the two stages, denoted by $\Delta_{|\mu|}$, represents the local shear and the quasi-conformal dissimilarity between them. While the local shear is a constant over the entire domain, the local area change and orientation change may vary. To better analyze these quantities, We construct a circle packing on the wing at the earlier temporal stage and map onto the wing at the latter stage using the Teichmüller map. This produces a deformed packing on the wing at the latter stage, where the circles are deformed to ellipses with different size and orientation. Then, the local area change under the mapping can be captured by the change in size of the ellipses relative to the original circles, and

the local orientation change can be captured by the change in the orientation of the ellipses relative to the original circles. More explicitly, we quantify the change in size by $\Delta_A = \frac{\text{Area of ellipse}}{\text{Area of circle}}$ for each small circle, and the change in orientation by $\Delta_p =< 2\cos^2\theta - 1 >$ which averages over the 1-ring neighborhood of every circle, where θ is the orientation change of the circle under the Teichmüller map. The proximal-distal orientation of the major axis is denoted by $\Delta_p = 1$ while an anterior-posterior orientation of the major axis yields $\Delta_p = -1$.

Figure 5.9 and Figure 5.10 show the Teichmüller maps between successive stages of *Manduca sexta* and *Junonia coenia* wings respectively. We first analyze the overall change between every two successive stages quantified by the quasi-conformal dissimilarity $\Delta_{|\mu|}$, the average local area change mean(Δ_A) and the average local orientation change mean(Δ_p). We observe that the largest $\Delta_{|\mu|}$ occurs between prepupa and pupa for *Manduca sexta*, while $\Delta_{|\mu|}$ increases throughout the development for *Junonia coenia*. It can also be observed that the average local area changes of the two species are different. For *Manduca sexta*, mean(Δ_A) decreases throughout the development, while for *Junonia coenia* the greatest mean(Δ_A) occurs between prepupa and pupa. Besides, it is noteworthy that the magnitude of mean(Δ_A) for *Junonia coenia* is greater than that for *Manduca sexta* between every two successive stages. For the local orientation change, both *Manduca sexta* and *Junonia coenia* wings undergo an overall proximal-distal orientation change at the earlier stages of the development, as mean$(\Delta_p) > 0$. However, the two species are different in the overall orientation change between pupa and adult. For *Manduca sexta* we have mean$(\Delta_p) \approx 0$, which indicates that the overall orientation change is small. By contrast, for *Junonia coenia* there is a notable overall proximal-distal orientation change.

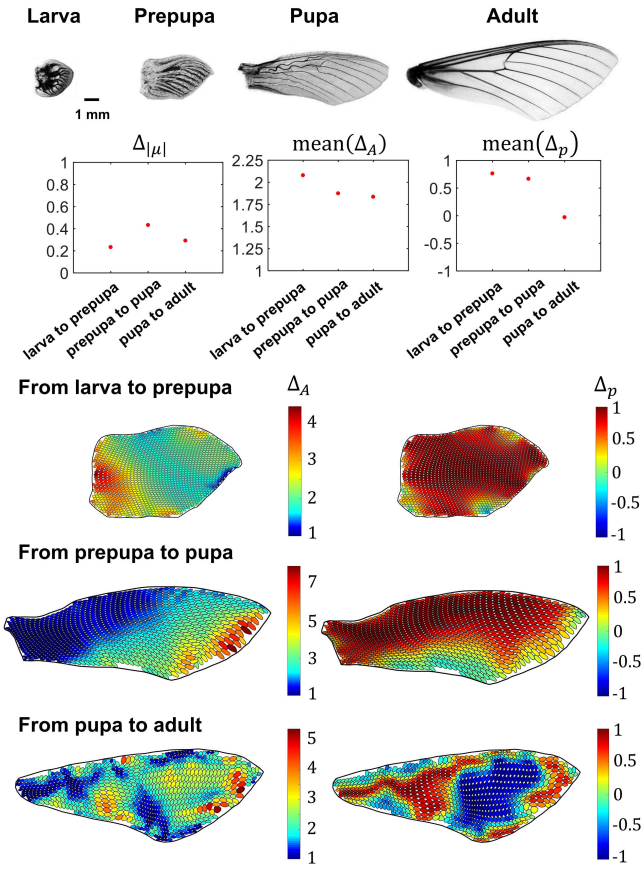

Figure 5.9: Analyzing the temporal development of *Manduca sexta* wings using Teichmüller map. Top row: The *Manduca sexta* wing images [101] at different developmental stages (displayed to scale). A Teichmüller map is computed between every two successive stages. Second row: The quasi-conformal dissimilarity $\Delta_{|\mu|}$, the average of the local size change Δ_A and the average of the local orientation change Δ_p between every two successive stages. The last three rows show the deformation of the circle packing under the Teichmüller map between every two successive stages, visualized on the wing at the latter stage. The left column shows the resulting ellipses color-coded by Δ_A. The right column shows the resulting ellipses color-coded by Δ_p. For visualization purpose, the shapes in the last three rows are rescaled to have the same height, with their aspect ratio kept unchanged.

Now, we analyze the variation of the local area and orientation changes across each deformed circle packing. From the heat maps of Δ_A for both *Manduca sexta* and *Junonia coenia*, it can be observed that the most significant local area change between pupa and adult occurs at the distal half of the wings. However, there is a notable difference between the regions with the greatest Δ_A at the

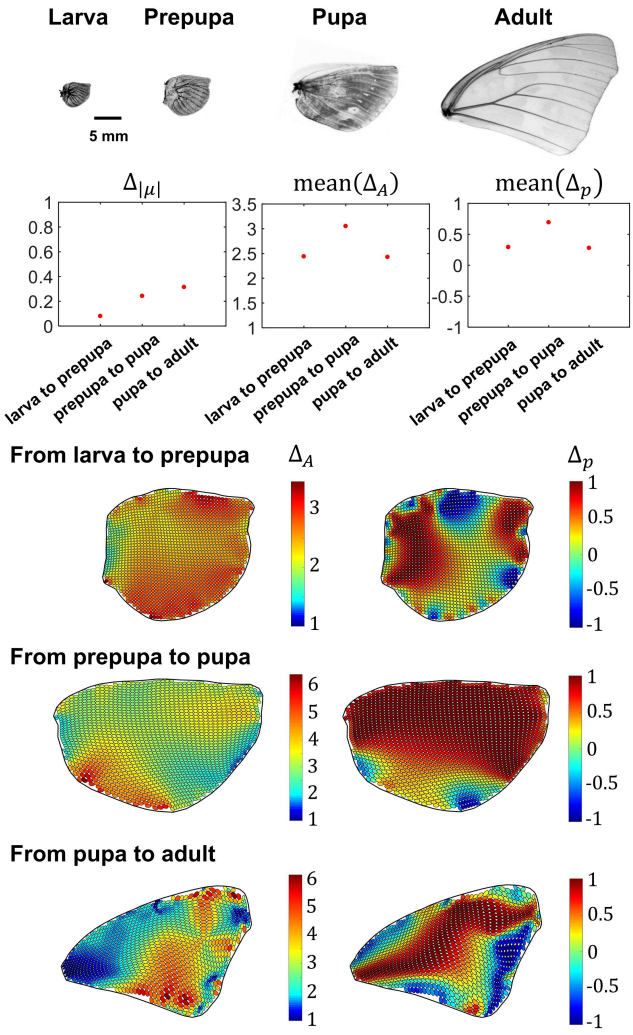

Figure 5.10: Analyzing the temporal development of *Junonia coenia* wings using Teichmüller map. The notation is the same as in Figure 5.9.

earlier stages for the two species. From the heat maps of Δ_p, we also observe a difference between the patterns of the orientation change of the two species. For *Manduca sexta*, the wing primarily undergoes a proximal-distal orientation change from larva to pupa, followed by a significant anterior-posterior orientation change at the central region from pupa to adult. By contrast, for

Junonia coenia, the wing undergoes a more diversified local orientation change from larva to pupa, followed by an anterior-posterior change at the distal region from pupa to adult.

We note that the local change reflected by Teichmüller maps throughout the development of *Manduca sexta* appears to be correlated with the local mitotic density [101]. More specifically, as shown in Figure 5.9, the Teichmüller map between prepupa and pupa results in a small Δ_A at the top part of the wing and a large Δ_A at the distal part, which agrees with the mitotic density distribution at that period. We also observe that both Δ_A and the mitotic density for the period from pupa to adult achieve the greatest value around the tip of the wing and the smallest value at the proximal part. By contrast, for *Junonia coenia*, the correlation between the local change reflected by Teichmüller maps and the local mitotic density is less significant.

We remark that under Teichmüller maps, all local changes are captured by the three quantities $\Delta_{|\mu|}$, Δ_A and Δ_p, where $\Delta_{|\mu|}$ and Δ_p together describe the shape change, and Δ_A describes the size change. Hence, shape and size can be analyzed separately. Therefore, Teichmüller maps provide a way to assess, identify and remove allometry.

5.4 Discussion

Studying shape variation is an important problem in biology. By combining complex analysis, computations and statistics, we have developed a new geometric morphometric approach for the quantification, comparison and classification of planar biological shape. It improves on previous methods by allowing for the mapping between two shapes with arbitrary boundary and landmark

correspondences. More generally, this work enables us to link phenotypes to genotypes by applying the proposed method for analyzing wing shape across species, as well as to describe the process of wing shape developmentally by applying the proposed method for analyzing wing shape over time.

If nature were not beautiful, it would not be worth knowing, and if nature were not worth knowing, life would not be worth living.

Henri Poincaré

6

Ferret brain morphogenesis

Understanding the growth and form of normal and abnormal cortical convolutions is important for the study of human neurodevelopmental diseases. In our recent works, we proposed a model for explaining gyrification based on a simple mechanical instability driven by tangential expansion of the gray matter constrained by the white matter[135] and

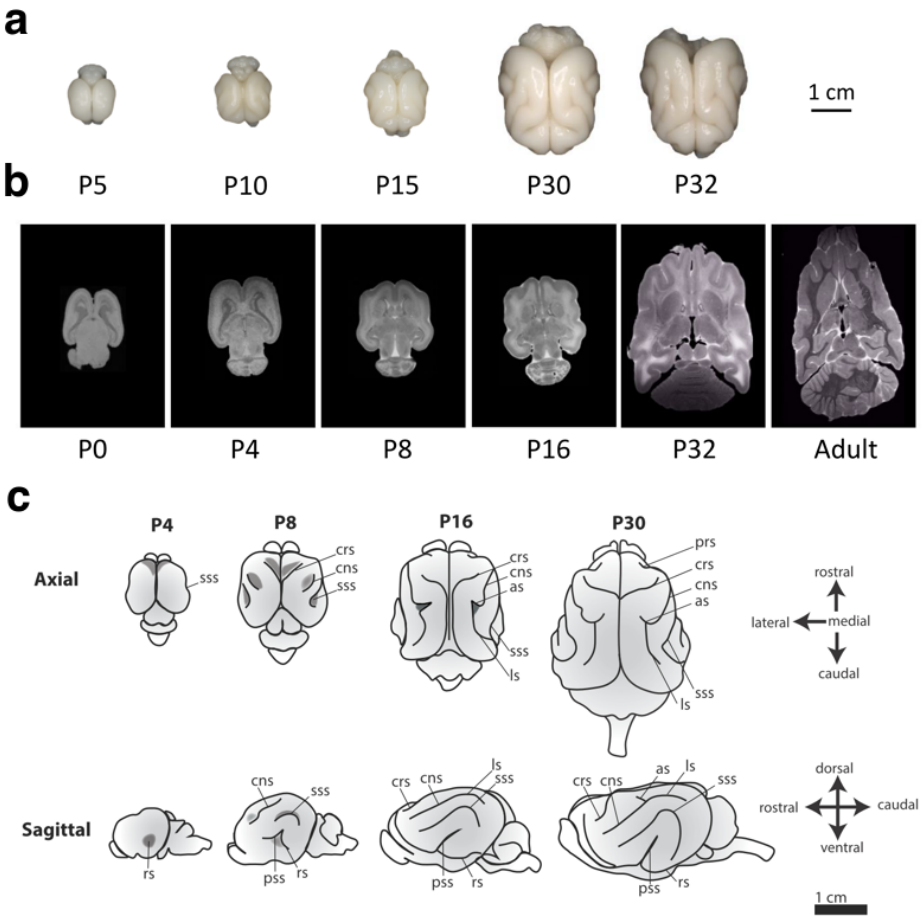

Figure 6.1: Time course of morphogenesis of ferret and gel brains. a, Whole brain samples from ferret. **b**, MRI ferret brain data. **c**, The development of the major sulci of the ferret brain defined in the literature [122,123,99,4,49].

deployed it to simulate normal human cortical convolution [136]. However, studies of cerebral cortical malformations (MCDs) are limited by our ability to identify ongoing malformation of the human fetal brain *in utero*. This limits our understanding of MCDs developmental trajectory.

Unlike human brain, there is a progressive development of the cortical gyri and sulci in ferret brain from postnatal day 0 (P0) to adolescence [122,123,99,4,49] (Figure 6.1). This suggests that ferret

is a good candidate for the study of normal and abnormal neurodevelopmental processes. In this chapter, we first simulate the growth of normal ferret brain using a gel model and a computational model based on the principle of constrained cortical expansion and show that the simulation results agree with the real brain development. We then proceed to use the models to reproduce defective developmental processes of ferret brain.

6.1 Physical gel model of ferret brain folding

We constructed a physical simulacrum of ferret brain folding using the observation that soft physical gels swell superficially when immersed in solvents. We first reconstructed surfaces of pre-swelling brain states from T_2-weighted motion corrected anatomical MR images of ferret brains at postnatal days: P0, P4, P8 and P16. Then, we produced bilayer gel models of the ferret brain at various ages through a combination of 3D printing and replica molding based on the reconstructed brain surfaces[135,136]. We passed two reconstructed surfaces (pial surface and inner cortical surface) to a 3D printer to obtain a 3D printed mold. Then we used the 3D printed mold to cast a negative mold made of supersoft silicone. We used the inner cortical surface mold to produce the core (white matter) composed of lightly cross-linked polydimethylsiloxane (PDMS) elastomer. We assembled the PDMS elastomer prepolymer by mixing the separated base and cross linker components in a 45:1 (base to cross-linker) mass ratio, degassed it in a vacuum to eliminate air bubbles, and poured it into the inner cortical surface negative mold. We brought the poured prepolymer to 75^o C for 45 to 75 minutes to form a PDMS core with a partially cured surface, which we then extracted from the

mold. We then used the pial surface mold to cast the outer layer around the partially cured PDMS core. We prepared a new PDMS prepolymer mixture with a base to cross-linker mass ratio of 28 : 1. We degassed the prepolymer and pour it into the pial surface mold with the previously cured core suspended inside. We brought the fully assembled two-layer gel brain model to 75^o C for a minimum of 2 hours such that it was completely cured. To mimic tangential cortical growth, we immersed the two-layer gel brain model in an organic solvent that permeated the outer (cortical) layer by diffusion and induced it to swell. We chose hexanes as the solvent based on its compatibility with PDMS and resultant ability to induce a high degree of swelling. Figure 6.2 shows the gel brains evolve folds via swelling between stages P8 and P16 in a ferret kit.

6.2 Computational model of ferret brain folding

For the numerical simulation of ferret brain development, we follow the approach in Ref. [135,136] and consider a material consisting of a layer of gray matter on top of a deep layer of white matter. The material is assumed to be neo-Hookean with volumetric strain energy density

$$W = \frac{\mu}{2}\left[Tr(\mathbf{F}\mathbf{F}^T)J^{-2/3} - 3\right] + \frac{K}{2}(J-1)^2, \tag{6.1}$$

where $\mathbf{F}$ is the deformation gradient, $J = \det(\mathbf{F})$, μ is the shear modulus and K is the bulk modulus. We assume that $K = 5\mu$ for a modestly compressible material.

Three geometrical parameters of the 3D brain models are the brain size R, the cortical thickness T and the tangential expansion g^2. For ferret brain development, we follow the empirical scaling law

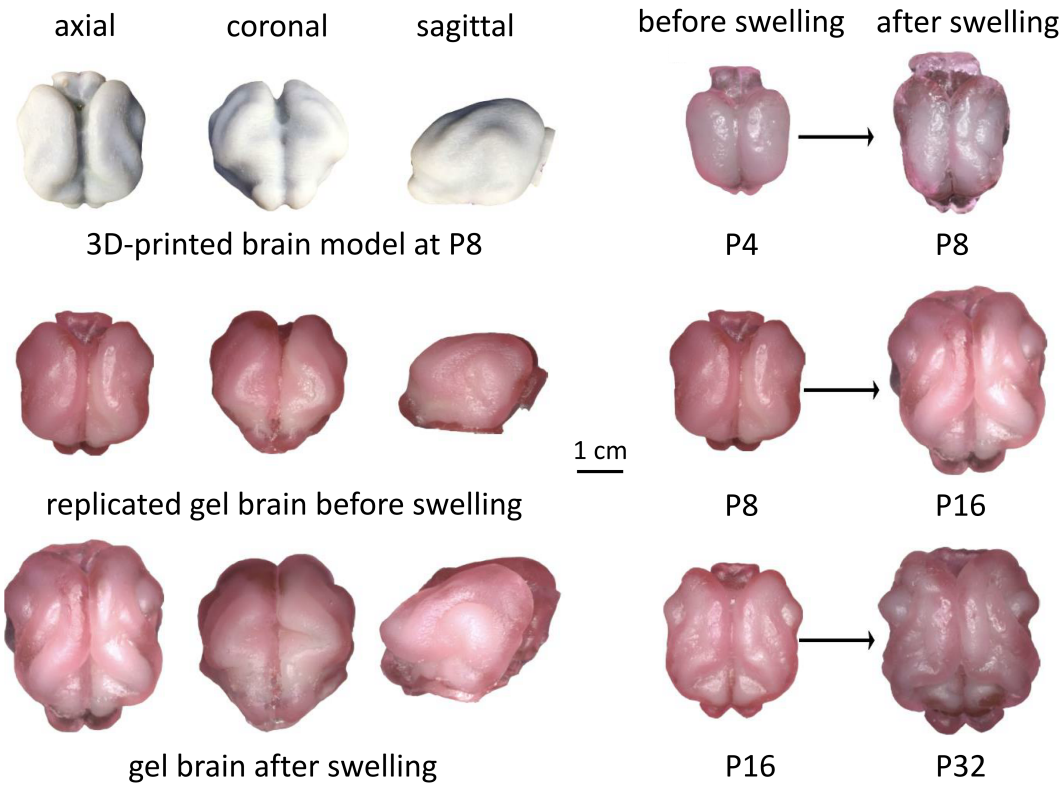

Figure 6.2: Gel model of ferret brain morphogenesis. (Left) Axial, coronal, and sagittal views of 3D-printed brain model at postnatal day 8 (P8) (top), replicated gel brain (middle), and gel brain after swelling (bottom). (Right) Physical gel models of ages P4, P8, and P16 are nonuniformly swollen to mimic progression to states that roughly resemble ages P8, P16, and P32, respectively.

for gray-matter volume to thickness and set $R/T \approx 10$ with the tangential expansion ratio $g \approx 1.9$.

An indicator function

$$\theta(y) = \frac{1}{1 + e^{10(y/T-1)}} \tag{6.2}$$

is applied for distinguishing between the cortical layer (with $\theta = 1$) and white matter zone (with $\theta = 0$). Here, y is the distance from surface in material coordinates.

The brains are discretized in the form of tetrahedral meshes with over 1 million tets using Netgen.

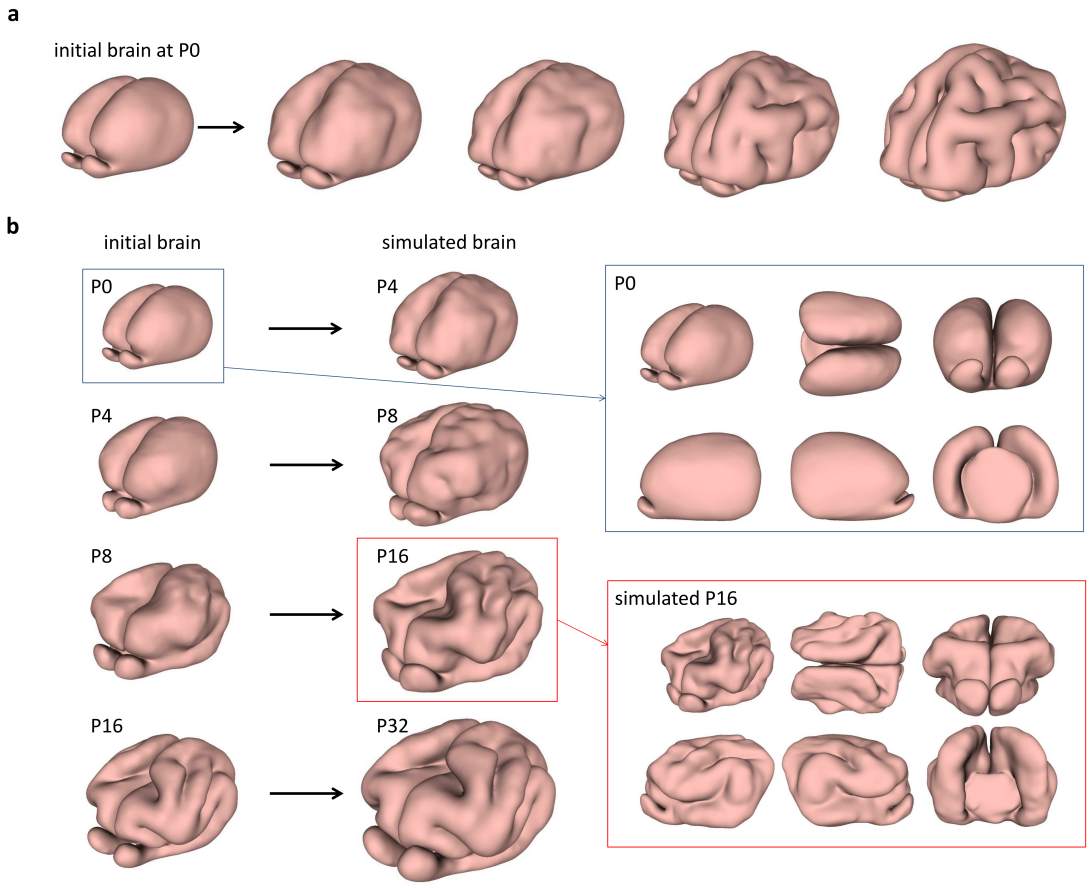

Figure 6.3: Numerical model of ferret brain morphogenesis. a, Continuous growth. **b**, Stepwise growth from P0 to P4, P4 to P8, P8 to P16, and P16 to P32.

The energy of the system is minimized by quasistatic equilibration using an explicit scheme. Growth is applied by expanding the tetrahedral elements with inversion handling [130] and nodal pressure formulation [10]. Self-avoidance of the surface is handled using the penalty based vertex-triangle contact processing [45]. We further enforce that there is no growth at the central part as well as the bottom portion of the brain to better simulate the development of ferret brains.

We consider both continuous simulations from P0 to P32 (Figure 6.3**a**) and stepwise simulations

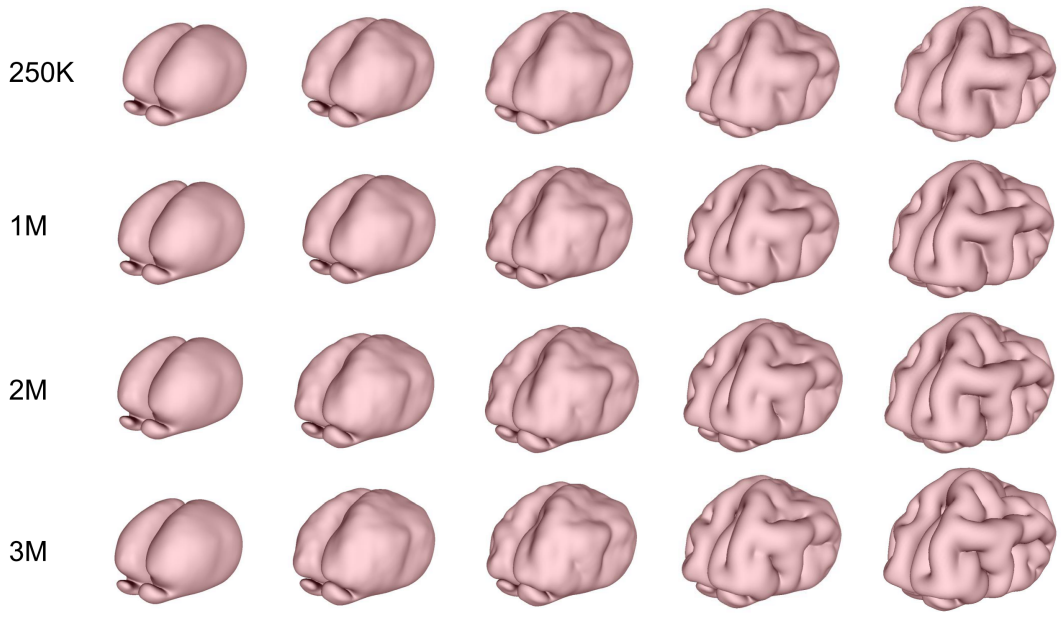

Figure 6.4: Mesh independence of the numerical simulation. It can be observed that the folding patterns produced from tetrahedral meshes with different number of tetrahedral elements are similar in shape.

from P0 to P4, from P4 to P8, from P8 to P16 and from P16 to P32 (Figure 6.3**b**). Comparing the stepwise numerical simulations, the stepwise gel simulations and the real brains, it can be observed that the folding patterns are highly similar.

It is natural to ask if the folding simulation is robust to the mesh resolution. Figure 6.4 shows the folding patterns produced from meshes with different number of tetrahedral elements. It can be observed that the results are not affected by the mesh resolution.

Figure 6.5 shows a comparison between the actual ferret brain gyrification, the numerical brain simulation, and the physical gel brain simulation from P8 to P16. It can be observed that development of major sulci such as the cruciate sulcus (crs), coronal sulcus (cns) and suprasylvian sulcus (sss) is captured by both the numerical model and the physical gel brain model. For a more

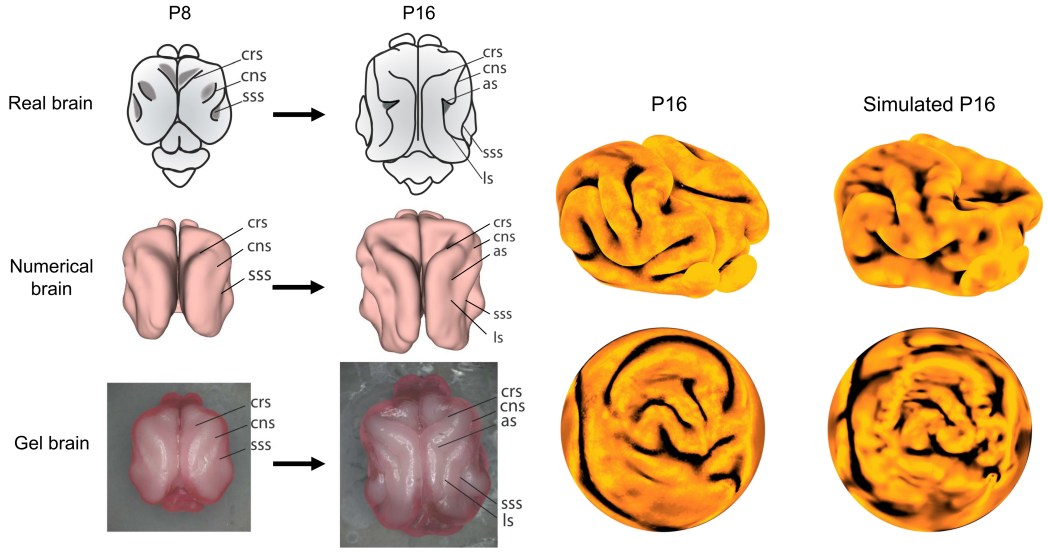

Figure 6.5: Actual and simulated gyrification from P8 to P16. (Left) The top row shows the development of the major sulci of the ferret brain [122,123,99,4,49] from P8 to P16. The middle row shows a numerical model of a P8 brain and its deformed state mimicking progression to P16. The bottom row shows a physical gel model of P8 and its post-swelling state mimicking progression to P16. The P8 initial states have invaginations corresponding to the cruciate sulcus (crs), coronal sulcus (cns) and suprasylvian sulcus (sss), and both the numerical deformed state and the physical post-swelling state have sulci corresponding in location and self-contacting nature to the crs, cns, sss, lateral sulcus (ls), and ansate sulcus (as) observed in P16 real ferrets. (Right) The real and simulated P16 brains and their spherical parameterizations.

quantitative comparison between the gyrification of the numerical simulation of ferret brain development and the real data, we evaluate the curvature difference using the spherical mappings produced by the FLASH method [27] (Figure 6.5, right). After obtaining the spherical parameterizations with landmarks optimally aligned, we compare the folding patterns of the two brain surfaces by evaluating their curvature-based intensity difference via the two spherical domains. More specifically, we compute the average intensity difference between the two spherical domains with a normalization such that the difference d is always between 0 and 1. For the simulated P16 brain and real P16 brain , the intensity difference between their spherical parametrizations is $d \approx 0.1$, which suggests that the folding pattern produced by our simulation is similar to the actual

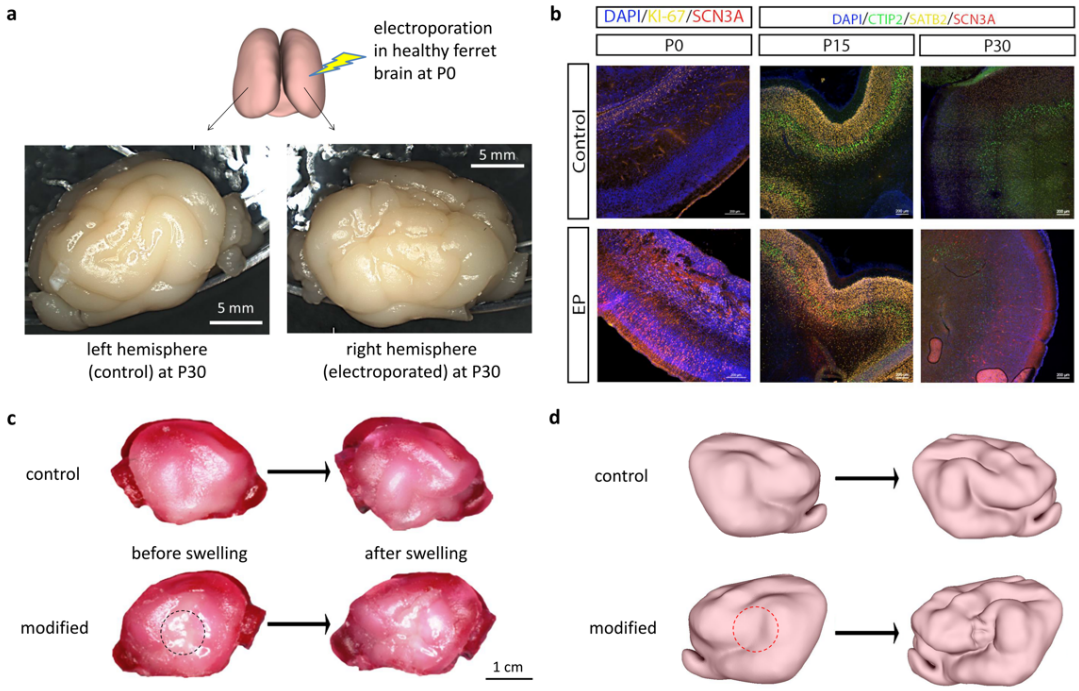

Figure 6.6: Modeling structural abnormalities associated with a pathogenic *SCN3A* gene. a, *In vivo* electroporation of pathogenic *SCN3A* in healthy ferret brain at P0. Left (control) and right (electroporated) hemispheres of the ferret brain at P30. **b**, Immunofluorescent analysis of cortical layer markers at P0, P15, and P30. **c**-**d**, Gel brain and simulated brain with localized modification of cortical layer. Modification of the cortical layer thickness and growth rate is found to induce variant folding patterns in a localized region. Dashed circles indicate modified region.

folding pattern.

6.3 Ferret cortical malformations

To study whether our model based on differential growth can capture malfunction in ferret brains, we consider the structural abnormalities associated with the MCD causing gene *SCN3A* [125].

Molecular analysis of cell-type and cortex layer specific markers provided a pathological basis of the structural malformations, informing physical and simulacrum models (Figure 6.6**b**). By modifying

the thickness and the growth rate at a localized region in the gel brain model and the numerical brain model, we see that we can qualitatively capture the variations seen in the real brain in both the gel model and in computations (Figure 6.6**c**,**d**).

6.4 Discussion

We have identified developmental mechanisms and morphological manifestations associated with genetic mutations in ferret models, validated extrapolation from these transgenic models to human neurodevelopment, and improved the comprehensiveness of the MCD classification scheme for facilitating more efficient and precise diagnosis and treatment. A natural next step is to move towards human brain and use the model to study cortical malformations.

It is through science that we prove, but through intuition

that we discover.

Henri Poincaré

7

Density-equalizing maps for surfaces

Surface parameterization, the process of mapping a complicated surface to a simpler domain, is a problem in computer graphics closely related to cartogram production. In recent decades, three-dimensional (3D) graphics have become widespread in the computer industry. To create realistic textures on 3D shapes, it is common to parameterize the 3D shapes onto $\mathbb{R}^2$, design the

textures on the plane, and map them back onto the 3D shapes. Analogous to the map-making problem, the surface parameterization problem involves distorting the shapes to a certain extent, determined by a prescribed criterion. Two major classes of surface parameterization algorithms are conformal (angle-preserving) parameterization [67,148,27,28] and authalic (area-preserving) parameterization [153,151,132] (see Refs. [48,118,64] for a discussion on the prior parameterization algorithms). In this chapter, we develop a new surface parameterization method inspired by the diffusion-based map-making method by Gastner and Newman (denoted as *GN*) [52]. Specifically, we propose a finite-element algorithm for computing density-equalizing flattening maps of simply-connected open surfaces in $\mathbb{R}^3$ onto the plane, based on certain quantities prescribed at every part of the surface. Different surface flattening effects can then be produced by altering the input quantities. For instance, area-preserving parameterizations can be easily achieved.

7.1 Diffusion-based cartogram

Given a planar map and a quantity called the *population* defined on every part of the map, define the density field ρ by the quantity per unit area. GN deforms the map by equalizing ρ using the advection equation

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot \vec{j}. \tag{7.1}$$

Here, the flux is given by Fick's law:

$$\vec{j} = -\nabla \rho, \tag{7.2}$$

yielding the diffusion equation

$$\frac{\partial \rho}{\partial t} = \Delta \rho. \tag{7.3}$$

Then, any tracers carried by this density flux will move with velocity

$$\vec{v}(\vec{r}, t) = \frac{\vec{j}}{\rho} = -\frac{\nabla \rho}{\rho}. \tag{7.4}$$

Hence, the deformation of the map can be tracked using the tracers $\vec{r}(t)$ that follow the velocity field:

$$\vec{r}(t) = \vec{r}(0) + \int_0^t \vec{v}(\vec{r}, \tau) d\tau. \tag{7.5}$$

As $t \to \infty$, Eq. (7.3) is solved to steady state, and the corresponding deformed map produced by Eq. (7.5) achieves equalized density per unit area. In particular, regions with a higher density expand, while those with a lower density shrink (Figure 7.1). As a remark, GN makes use of a large rectangular auxiliary region surrounding the region of interest, called the *sea*, to avoid infinite expansion of the map. By defining the density at the sea as the average density of the region of interest, one can ensure that the total area of the map is kept constant under the deformation.

7.2 Surface density-equalizing maps

In the following, we develop a method for surface parameterization based on the idea of density diffusion as described above. Let S be a simply-connected open surface in $\mathbb{R}^3$, and ρ be a prescribed density distribution. Our method computes a flattening map $f : S \to \mathbb{R}^2$ such that the Jacobian J_f

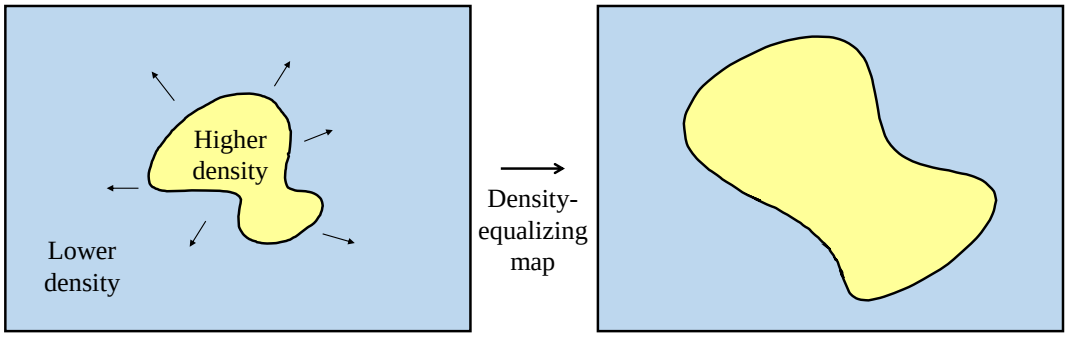

Figure 7.1: An illustration of the density-equalizing maps. Regions with a higher density expand, while those with a lower density shrink.

satisfies $J_f \propto \rho$. Equivalently, the final density per unit area in the flattening map is a constant. Our method primarily consists of three steps, including the creation of an initial flattening map (in case the input surface is non-planar), the construction of the sea, and an iterative finite-element scheme for computing density-equalizing maps.

In the following discussion, S is discretized as a triangle mesh $(\mathcal{V}, \mathcal{E}, \mathcal{F})$ where $\mathcal{V}$ is the set of all vertices, $\mathcal{E}$ is the set of all edges, and $\mathcal{F}$ is the set of all triangular faces. The input density distribution ρ is discretized as $\rho^{\mathcal{F}}$ on every triangle element $T \subset \mathcal{F}$.

7.2.1 Curvature-based initial flattening map

We start by developing a method that flattens S onto $\mathbb{R}^2$ efficiently. To keep the flattened shape as close to S as possible, it is natural to consider flattening the surface boundary onto the plane based on its curvature.

Curvature-based flattening of the surface boundary

Let γ be the boundary of S. As γ is a simple closed curve in $\mathbb{R}^3$, it can be written 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 find a map $\phi : [0, l_\gamma] \to \mathbb{R}^2$ to flatten γ onto $\mathbb{R}^2$, and then obtain the entire flattening map for S. Two important geometric quantities for γ are the curvature κ_γ and the torsion τ_γ, which respectively measure the deviation of γ from a straight line and from a planar curve. By the fundamental theorem of space curves [37], γ is completely determined by κ_γ and τ_γ up to rigid motion.

Now, we consider a map ϕ such that $\kappa_\gamma \approx \kappa_{\phi(\gamma)}$ and $\tau_{\phi(\gamma)} = 0$. In other words, ϕ projects γ onto the space of planar convex curves, with the curvature of γ preserved as much as possible. By Frenet–Serret formulas [37],

$$\vec{T}'(t) = \kappa_\gamma(t)\|\gamma'(t)\|\vec{N}(t) \tag{7.6}$$

where $\vec{T}$ and $\vec{N}$ are the unit tangent and unit normal of γ respectively. We have

$$\kappa_\gamma(t) = \frac{\|\vec{T}'(t)\|}{\|\gamma'(t)\|}. \tag{7.7}$$

Note that for any simple closed plane curve $\mathcal{C} \subset \mathbb{R}^2$, the total signed curvature of $\mathcal{C}$ is a constant [37]:

$$\int_{\mathcal{C}} k_{\mathcal{C}}(t)dt = 2\pi. \tag{7.8}$$

To construct a closed plane curve $\phi(\gamma)$ with total arclength l_γ, we set the target signed curvature k to

be

$$k(s) = \frac{2\pi\kappa_\gamma(s)}{\int_\gamma \kappa_\gamma(t)dt} \geq 0. \tag{7.9}$$

Then, we consider the map

$$\phi(s) = \left(\int_0^s \cos\theta(u)du, \int_0^s \sin\theta(u)du \right), \tag{7.10}$$

where

$$\theta(u) = \int_0^u k(t)dt. \tag{7.11}$$

We can easily check that

$$\phi'(s) = (\cos\theta(s), \sin\theta(s)), \tag{7.12}$$

and hence ϕ is an arc-length parameterized curve. Moreover, we have

$$k_\phi(s) = \theta'(s) = k(s) \geq 0. \tag{7.13}$$

However, ϕ may not be a closed curve as there may be a discrepancy between $\phi(0)$ and $\phi(l_\gamma)$ with $0 \leq \|\phi(l_\gamma) - \phi(0)\| \ll L$, where L is the total arclength of ϕ. To fix this issue, we update ϕ by

$$\phi(s) \leftarrow \phi(s) - \frac{s}{l_\gamma}\left(\phi(l_\gamma) - \phi(0)\right). \tag{7.14}$$

Then, ϕ will be a simple closed convex plane curve.

Proposition 7.1. *Let $\phi : [0, l_\gamma] \to \mathbb{R}^2$ be the arc-length parameterized curve defined above. If we define a new curve $\Phi : [0, l_\gamma] \to \mathbb{R}^2$ by*

$$\Phi(s) = \phi(s) - \frac{s}{l_\gamma}\left(\phi(l_\gamma) - \phi(0)\right). \tag{7.15}$$

Then, Φ is a simple closed convex plane curve.

Proof. First, note that $\Phi(0) = \phi(0) = \Phi(l_\gamma)$ and hence Φ is closed. Since ϕ is arc-length parameterized, for any $0 \leq a < b \leq l_\gamma$, we have

$$\|\phi(b) - \phi(a)\| \leq \int_a^b \|\phi'(s)\| ds = b - a. \tag{7.16}$$

The equality holds if and only if $\phi([a, b])$ is a straight line. In particular, as γ is the boundary of S, by our construction of ϕ we have $\|\phi(l_\gamma) - \phi(0)\| \ll l_\gamma$.

Now, we prove that the signed curvature of Φ, denoted by k_Φ, is non-negative for all $s \in [0, l_\gamma]$. Denote $\phi(s) = (x(s), y(s))$ and $\Phi(s) = (X(s), Y(s))$. We have

$$\begin{aligned}
\Phi' &= (X', Y') \\
&= \left(x'(s) - \frac{1}{l_\gamma}\left(x(l_\gamma) - x(0)\right), y'(s) - \frac{1}{l_\gamma}\left(y(l_\gamma) - y(0)\right)\right) \\
&= \left(\cos\theta(s) - \frac{1}{l_\gamma}\left(x(l_\gamma) - x(0)\right), \sin\theta(s) - \frac{1}{l_\gamma}\left(y(l_\gamma) - y(0)\right)\right)
\end{aligned} \tag{7.17}$$

and

$$\Phi'' = (X'', Y'') = (x'', y'') = \left(-k_\phi(s)\sin\theta(s), k_\phi(s)\cos\theta(s)\right). \tag{7.18}$$

Therefore, we have

$$X'Y'' - X''Y' = k_\phi(s)\left(1 - \frac{\big(x(l_\gamma) - x(0)\big)\cos\theta(s) - \big(y(l_\gamma) - y(0)\big)\sin\theta(s)}{l_\gamma}\right). \tag{7.19}$$

Recall that by Eq. (7.9), $k_\phi(s) \geq 0$ for all s. Also, we have

$$\begin{aligned}
&\big(x(l_\gamma) - x(0)\big)\cos\theta(s) - \big(y(l_\gamma) - y(0)\big)\sin\theta(s) \\
\leq &\sqrt{\big(x(l_\gamma) - x(0)\big)^2 + \big(y(l_\gamma) - y(0)\big)^2}\sqrt{\cos^2\theta(s) + \sin^2\theta(s)} \\
= &\sqrt{\big(x(l_\gamma) - x(0)\big)^2 + \big(y(l_\gamma) - y(0)\big)^2} \\
= &\big\|\phi(l_\gamma) - \phi(0)\big\| \\
\leq &l_\gamma,
\end{aligned} \tag{7.20}$$

where the first inequality follows from the Cauchy–Schwarz inequality, and the second inequality follows from Eq. (7.16). Hence, we have

$$1 - \frac{\big(x(l_\gamma) - x(0)\big)\cos\theta(s) - \big(y(l_\gamma) - y(0)\big)\sin\theta(s)}{l_\gamma} \geq 0 \text{ for all } s \in [0, l_\gamma], \tag{7.21}$$

yielding

$$k_\Phi(s) = \frac{X'Y'' - X''Y'}{(X'^2 + Y'^2)^{3/2}} \geq 0 \text{ for all } s \in [0, l_\gamma]. \tag{7.22}$$

To show that Φ is simple, note that as Φ is a closed plane curve, the total curvature of Φ should

satisfy

$$\int_0^{l_\phi} k_\Phi(s)ds = 2\pi n_\Phi, \tag{7.23}$$

where n_Φ is the turning number of Φ. It follows from the above results that

$$\begin{aligned} 2\pi n_\Phi &= \int_0^{l_\phi} \frac{k_\phi(s)\left(1 - \frac{\left(x(l_\gamma)-x(0)\right)\cos\theta(s) - \left(y(l_\gamma)-y(0)\right)\sin\theta(s)}{l_\gamma}\right)}{(X'^2+Y'^2)^{3/2}}ds \\ &\leq \left(\int_0^{l_\phi} k_\phi(s)ds\right) \max_{s\in[0,l_\gamma]} \left| \frac{\left(1 - \frac{\left(x(l_\gamma)-x(0)\right)\cos\theta(s) - \left(y(l_\gamma)-y(0)\right)\sin\theta(s)}{l_\gamma}\right)}{(X'^2+Y'^2)^{3/2}} \right| \\ &= 2\pi \max_{s\in[0,l_\gamma]} \left(\frac{1 - A\cos\theta(s) + B\sin\theta(s)}{\left(1 + A^2 + B^2 - 2A\cos\theta(s) - 2B\sin\theta(s)\right)^{3/2}} \right), \end{aligned} \tag{7.24}$$

where $A = \frac{x(l_\gamma)-x(0)}{l_\gamma}$ and $B = \frac{y(l_\gamma)-y(0)}{l_\gamma}$. We can further simplify the above equation as

$$2\pi \max_{s\in[0,l_\gamma]} \left(\frac{1 - C\cos(\theta(s)+\eta)}{\left(1 + C^2 - 2C\cos(\theta(s)-\eta)\right)^{3/2}} \right), \tag{7.25}$$

where $C = \sqrt{A^2+B^2} = \frac{\left\|\phi(l_\gamma)-\phi(0)\right\|}{l_\gamma} \ll 1$ and $\eta = \tan^{-1}\frac{B}{A} = \tan^{-1}\frac{y(l_\gamma)-y(0)}{x(l_\gamma)-x(0)}$. It is therefore easy to see that

$$\max_{s\in[0,l_\gamma]} \left(\frac{1 - C\cos(\theta(s)+\eta)}{\left(1 + C^2 - 2C\cos(\theta(s)-\eta)\right)^{3/2}} \right) \leq \frac{1+C}{(1-C)^3} < 2, \tag{7.26}$$

which implies that

$$2\pi n_\Phi < 4\pi \Rightarrow n_\Phi < 2. \tag{7.27}$$

Therefore, $n_\Phi = 1$ and Φ is simple. As a simple closed curve is convex if and only if its signed

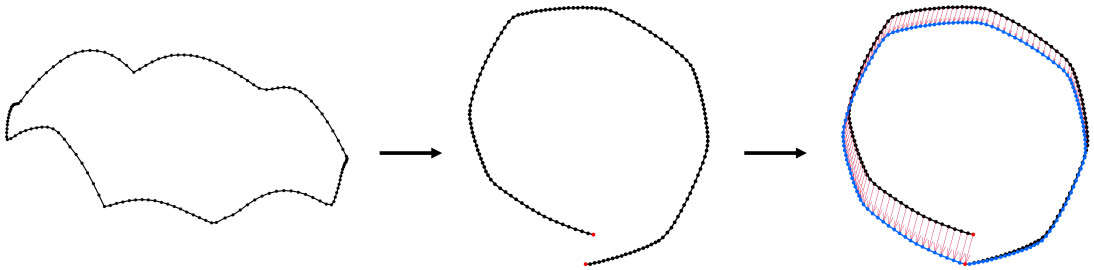

Figure 7.2: The proposed curvature-based curve flattening procedure. Given a closed curve representing the surface boundary, we first construct a plane curve using Eq. (7.10). Then, we use Eq. (7.14) to obtain a simple closed convex curve.

curvature does not change sign, the result follows from Eq. (7.22). ■

An illustration of the curve flattening procedure is provided in Figure 7.2. The algorithm is summarized in Algorithm 7.1.

Algorithm 7.1: Curvature-based curve flattening

Input: The boundary γ of a simply-connected open surface S in $\mathbb{R}^3$.
Output: A simple closed convex plane curve ϕ.

1. Let $\gamma = \{v_j\}_{j=1}^{b}$ be the boundary vertices of S in anti-clockwise order. Compute the curvature $\kappa = \frac{\|\vec{T}'\|}{\|\gamma'\|}$;
2. Rescale κ using Eq. (7.9);
3. Obtain a plane curve $\phi(s)$ by solving Eq. (7.10);
4. Adjust $\phi(s)$ using Eq. (7.14) to obtain a simple closed convex curve;

After obtaining ϕ, we use it as a boundary constraint and compute a flattening map $\varphi : S \to \mathbb{R}^2$ for the entire surface S as an initialization. Two methods for obtaining φ are discussed below.

Curvature-based Tutte flattening map

The graph embedding method proposed by Tutte [140],which has been widely used as an initialization for surface parameterization [60,124], is capable of producing a bijective planar map. One possible method for obtaining φ is to combine the Tutte mapping method with our curvature-based curve flattening result.

The *adjacency matrix* M is a $|\mathcal{V}| \times |\mathcal{V}|$ matrix with

$$M_{ij} = \begin{cases} 1 & \text{if } [i,j] \in \mathcal{E}, \\ 0 & \text{otherwise.} \end{cases} \tag{7.28}$$

Remarkably, there exists a bijective map φ between any simply-connected open triangulated surface S in $\mathbb{R}^3$ and any convex polygon P on $\mathbb{C}$ [140]. More explicitly, by representing φ as a $|\mathcal{V}| \times 1$ column vector with complex entries, we can obtain φ by solving the complex linear system

$$\begin{cases} M^{\text{Tutte}}\varphi(v) = 0 & \text{if } v \in S \setminus \partial S, \\ \varphi(\partial S) = \partial P, \end{cases} \tag{7.29}$$

where

$$M_{ij}^{\text{Tutte}} = \begin{cases} M_{ij} & \text{if } [x_i, x_j] \in \mathcal{E}, \\ -\sum_{t \neq i} M_{it} & \text{if } j = i, \\ 0 & \text{otherwise.} \end{cases} \tag{7.30}$$

Here, the boundary correspondence $\varphi(\partial S) = \partial P$ can be any prescribed bijective map. Now, since

the curvature-based curve flattening map ϕ resembles the shape of the surface boundary well, we use ϕ as the convex boundary constraint and obtain a bijective Tutte flattening map $\varphi : S \to P$. The procedure is summarized in Algorithm 7.2.

Algorithm 7.2: Curvature-based Tutte flattening map

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

1 Let $\gamma = \{v_j\}_{j=1}^{b}$ be the boundary vertices of S. Compute the curvature-based curve flattening map $\phi : \gamma \to \mathbb{C}$ using Algorithm 7.1;
2 Compute the Tutte matrix M^{Tutte};
3 Obtain the desired map φ by solving the linear system 7.29, with the boundary constraint given by ϕ;

Curvature-based locally authalic flattening map

Another way to obtain a surface flattening map φ is to use a variant of the matrix M^{Tutte}. Desbrun et al.[35] proposed a mapping scheme that minimizes the following quadratic Chi energy

$$E_\chi(\varphi) = \sum_{j \in N(i)} \frac{\cot \gamma_{ij} + \cot \delta_{ij}}{|x_i - x_j|^2} |\varphi(x_i) - \varphi(x_j)|^2, \tag{7.31}$$

where γ_{ij} and δ_{ij} are the two angles at x_j as illustrated in Figure 7.3. This gives a locally authalic map $\varphi : S \to \mathbb{R}^2$ that preserves the local 1-ring area at every vertex as much as possible. The associated

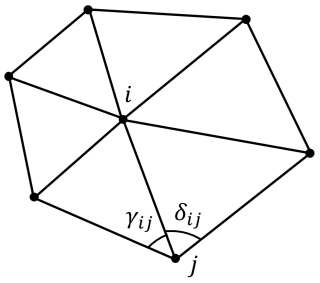

Figure 7.3: The angles γ_{ij} and δ_{ij} in the authalic matrix.

authalic matrix is given by

$$M^{\chi}_{ij} = \begin{cases} \frac{\cot \gamma_{ij} + \cot \delta_{ij}}{|x_i - x_j|^2} & \text{if } [x_i, x_j] \in \mathcal{E}, \\ -\sum_{t \neq i} M^{\chi}_{it} & \text{if } j = i, \\ 0 & \text{otherwise.} \end{cases} \tag{7.32}$$

Now, we replace M^{Tutte} in Algorithm 7.2 with the authalic matrix M^{χ} and solve for a new surface flattening map. More explicitly, we obtain φ by solving the following complex linear system with our curvature-based boundary constraint:

$$\begin{cases} M^{\chi}\varphi(v) = 0 & \text{if } v \in S \setminus \partial S, \\ \varphi(\partial S) = \phi. \end{cases} \tag{7.33}$$

While the minimizer of the Chi energy is not a globally optimal area-preserving map, this approach provides a reasonably good initialization with the local area taken into account. However, it is noteworthy that unlike the Tutte map, the bijectivity of the locally authalic map is only guaranteed when the input mesh satisfies the convex combination mapping property, i.e. all $\cot \gamma_{ij} + \cot \delta_{ij}$ in

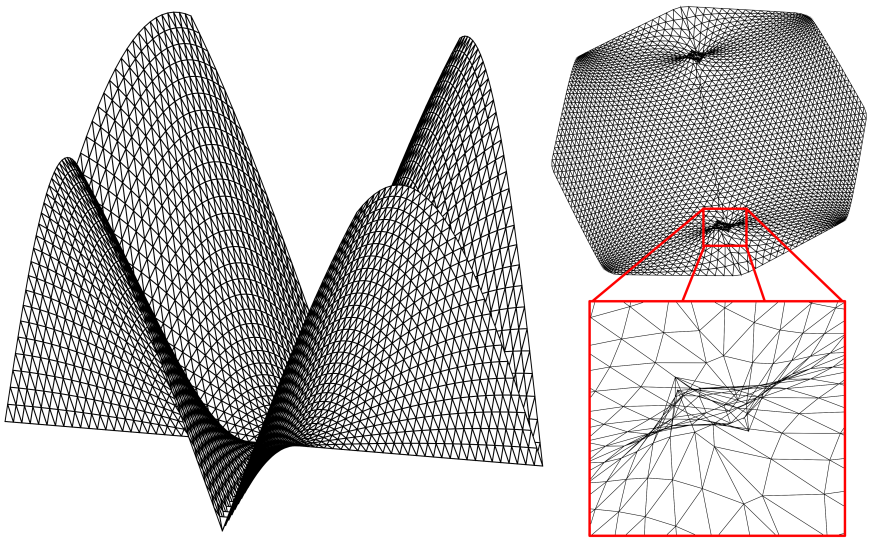

Figure 7.4: An example for which the locally authalic map contains overlaps. Left: A mesh that contains many sliver triangles and violates the convex combination mapping property[48]. Right: The curvature-based locally authalic flattening map, with triangle overlaps observed.

$\mathcal{M}^{\chi}$ are nonnegative[48], which is equivalent to that $\gamma_{ij} + \delta_{ij} \leq \pi$ for all i, j. Figure 7.4 shows a mesh violating this condition, and it can be observed that the locally authalic flattening map contains mesh fold-overs. In this case, we can simply resort to the Tutte flattening map for ensuring the bijectivity. The method is summarized in Algorithm 7.3.

Algorithm 7.3: 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 $\varphi : S \to \mathbb{R}^2$.

1. Let $\gamma = \{v_j\}_{j=1}^{b}$ be the boundary vertices of S. Compute the curvature-based curve flattening $\phi : \gamma \to \mathbb{C}$;
2. Compute the authalic matrix $\mathcal{M}^{\chi}$;
3. Obtain the desired map φ by solving the linear system 7.33, with the boundary constraint given by ϕ;
4. In case φ contains overlaps, resort to the Tutte map in Algorithm 7.2;

7.2.2 Construction of sea via reflection

One important step in GN is to construct a sea surrounding the region of interest so that it can deform freely. Here, we propose a new method for the construction of such a sea. If the simply-connected open surface S is non-planar, then the curvature-based flattening methods give us an initial flattening map $\vec{r}_0 = \varphi(S)$ in $\mathbb{R}^2$. If S is planar, we can simply skip the above step and treat S itself as the initial flattening map, i.e. setting $\vec{r}_0 = S$.

Now, we rescale the initial flattening map $\vec{r}_0$ and put it inside the unit circle $\mathbb{S}^1 := \{z \in \mathbb{C} : |z| = 1\}$. Denote the edge length of the shrunk flattening map by l. Then, we fill up the gaps between the rescaled initial flattening map and the circular boundary using uniformly distributed points with distance l. This gives us a set of evenly distributed points over the unit disk $\mathbb{D} := \{z \in \mathbb{C} : |z| \leq 1\}$. We triangulate the new points using the Delaunay triangulation and obtain a triangulation $\mathbb{D}_T$.

To construct a sea surrounding the unit disk, we consider the reflection mapping $g : \mathbb{D} \to \mathbb{C} \setminus \mathbb{D}$ defined by

$$g(z) = \frac{1}{\bar{z}}. \tag{7.34}$$

It is easy to see that g is bijective. In the discrete case, g maps the triangulated unit disk $\mathbb{D}_T$ to a large polygonal region $R \; subset \mathbb{C}$ with the unit disk $\mathbb{D}$ punctured. We can then glue $\mathbb{D}_T$ and $g(\mathbb{D}_T)$ along

the circular boundary $\partial\mathbb{D}_T$. Denote the glued mesh by $\widetilde{S} = (\widetilde{\mathcal{V}}, \widetilde{\mathcal{E}}, \widetilde{\mathcal{F}})$. We have

$$\widetilde{\mathcal{V}} = \{z\}_{z\in\mathbb{D}_T} \cup \left\{\frac{1}{\bar{z}}\right\}_{z\in\mathbb{D}_T\setminus\partial(\mathbb{D}_T)}, \tag{7.35}$$

$$\widetilde{\mathcal{F}} = \mathcal{F} \cup \left\{\left[\frac{1}{\bar{z}_i}, \frac{1}{\bar{z}_j}, \frac{1}{\bar{z}_k}\right] : [z_i, z_j, z_k] \in \mathcal{F}\right\}, \tag{7.36}$$

and

$$\widetilde{\mathcal{E}} = \{[z_i, z_j] : [z_i, z_j] \text{ is an edge of a face } T \in \widetilde{\mathcal{F}}\}. \tag{7.37}$$

In practice, the extremely large triangles at the outermost part of the glued mesh produced by the reflection may cause numerical instability in the subsequent computations. Therefore, we remove them by a simple truncation of the part far away from $\mathbb{D}$. More specifically, we remove all vertices and faces of $\widetilde{S}$ outside $\{z : |z| > \eta\}$, where η is a thresholding parameter. According to GN, having a sea with dimensions a few times the linear extent of the domain of interest is sufficient. Therefore, we set $\eta = 5$. Finally, the glued mesh is rescaled in order to restore the size of the flattening map. By an abuse of notation, we use $\vec{r}_0$ to represent the entire region. Algorithm 7.4 summarizes the above procedures. An graphical illustration is given in Figure 7.5.

Our construction is advantageous in the sense that the mesh size of the resulting sea is adaptive. Unlike the approach in GN, which used a uniform finite difference grid for constructing the sea, our approach produces a natural distribution of points at the sea that avoids redundant computation. To see this, let z_1, z_2 be two points at the interior of $\mathbb{D}$. Under the reflection mapping $z \mapsto \frac{1}{\bar{z}}$, we

Algorithm 7.4: Construction of sea via reflection.

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

1 Rescale $\vec{r}_0$ to lie inside the unit circle $\mathbb{S}^1$;
2 Fill up the gaps between the unit circle and the rescaled map by uniformly distributed points with spacing l, where l is the average edge length of $\vec{r}_0$;
3 Perform a constrained Delaunay triangulation that triangulates the unit disk with the newly added points, with the connectivity of $\vec{r}_0$ 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 $g(\mathbb{D}_T)$, and update $\vec{r}_0$ by the glued result;
6 Remove all vertices and faces of $\vec{r}_0$ outside $\{z : |z| > \eta\}$;
7 Rescale $\vec{r}_0$ to restore the size of the flattening map;

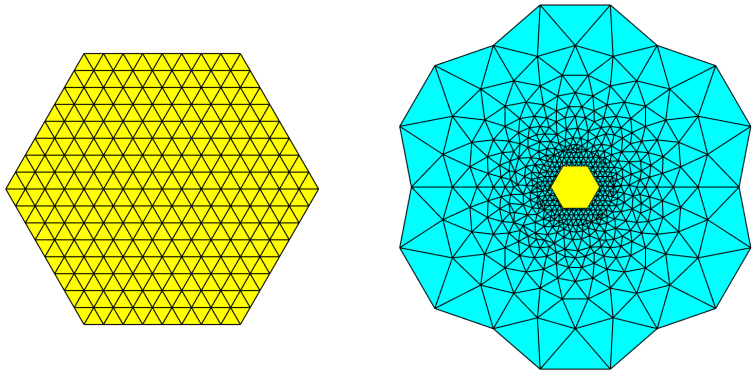

Figure 7.5: An illustration of our algorithm for constructing the sea. Left: the initial flattening map. We rescale it to lie inside the unit circle, and fill up the gaps with uniformly distributed points. Right: the sea constructed via the reflection mapping (in cyan) and the initial flattening map (in yellow).

have

$$\left|\frac{1}{\overline{z_1}} - \frac{1}{\overline{z_2}}\right| = \frac{|\overline{z_1} - \overline{z_2}|}{|\overline{z_1 z_2}|} = \frac{|z_1 - z_2|}{|z_1 z_2|}. \tag{7.38}$$

Therefore, the edges formed under the reflection are shorter near the unit disk boundary, and get longer further away. The outermost part of the sea, which stays far away from $\mathbb{D}$, then consists of the coarsest triangulations. By contrast, the innermost part of the sea closest to $\mathbb{D}$ consists of the

densest triangulations. This natural transition of mesh sparsity reduces the number of points needed for the subsequent computation without affecting the numerical accuracy.

Another advantage of our approach is about the improvement of the shape of the sea. In GN, a rectangular sea is used. The four corners are usually unimportant for the subsequent deformation, and hence computational efforts are wasted there. By contrast, in our reflection-based construction, the reflection together with the truncation produces a sea with a more regular shape, thereby utilizing the use of every part of the sea and reducing redundant computations.

7.2.3 Iterative scheme for producing density-equalizing maps

Suppose we are given a population on each triangle element of the input surface mesh. Define the density $\rho^{\mathcal{F}}$ on each triangle element of the flattened map ϕ by $\frac{\text{Population}}{\text{Area of the triangle}}$. We interpolate $\rho^{\mathcal{F}}$ on the vertices and develop an iterative scheme for deforming the flattened map based on density diffusion.

To solve the diffusion equation on triangle meshes, it is important to discretize the Laplacian. Let $u : \widetilde{\mathcal{V}} \to \mathbb{R}$ be a function. To compute the Laplacian of u at every vertex i, we use the following discrete finite-element Laplacian [112]

$$\Delta u(i) = -\frac{1}{2A(i)} \sum_{j \in \mathcal{N}(i)} \left(\cot \alpha_{ij} + \cot \beta_{ij}\right) (u(i) - u(j)), \tag{7.39}$$

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

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

$$A(i) = \frac{1}{3} \sum_{T \in \mathcal{N}^{\widetilde{\mathcal{F}}}(i)} \text{Area}(T), \tag{7.40}$$

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

$$\sum_{i \in \widetilde{\mathcal{V}}} A(i) = \sum_{T \in \widetilde{\mathcal{F}}} \text{Area}(T). \tag{7.41}$$

Therefore, the vertex area is a good discretization of the total surface area at the vertices. We remark that the sea we constructed has an additional purpose of complementing the use of the FEM Laplacian (7.39), which assumes the natural boundary condition $\nabla u \cdot \mathbf{n} = 0$ in its derivation [112]. The region of interest is not restricted by the above natural boundary condition and hence can deform freely.

Note that the density ρ is originally defined on the triangle faces, while the above Laplacian is defined at the vertices. To interpolate $\rho^{\mathcal{F}}$ at the vertices, we note that for every vertex $v \in \mathcal{V}$, $\rho^{\mathcal{V}}(v)$ should only depend on the value of $\rho^{\mathcal{F}}$ within its one-ring face neighborhood. This property is related to the Whitney forms [147], which were originally introduced for algebraic topology and subsequently used as finite elements [13]. The Whitney 2-forms are piecewise-constant functions

supported on triangle elements[33]:

$$\varphi_T^W(x) = \begin{cases} \frac{1}{\text{Area}(T)} & \text{if } x \text{ lies on } T, \\ 0 & \text{otherwise.} \end{cases} \tag{7.42}$$

Now, any face-valued function $f : \mathcal{F} \to \mathbb{R}$ can be interpolated at all vertices $v \in \mathcal{V}$ by

$$f(v) = \frac{\sum_{T\in\mathcal{F}} \int_T f(T)\varphi_T^W(v)dA}{\sum_{T\in\mathcal{F}} \int_T \varphi_T^W(v)dA} = \frac{\sum_{T\in\mathcal{N}^{\mathcal{F}}(v)} \frac{f(T)}{\text{Area}(T)}\text{Area}(T)}{\sum_{T\in\mathcal{N}^{\mathcal{F}}(v)} \frac{1}{\text{Area}(T)}\text{Area}(T)} = \frac{\sum_{T\in\mathcal{N}^{\mathcal{F}}(v)} f(T)}{|\mathcal{N}^{\mathcal{F}}(v)|}. \tag{7.43}$$

It is noteworthy that the above interpolation only depends on the connectivity but not the geometry of the mesh. In our case, since ρ is related to the deformation of the face area, it is desirable to emphasize on the weight of different faces in the interpolation. Therefore, we consider the following modified Whitney 2-forms

$$\widetilde{\varphi_T^W}(x) = \begin{cases} 1 & \text{if } x \text{ lies on } T, \\ 0 & \text{otherwise,} \end{cases} \tag{7.44}$$

which gives the desired interpolation

$$\rho^{\mathcal{V}}(v) = \frac{\sum_{T\in\mathcal{F}} \int_T \rho^{\mathcal{F}}(T)\widetilde{\varphi_T^W}(v)dA}{\sum_{T\in\mathcal{F}} \int_T \widetilde{\varphi_T^W}(v)dA} = \frac{\sum_{T\in\mathcal{N}^{\mathcal{F}}(v)} \rho^{\mathcal{F}}(T)\text{Area}(T)}{\sum_{T\in\mathcal{N}^{\mathcal{F}}(v)} \text{Area}(T)}. \tag{7.45}$$

By treating $\rho^{\mathcal{V}}$ as a $|\mathcal{V}| \times 1$ matrix and $\rho^{\mathcal{F}}$ as a $|\mathcal{F}| \times 1$ matrix, we can represent the above formula

as a matrix multiplication

$$\rho^{\mathcal{V}} := W^{\mathcal{F}\mathcal{V}}\rho^{\mathcal{F}}, \tag{7.46}$$

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

$$W^{\mathcal{F}\mathcal{V}} := \begin{pmatrix} W_{1,:}/\|W_{1,:}\|_1 \\ W_{2,:}/\|W_{2,:}\|_1 \\ \vdots \\ W_{|\mathcal{V}|,:}/\|W_{|\mathcal{V}|,:}\|_1 \end{pmatrix}. \tag{7.47}$$

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

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

$$\rho^{\mathcal{V}}(v') = \text{mean}_v \rho^{\mathcal{V}}(v). \tag{7.48}$$

We remark that the density at the boundary of ϕ may be affected by the density at the sea if we perform the interpolation with the sea included. Hence, it is important to perform the interpolation and obtain $\rho^{\mathcal{V}}$ at ϕ first, and then set the density at the sea. By an abuse of notation, we continue using $\mathcal{V}$, $\mathcal{F}$ without tilde in the following discussions whenever referring to the discrete mesh

structure with the sea included.

With the density interpolated and set at all vertices, we use the following semi-discrete backward Euler scheme to solve the diffusion equation (7.3):

$$\frac{\rho_n^{\mathcal{V}} - \rho_{n-1}^{\mathcal{V}}}{\delta t} = \Delta_{n-1}\rho_n^{\mathcal{V}} \Longleftrightarrow \rho_n^{\mathcal{V}} = (I - \delta t \Delta_{n-1})^{-1}\rho_{n-1}^{\mathcal{V}}, \tag{7.49}$$

where $\rho_n^{\mathcal{V}}$ is the value of $\rho^{\mathcal{V}}$ at the n-th iteration, Δ_n is the FEM Laplacian of the deformed map $\vec{r}_n$, and δt is the timestep. Note that Δ_n can be represented as $\Delta_n = -A_n^{-1}L_n$, where A_n is a $|\mathcal{V}| \times |\mathcal{V}|$ diagonal matrix containing the vertex area of each vertex, and L_n is a $|\mathcal{V}| \times |\mathcal{V}|$ sparse symmetric matrix containing the cotangent components in Eq. (7.39). Therefore, we can rewrite the above equation as

$$\rho_n^{\mathcal{V}} = (A_{n-1} + \delta t L_{n-1})^{-1} \left(A_{n-1}\rho_{n-1}^{\mathcal{V}}\right). \tag{7.50}$$

The above semi-discrete backward Euler scheme is unconditionally stable. Also, the matrix $A_{n-1} + \delta t L_{n-1}$ is a symmetric sparse matrix and hence Eq. (7.50) can be efficiently solved.

After discretizing the diffusion equation, we consider the induced vector field. To discretize the gradient operator ∇, let $(\nabla\rho)_n^{\mathcal{F}}(T)$ be the face-based discretization defined on every triangle element $T = [i, j, k]$ at the n-th iteration. Denote the three directed edges of T in the form of vectors by $e_{ij} = [i, j], e_{jk} = [j, k], e_{ki} = [k, i]$, and the unit normal vector of T by N. We can then derive a

formula for $(\nabla\rho)_n^{\mathcal{F}}(T)$ using the Whitney 0-forms [33], which are hat functions at the vertices:

$$\varphi_i^W(p) = \begin{cases} 1 & \text{if } p = i, \\ 0 & \text{if } p \text{ is outside } N^{\mathcal{F}}(i), \\ \text{affine} & \text{if } p \text{ lies on } N^{\mathcal{F}}(i). \end{cases} \tag{7.51}$$

Give any vertex-based function f, we can interpolate f at any point x lying on the triangle face $T = [i, j, k]$ by

$$f(x) = f_i\varphi_i^W(x) + f_j\varphi_j^W(x) + f_k\varphi_k^W(x), \tag{7.52}$$

where f_i, f_j, f_k are the values of f at the vertices i, j, k. Using the property that $\nabla\varphi_i^W = \frac{N \times e_{jk}}{2\text{Area}(T)}$ [33], we have

$$\begin{aligned} (\nabla\rho)_n^{\mathcal{F}}(T) &= \nabla\left(\rho_n^{\mathcal{V}}(i)\varphi_i^W + \rho_n^{\mathcal{V}}(j)\varphi_j^W + \rho_n^{\mathcal{V}}(k)\varphi_k^W\right) \\ &= \rho_n^{\mathcal{V}}(i)\nabla\varphi_i^W + \rho_n^{\mathcal{V}}(j)\nabla\varphi_j^W + \rho_n^{\mathcal{V}}(k)\nabla\varphi_k^W \\ &= \frac{1}{2\text{Area}(T)} N \times \left(\rho_n^{\mathcal{V}}(i)e_{jk} + \rho_n^{\mathcal{V}}(j)e_{ki} + \rho_n^{\mathcal{V}}(k)e_{ij}\right). \end{aligned} \tag{7.53}$$

The above formula provides us with an accurate approximation of the gradient $(\nabla\rho)_n^{\mathcal{F}}$ on triangulated surfaces. We can then use the Whitney 2-forms (7.45) to obtain $(\nabla\rho)_n^{\mathcal{V}}$ at the vertices.

With all differential operators discretized, we now introduce our iterative scheme for computing density-equalizing maps. At each iteration, we update the density by solving Eq. (7.50) and compute the induced gradient $(\nabla\rho)_n^{\mathcal{V}}$ using the above-mentioned procedures. Then, we deform the

map by

$$\vec{r}_n = \vec{r}_{n-1} - \delta t(\nabla\rho)_n^{\mathcal{V}} / \rho_n^{\mathcal{V}}. \tag{7.54}$$

For the stopping criterion, note that the standard deviation $\mathrm{sd}(\rho_n^{\mathcal{V}})$ measures the dispersion of the updated density $\rho_n^{\mathcal{V}}$. To remove the effect of scaling of $\rho_n^{\mathcal{V}}$, we divide the standard deviation by $\mathrm{mean}(\rho_n^{\mathcal{V}})$. Also, note that the normalized quantity $\mathrm{sd}(\rho_n^{\mathcal{V}})/\mathrm{mean}(\rho_n^{\mathcal{V}})$ equals 0 if and only if the density is completely equalized. Therefore, this normalized quantity can be used for determining the convergence of the iterative scheme. Finally, we rescale the mapping result so that the total area of S remains unchanged under the algorithm.

For the timestep δt, by dimensional analysis of the diffusion equation (7.3), we can see that an appropriate dimension of δt would be L^2. Also, δt should be independent of the magnitude of ρ. Therefore, we set

$$\delta t = \min\left\{ \frac{\min(\rho_0^{\mathcal{V}})}{\mathrm{mean}(\rho_0^{\mathcal{V}})}, \frac{\mathrm{mean}(\rho_0^{\mathcal{V}})}{\max(\rho_0^{\mathcal{V}})} \right\} \times \mathrm{Area}(S), \tag{7.55}$$

where the first term is a dimensionless quantity taking extreme relative density ratios into account, and the second term is a natural quantity with dimension L^2. One may also rescale δt by a constant. Our proposed method for producing density-equalizing maps (DEM) of simply-connected open surfaces is summarized in Algorithm 7.5.

The proposed algorithm can be easily modified to obtain a flattening map with a prescribed simple boundary shape such as a rectangle or a unit disk. To achieve such a shape-prescribed density-equalizing map, we first replace the initialization (line 1–4) in Algorithm 7.5 with an initial map with the desired boundary shape, such as a disk Tutte map or a rectangular Tutte map. The

Algorithm 7.5: Density-equalizing map (DEM) for simply-connected open surfaces

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

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

1 **if** *S is planar* **then**
2 Set $\vec{r}_0 = S$; ;
3 **else**
4 Compute a curvature-based flattening map $\varphi : S \to \mathbb{C}$ using Algorithm 7.2 or Algorithm 7.3. Denote $\vec{r}_0 = \varphi(S)$;
5 Define the density $\rho_0^{\mathcal{F}} = \frac{\text{Given population}}{\text{Area of the triangle}}$ on each triangle of $\vec{r}_0$;
6 Compute $\rho_0^{\mathcal{V}} = W^{\mathcal{F}\mathcal{V}}\rho_0^{\mathcal{F}}$;
7 Update $\vec{r}_0$ with the sea constructed using Algorithm 7.4;
8 Extend $\rho_0^{\mathcal{V}}$ to the whole domain by setting $\rho_0^{\mathcal{V}}$ at the sea to be the average value of $\rho_0^{\mathcal{V}}$;
9 Set $\delta t = \min\left\{\frac{\min(\rho_0^{\mathcal{V}})}{\text{mean}(\rho_0^{\mathcal{V}})}, \frac{\text{mean}(\rho_0^{\mathcal{V}})}{\max(\rho_0^{\mathcal{V}})}\right\} \times \text{Area}(S)$;
10 Set $n = 0$;
11 **repeat**
12 Update $n = n + 1$;
13 Solve $\rho_n^{\mathcal{V}} = (A_{n-1} + \delta t L_{n-1})^{-1}\left(A_{n-1}\rho_{n-1}^{\mathcal{V}}\right)$;
14 Compute the face-based gradient $(\nabla\rho)_n^{\mathcal{F}}(T) = \frac{1}{2\text{Area}(T)} N \times \left(\rho_n^{\mathcal{V}}(i)e_{jk} + \rho_n^{\mathcal{V}}(j)e_{ki} + \rho_n^{\mathcal{V}}(k)e_{ij}\right)$;
15 Compute $(\nabla\rho)_n^{\mathcal{V}} = W^{\mathcal{F}\mathcal{V}}(\nabla\rho)_n^{\mathcal{F}}$;
16 Update $\vec{r}_n = \vec{r}_{n-1} - \delta t(\nabla\rho)_n^{\mathcal{V}}/\rho_n^{\mathcal{V}}$;
17 **until** $\frac{sd(\rho_n^{\mathcal{V}})}{mean(\rho_n^{\mathcal{V}})} < \varepsilon$;
18 Obtain $f(S) = \vec{r}_n \times \frac{\text{Area}(\vec{r}_0)}{\text{Area}(\vec{r}_n)}$;

construction of the sea (line 7–8) can be skipped as we do not need to change the overall boundary shape of the initial map in this situation.

Note that the density-equalizing process is driven by the density gradient field $\nabla\rho$. To preserve the overall boundary shape of the initial map throughout the iterations, the following Neumann

boundary condition is needed:

$$(\nabla\rho) \cdot \mathbf{n} = 0. \tag{7.56}$$

In fact, the above condition is implicitly enforced in the derivation of the contangent Laplacian formulation (7.39) (see Ref. [112] for the detailed derivation). This suggest that the iterative scheme is theoretically applicable to this problem.

However, in the discrete case, the numerical error in approximating the density gradient may lead to a small nonzero normal component in $\nabla\rho$ at the boundary. To fix this issue, we can simply project all boundary vertices onto the prescribed boundary shape at the end of each iteration. thereby ensuring that all boundary vertices stay on the prescribed boundary while having the freedom to slide along it throughout the density-equalizing process. Algorithm 7.6 summarizes the procedure for computing shape-prescribed density-equalizing maps (SPDEM).

7.2.4 The choice of population and its effects

We conclude this section by listing some possible choices of the initial population and the corresponding effects on the final mapping result by our density-equalizing mapping algorithm:

- Setting a relatively high population at a certain region of the input surface will lead to an expansion of that region in the final density-equalizing mapping result.
- Similarly, setting a relatively low population at a certain region of the input surface will lead to a shrinkage of that region in the final density-equalizing mapping result.
- Setting the population to be the area of every triangle element of the input surface will lead to an area-preserving parameterization of the input surface, as we have

$$\frac{\text{Initial area}}{\text{Final area}} = \frac{\text{Given population}}{\text{Final area}} = \text{Density} = \text{Constant}. \tag{7.57}$$

Algorithm 7.6: Shape-prescribed density-equalizing map (SPDEM) for simply-connected open surfaces

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

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

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

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

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

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

Set $n = 0$;

repeat

  Update $n = n + 1$;

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

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

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

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

  Project all boundary vertices onto the prescribed boundary shape;

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

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

7.3 Experimental results

The proposed algorithms are implemented in MATLAB. The backslash operator in MATLAB is used for solving the linear systems, and the Fast InPolygon detection MEX custom MATLAB function* is used for constructing the sea. We demonstrate the effectiveness of our algorithms using various experiments. All experiments are performed on a PC with Intel i7-6700K CPU and 16 GB

*https://www.mathworks.com/matlabcentral/fileexchange/20754-fast-inpolygon-detection-mex

RAM. Some of the surface meshes in the experiments are adapted from the AIM@SHAPE Shape Repository[†]. In all experiments, the stopping parameter ε of our algorithms is set to be 10^{-3}.

7.3.1 Examples of density-equalizing maps produced by our algorithm

Figures 7.6 and 7.7 respectively show the density-equalizing maps of a square and a hexagon with a prescribed population on every triangle element. The final density $\frac{\text{Given population}}{\text{Final area}}$ highly concentrates at 1 for both examples, which indicates that our proposed DEM algorithm effectively equalizes the density. Also, the plots of $\frac{\text{sd}(\rho_n^{\mathcal{V}})}{\text{mean}(\rho_n^{\mathcal{V}})}$ versus the number of iterations show that the proposed algorithm converges rapidly.

Figure 7.8 shows the mapping result of a Gaussian shape in $\mathbb{R}^3$. The domain of the shape is $[0,1] \times [0,1]$. The population is given by $2.2 - |x| - |y|$, where (x, y) are the x- and y-coordinates of the centroid of each triangle element. Algorithm 7.2 is used for the initialization for computing the density-equalizing map. It can again be observed that the density is well equalized.

It is noteworthy that the curvature-based initial flattening map and the final density-equalizing map can be significantly different in shape (Figure 7.9). In particular, a convex initial map can become non-convex under the algorithm.

Our algorithm is capable of producing area-preserving flattening maps. Figure 7.10 shows a surface with multiple peaks in $\mathbb{R}^3$ and the mapping result. Here, we set the population as the area of each triangle element on the initial surface and use Algorithm 7.2 for the initialization. It can be observed that the flattening map effectively preserves the area ratios.

[†] http://visionair.ge.imati.cnr.it/ontologies/shapes/

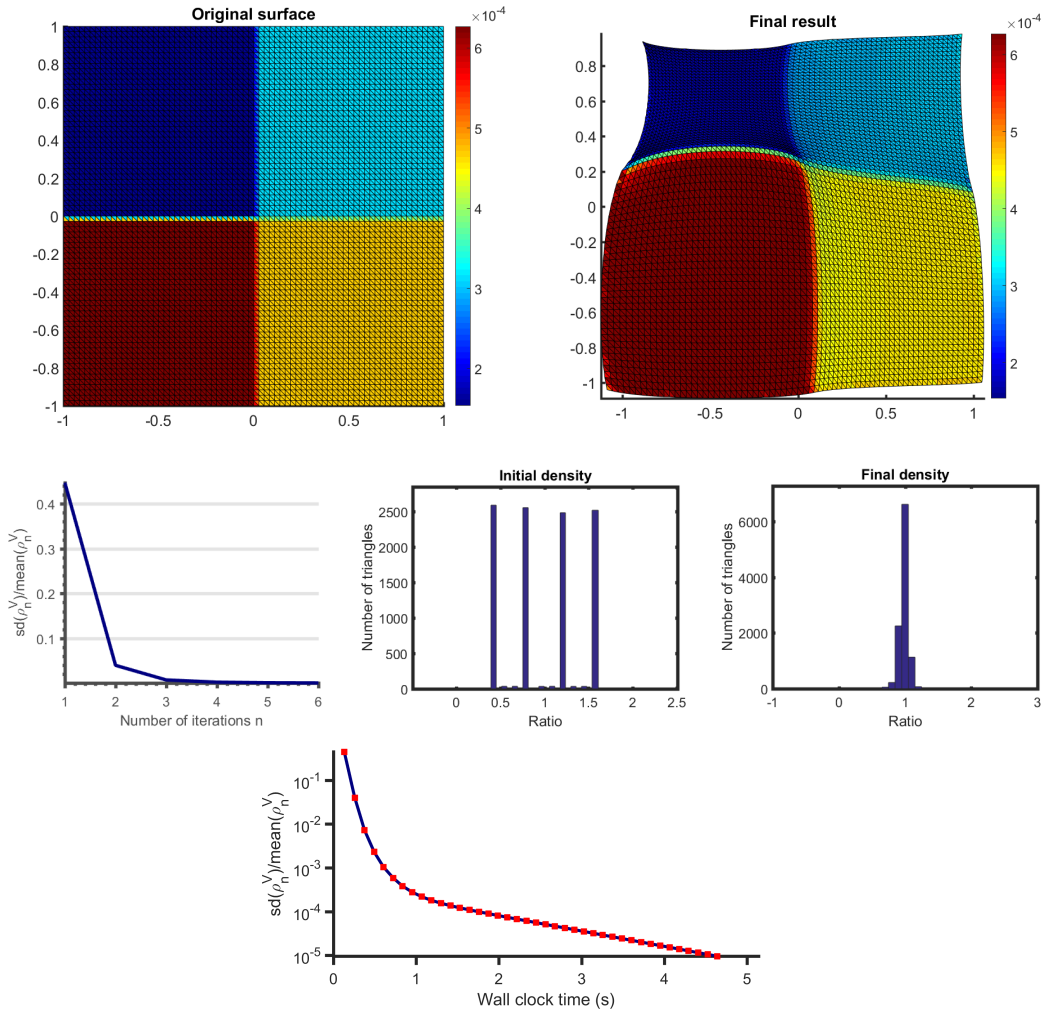

Figure 7.6: Density-equalization on a square. Top row (left to right): the initial shape colored with a given population distribution, and the resulting density-equalizing map colored with the final area of each triangle element. Middle row (left to right): the values of $\frac{\mathrm{sd}(\rho_n^{\mathcal{V}})}{\mathrm{mean}(\rho_n^{\mathcal{V}})}$, the histogram of the initial density $\frac{\text{Given population}}{\text{Initial area}}$ on each triangle element, and the histogram of the final density $\frac{\text{Given population}}{\text{Final area}}$ on each triangle element (see Table 7.1 for statistics). Bottom: A semilog plot of $\frac{\mathrm{sd}(\rho_n^{\mathcal{V}})}{\mathrm{mean}(\rho_n^{\mathcal{V}})}$ versus time with a stronger threshold of $\varepsilon = 10^{-5}$ showing rapid convergence.

Another example of area-preserving maps produced by our algorithm is shown in Figure 7.11. This time, we use Algorithm 7.3 for the initialization and then compute the density-equalizing map with the population being the area of the triangle elements. It can be observed that the locally

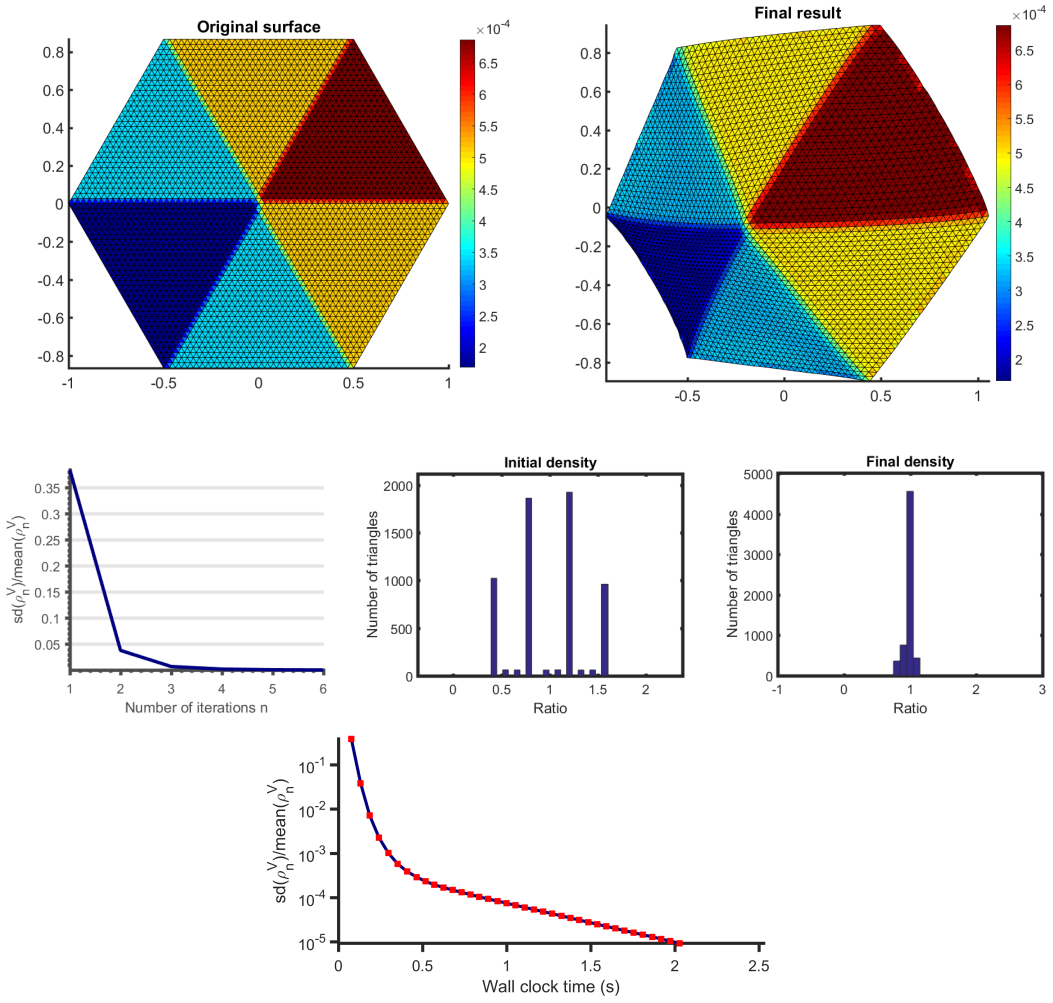

Figure 7.7: Density-equalization on a hexagon. Refer to the caption of Figure 7.6 for the description of the plots.

authalic initialization does not preserve the global area ratio, with the eyes and nose of the lion significantly shrunk. By contrast, the final density-equalizing map effectively achieves area preservation.

By changing the input population, we can use our proposed DEM algorithm for producing density-equalizing flattening maps with different effects. Two examples are provided in Figure 7.12.

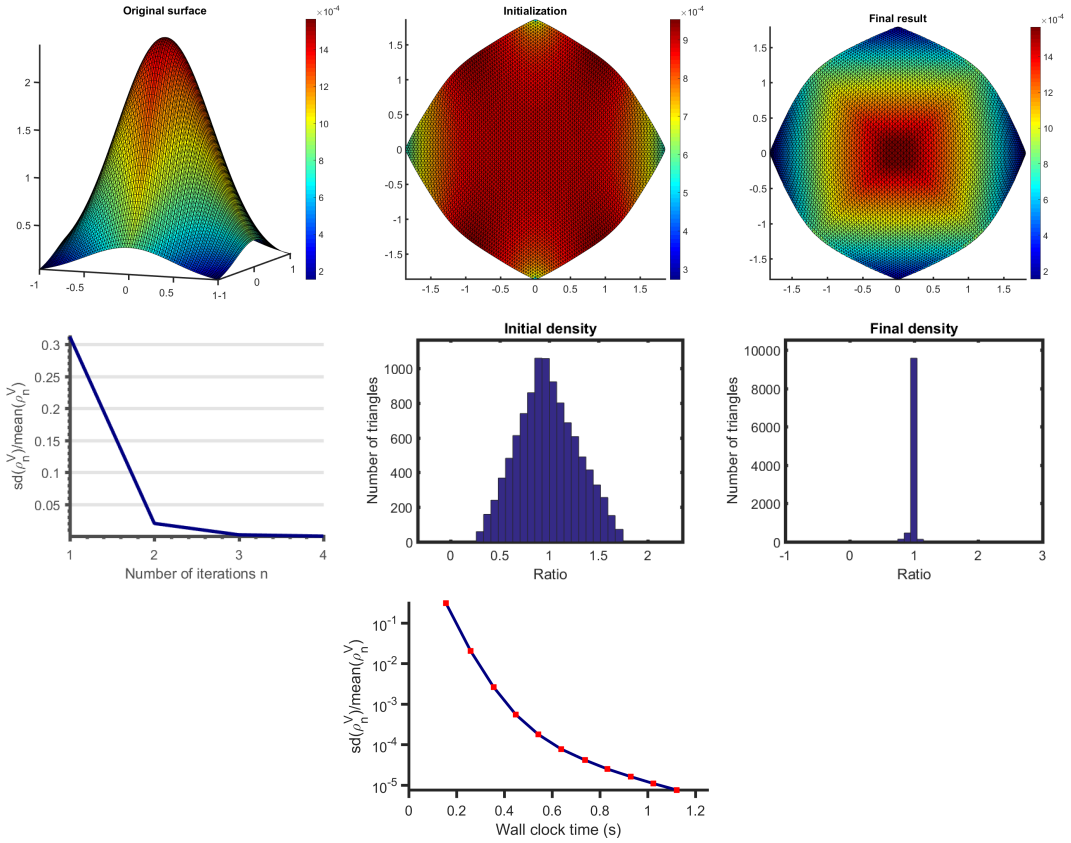

Figure 7.8: Density-equalizing map for a surface in $\mathbb{R}^3$ with Gaussian shape. Top row (left to right): the initial shape colored with a given population distribution, the curvature-based Tutte flattening initialization colored with the area of each flattened triangle element, and the final density-equalizing map colored with the final area of each triangle element. Middle row (left to right): the values of $\frac{\mathrm{sd}(\rho_n^{\mathcal{V}})}{\mathrm{mean}(\rho_n^{\mathcal{V}})}$, the histogram of the density $\frac{\text{Given population}}{\text{Initial flattened area}}$ on each flattened triangle element after the Tutte flattening initialization, and the histogram of the density $\frac{\text{Given population}}{\text{Final area}}$ on each triangle element of the final result. See Table 7.1 for statistics. Bottom: A semilog plot of $\frac{\mathrm{sd}(\rho_n^{\mathcal{V}})}{\mathrm{mean}(\rho_n^{\mathcal{V}})}$ versus time with a stronger threshold of $\varepsilon = 10^{-5}$ showing rapid convergence.

For the Niccolò da Uzzano model, by setting the input population at the eyes to be twice the area of the triangles there and that at the remaining parts to be the area of each triangle element, we can achieve a flattening map with the eyes magnified. Similarly, for the Max Planck model, by setting the input population at the nose to be 1.5 times the area of the triangles there and that at the remaining

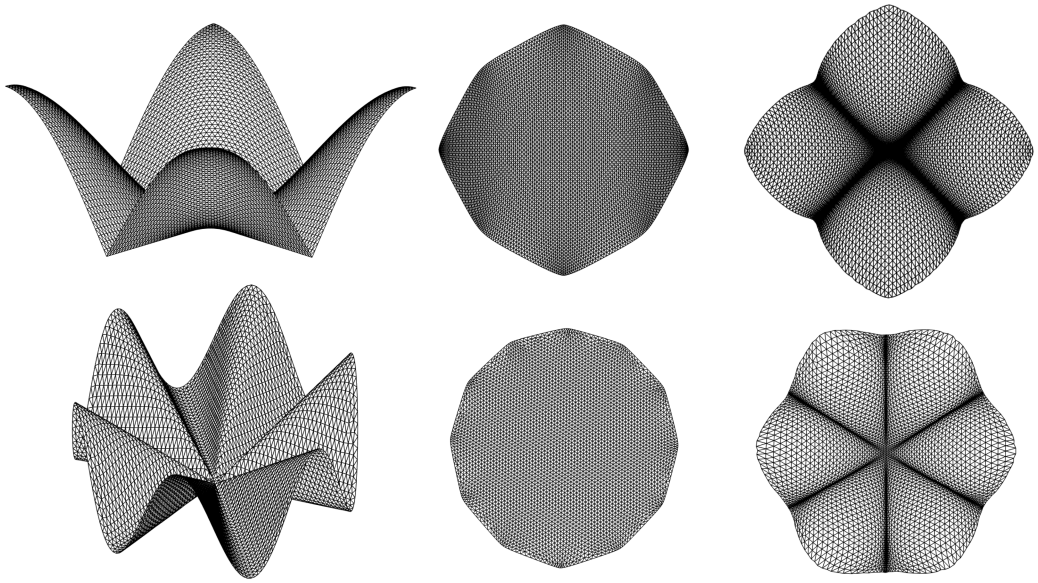

Figure 7.9: Two more examples of density-equalizing maps. Here we set the population according to the height of different parts of the surfaces, aiming to achieve an expansion at the peaks. Left: The original surfaces. Middle: The curvature-based initial flattening maps. Right: The density-equalizing mapping results. Note that the convex boundaries in the initialization can become non-convex under DEM.

parts to be the area of each triangle element, we can effectively magnify the nose in the flattening map.

As discussed in Section 7.2.3, the proposed SPDEM algorithm is capable of producing density-equalizing maps with a prescribed target shape. Figure 7.13 shows four examples, including two examples of mapping the square with the prescribed population in Figure 7.6 to a square and a rectangle with aspect ratio $2:3$ respectively, and two examples of mapping the hexagon with prescribed population in Figure 7.7 to a circle and an ellipse with aspect ratio $2:1$ respectively. It can be observed that the target shapes are effectively achieved, with the boundary vertices optimally moved along the boundary to achieve density-equalization. Besides, similar to the proposed DEM

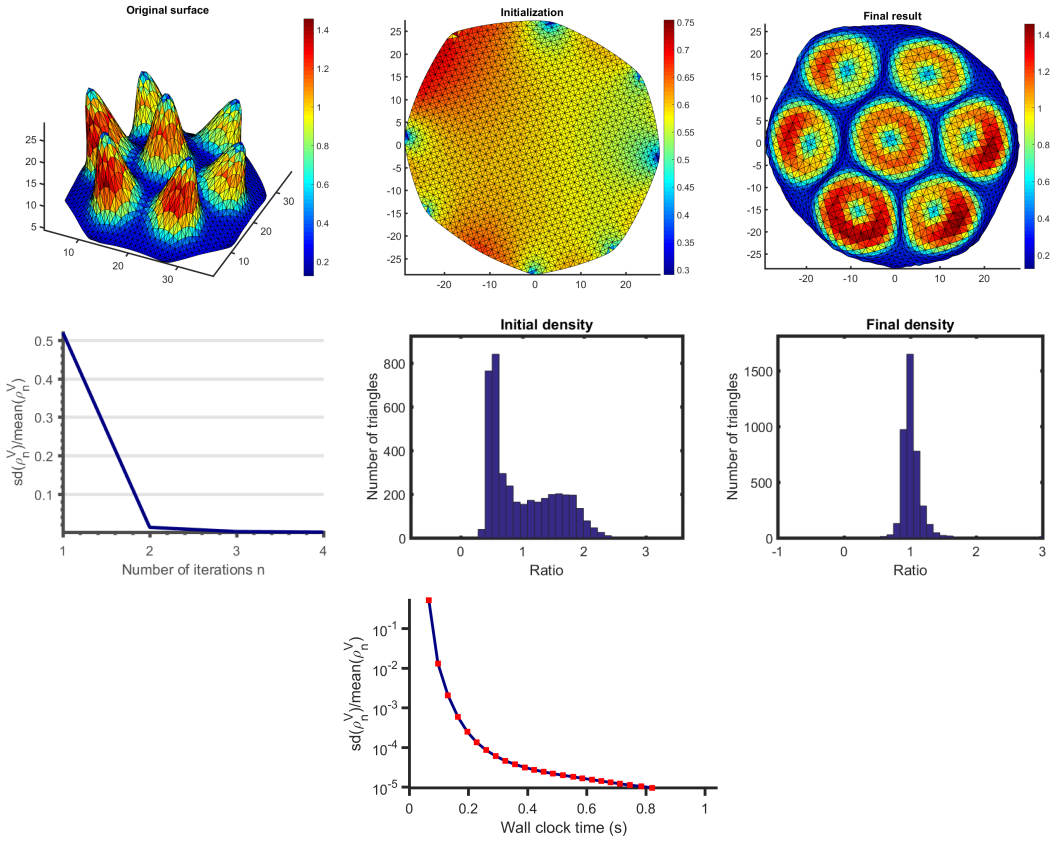

Figure 7.10: Area-preserving parameterization of a surface with multiple peaks in $\mathbb{R}^3$. Top row (left to right): the initial surface colored with the area of each triangle element, the curvature-based Tutte flattening map colored with the area of each flattened triangle element, and the final density-equalizing map colored with the final area of each triangle element. Middle row (left to right): the values of $\frac{\text{sd}(\rho_n^{\mathcal{V}})}{\text{mean}(\rho_n^{\mathcal{V}})}$, the histogram of the density $\frac{\text{Initial area}}{\text{Initial flattened area}}$ on each flattened triangle element after the Tutte initialization, and the histogram of the density $\frac{\text{Initial area}}{\text{Final area}}$ on each triangle element of the final result (see Table 7.1 for statistics). Bottom: A semilog plot of $\frac{\text{sd}(\rho_n^{\mathcal{V}})}{\text{mean}(\rho_n^{\mathcal{V}})}$ versus time with a stronger threshold of $\varepsilon = 10^{-5}$ showing rapid convergence.

algorithm, the SPDEM algorithm can be used for computing area-preserving parameterizations onto a prescribed domain such as a disk or a rectangle (Figure 7.14). We can further achieve the effects shown in Figure 7.12 with the boundary shape prescribed (Figure 7.15).

For surfaces with a highly convoluted boundary, it may be difficult to directly compute the

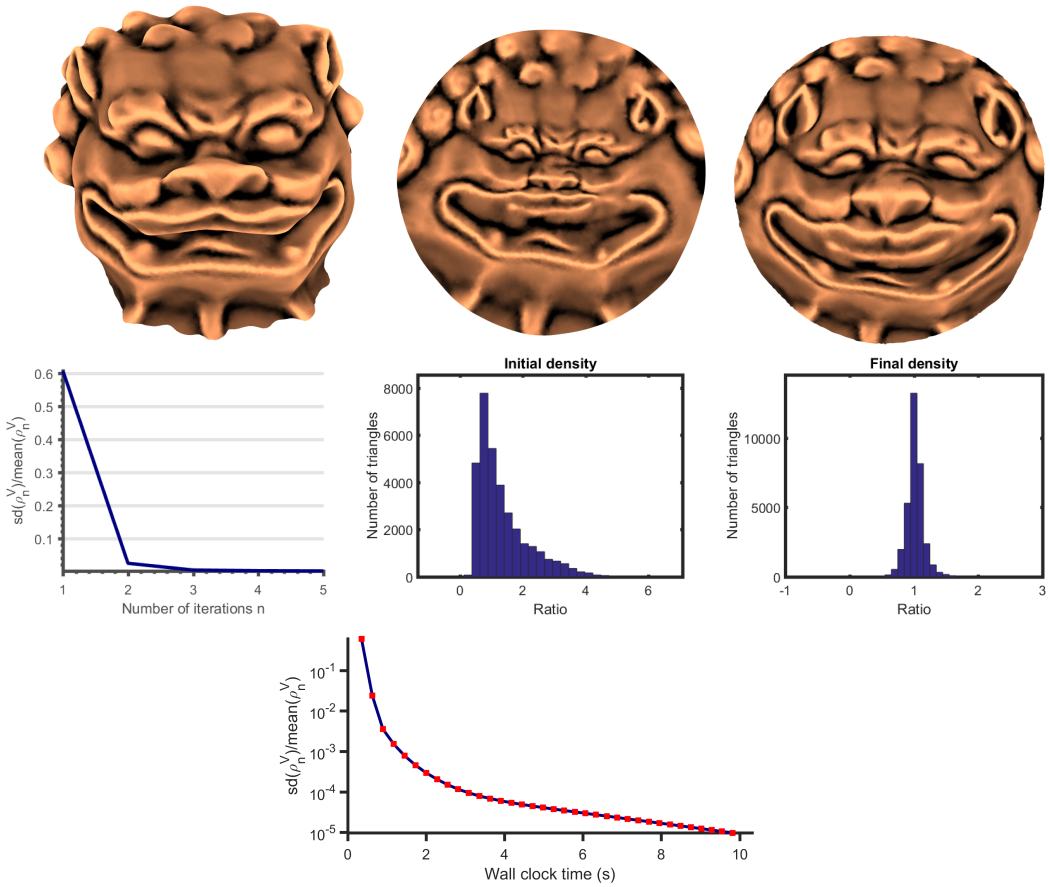

Figure 7.11: Area-preserving parameterization of a lion face in $\mathbb{R}^3$. Top row (left to right): the initial surface, the curvature-based locally authalic flattening map, and the final density-equalizing map. All meshes are colored with the mean curvature of the input lion face. Refer to the caption of Figure 7.10 for the description of the four remaining plots.

curvature-based flattening map. To overcome this problem, a simply remedy is to extend our sea approach and prescribe a sea with a simpler shape around the given surface. With the simpler overall shape, the new surface can be easily flattened, and the density-equalizing map can then be computed for achieving different desired effects. Figure 7.16 shows a simply-connected open torus surface with a space-filling curve pattern. Here, we define the population on the mesh with $\frac{\max(\text{population})}{\min(\text{population})} \approx 25$ to produce a large deformation under density-equalization. By prescribing a sea to fill up the gaps

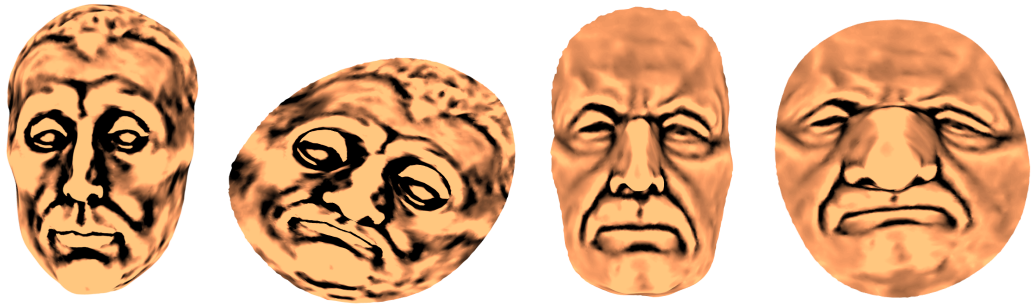

Figure 7.12: Density-equalizing flattening maps with different effects obtained by our density-equalizing map (DEM) algorithm. Left: The Niccolò da Uzzano model and the mapping result with the eyes magnified. Right: The Max Planck model and the mapping result with the nose magnified (see Table 7.1 for statistics).

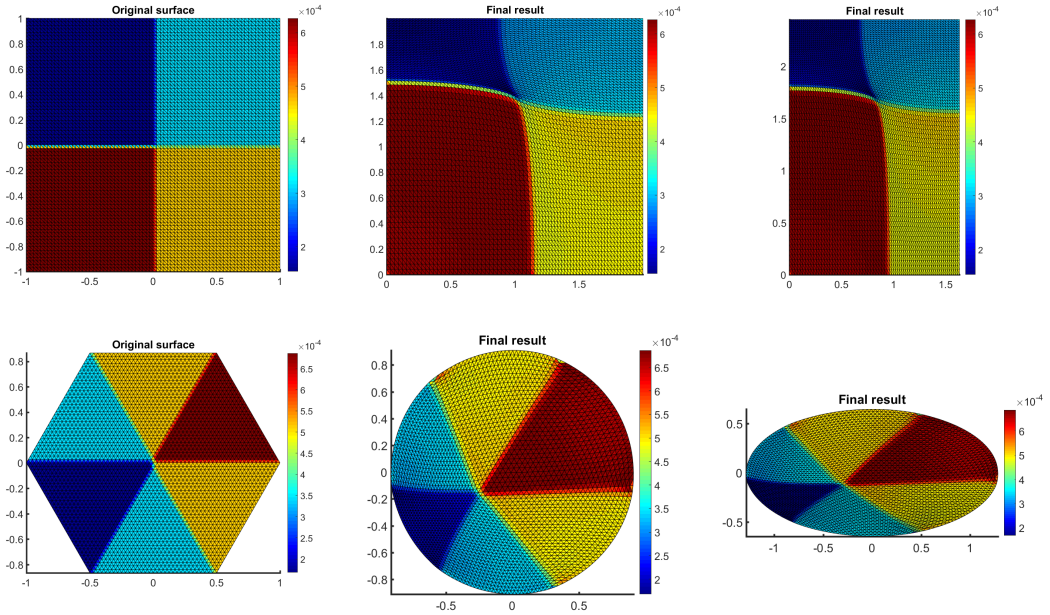

Figure 7.13: Density-equalizing maps produced by our SPDEM algorithm. Top row: We compute two density-equalizing maps for the example in Figure 7.6 with the target shapes being a square and a rectangle with aspect ratio 2:3 respectively. Bottom row: We compute two density-equalizing maps for the example in Figure 7.7 with the target shapes being a disk and an ellipse with aspect ratio 2:1 respectively (see Table 7.2 for statistics).

on the torus, we can flatten the entire shape onto a rectangle and compute the density-equalizing map, thereby achieving the desired mapping effect of the original surface. It is also noteworthy that

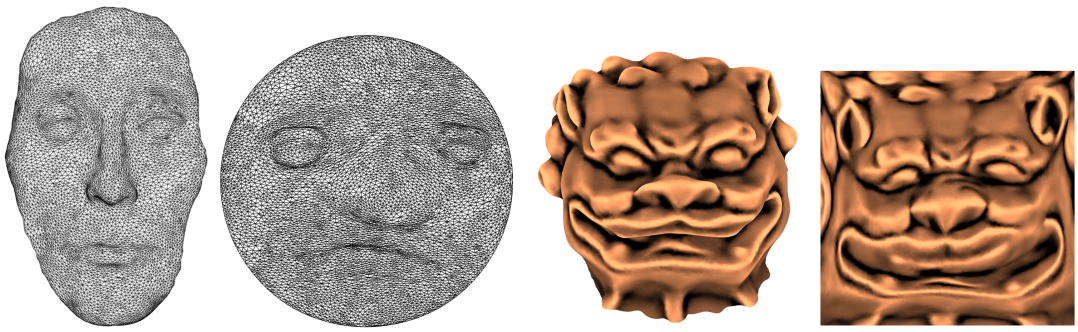

Figure 7.14: Area-preserving parameterizations produced by our SPDEM algorithm. Left: The human face model and the area-preserving parameterization of it onto a disk. Right: The lion model and the area-preserving parameterization onto a square (see Table 7.2 for statistics).

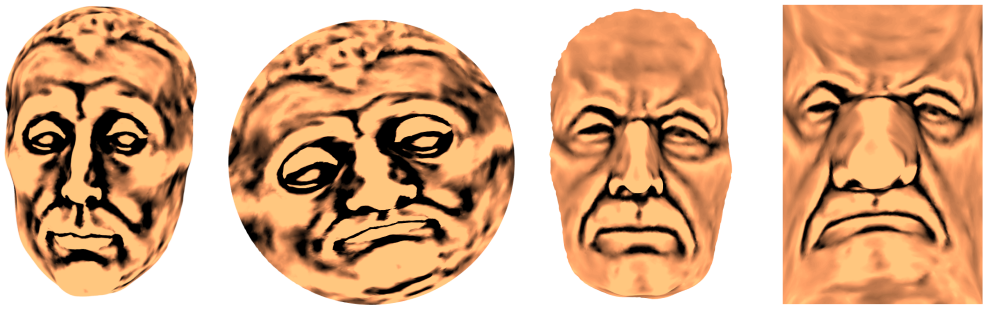

Figure 7.15: shape-prescribed density-equalizing flattening maps with the effects in Figure 7.12. Left: The Niccolò da Uzzano model and the disk mapping result with the eyes magnified. Right: The Max Planck model and the rectangular mapping result with the nose magnified (see Table 7.2 for statistics).

the sea helps regularize the deformation and avoid mesh overlaps. As the density information at two geometrically close but topologically distant regions on a convoluted surface can be transmitted between each other via the sea directly without going through the complicated domain, the two regions can coordinate with each other and find a non-overlapping direction for the expansion.

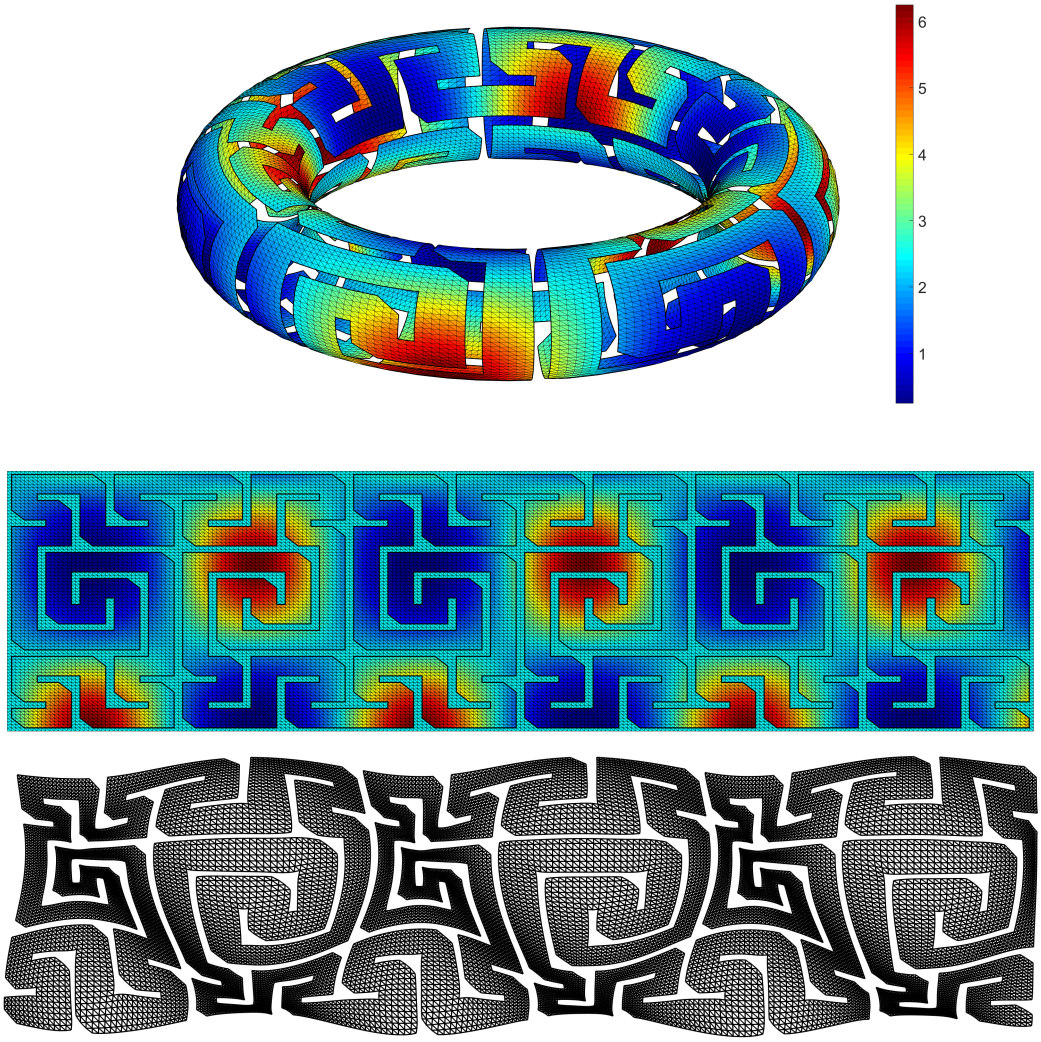

Figure 7.16: Density-equalizing map for a simply-connected open torus surface with a space-filling curve pattern. Top: The torus surface color-coded with the input population. Middle: We prescribe a sea with a simpler shape around the surface and flatten the new surface for computing the density-equalizing map. Bottom: The resulting density-equalizing map.

7.3.2 Quantitative results of our algorithm

We then analyze the performance of our DEM algorithm more quantitatively (Table 7.1). It can be observed that the convergence of DEM is fast. Also, from the median and the inter-quartile range of

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

Table 7.1: The performance of our density-equalizing map (DEM) algorithm. The number of triangle elements, the time taken (in seconds) for the entire algorithm (including the computation of the initial map and the construction of sea), the number of iterations taken, and the median and interquartile range of the density defined on each triangle element by $\frac{\text{Given population}}{\text{Final area}}$ are recorded.

the density, it can be observed that the density is well-equalized. The experiments show that DEM is efficient and accurate.

Table 7.2 shows the performance of the proposed SPDEM algorithm. Since SPDEM does not involve the sea, it is faster than DEM. However, because of the extra shape constraints, SPDEM is not as accurate as DEM.

We now compare the performance of our proposed algorithm and GN (with the implementation available online[‡]). As GN only works on finite difference grids, we deploy the two methods on a 100×100 square grid $\{(x, y) \in \mathbb{Z}^2 : 0 \leq x, y \leq 99\}$ for a fair comparison. For our triangle-based

[‡] http://www-personal.umich.edu/~mejn/cart/

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

Table 7.2: The performance of our shape-prescribed density-equalizing map (SPDEM) algorithm.

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

Table 7.3: The performance of our density-equalizing map (DEM) algorithm and GN deployed on a 100×100 square mesh with different input population functions. Here, we evaluate the map difference by mean$\left(\frac{|z_{\text{prev}}-z_{\text{ours}}|}{\text{side length of square}}\right)$, where z_{prev} and z_{ours} are the complex coordinates of the density-equalizing mapping results by GN and our method respectively. $\bar{x}$ and $\bar{y}$ denote the mean of the x- and y-coordinates of the square.

DEM algorithm, we divide each square into two right-angled triangles. For GN, the dimension of the sea is set to be two times the linear extent of the square grid. We test the two methods with various population functions (Table 7.3). It can be observed that our DEM algorithm is faster than GN by over 80%, while the mapping results produced by the two methods are nearly identical (see also Figure 7.17). The experiments suggest that our proposed method is advantageous over GN even for computing density-equalizing maps on 2D grids.

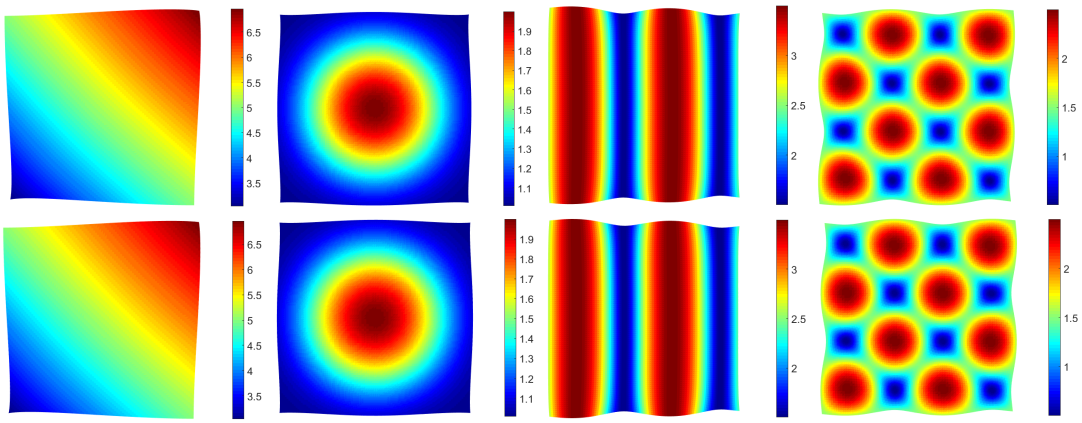

Figure 7.17: The density-equalizing maps produced by our DEM algorithm and GN with various input population functions. Each column shows a set of results color-coded by an input population function in Table 7.3. Top row: The mapping results by GN. Bottom row: The mapping results by our DEM algorithm.

7.3.3 Comparison with other parameterization methods

We first qualitatively compare our DEM algorithm with the various prior parameterization algorithms [35,151,107,26] for parameterizing a surface with multiple peaks (Figure 7.18). It can be observed that the peaks are substantially shrunk under the free-boundary conformal parameterization [35] and the disk conformal parameterization [26]. Also, the boundary of the free-boundary conformal parameterization is significantly distorted. By contrast, the peaks are flattened without being shrunk under our DEM algorithm, as well as the optimal mass transport (OMT) map [151] and the scalable locally injective map (SLIM) [107]. More mapping results produced by our DEM algorithm, OMT and SLIM are shown in Figure 7.19, from which we observe that DEM and OMT are more capable than SLIM in avoiding squeezed regions in the mapping results (such as the peak of the Gaussian surface and the nose of the human face).

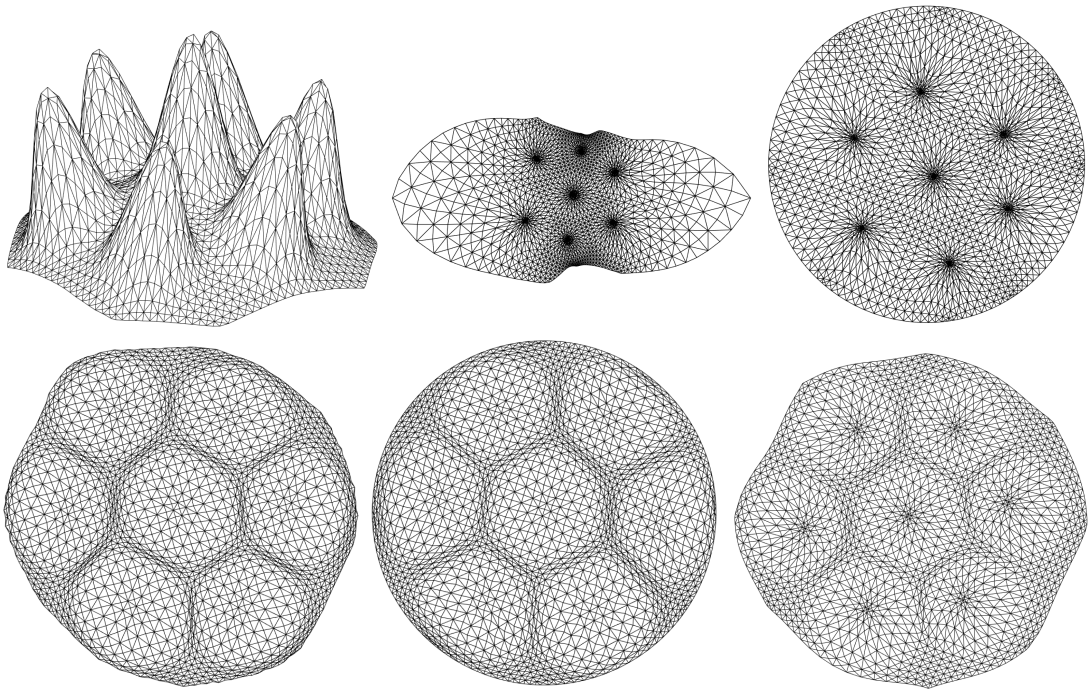

Figure 7.18: Comparison of different parameterization methods for a surface with multiple peaks in $\mathbb{R}^3$. Top left: The input surface. Top middle: The free-boundary conformal parameterization [35]. Top right: The disk conformal parameterization [26]. Bottom left: The area-preserving parameterization by our DEM algorithm. Bottom middle: The optimal mass transport (OMT) map [151]. Bottom right: The scalable locally injective map (SLIM) [107].

We then quantitatively compare our DEM algorithm with OMT[§] and SLIM[¶], in terms of the efficiency and accuracy, for computing area-preserving parameterizations. The parameterization results computed by the three algorithms are rescaled so that the total area of each map is the same as that of the input surface. We then evaluate the area-preserving property of the three algorithms by computing the absolute relative error in triangle area for every triangle T:

$$E_A(T) = \left| \frac{\text{Area of } T \text{ on the parameterization}}{\text{Area of } T \text{ on the original surface}} - 1 \right|. \tag{7.58}$$

[§] http://www3.cs.stonybrook.edu/~gu/software/omt/index.html
[¶] http://github.com/MichaelRabinovich/Scalable-Locally-Injective-Mappings

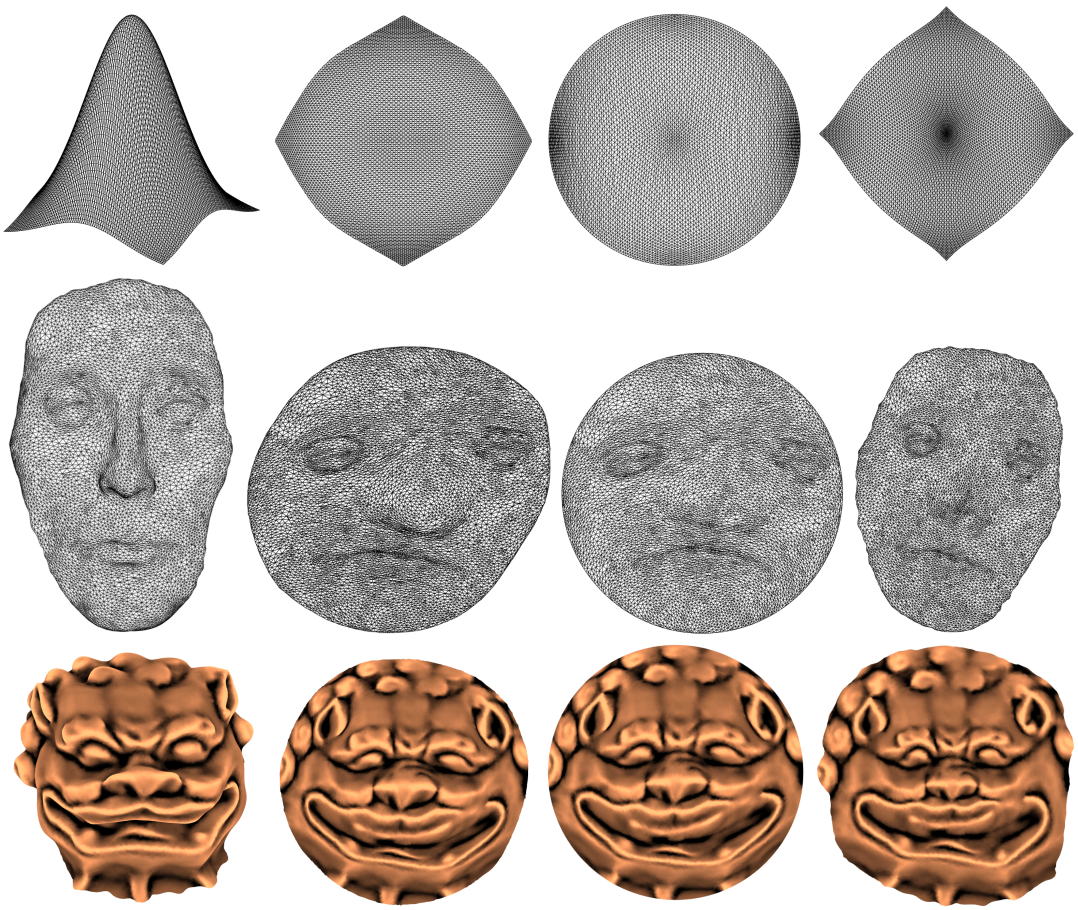

Figure 7.19: More comparisons between our density-equalizing map (DEM) algorithm with other surface parameterization methods. Left to right: The input surface, our DEM algorithm, the optimal mass transport (OMT) map [151], and the scalable locally injective map (SLIM) [107].

Here, a smaller E_A indicates that the parameterization is more area-preserving. From Table 7.4, it can be observed that our DEM algorithm is more efficient and accurate for computing area-preserving parameterizations when compared to OMT and SLIM.

Surface	No. of triangles	Measure	OMT [151]	SLIM [107]	DEM
Gaussian	10368	Time (s)	0.956	3.413	0.333
		# iterations	22	20	3
		mean(E_A)	0.1014	0.2678	0.0164
		std(E_A)	0.0791	0.2041	0.0233
		median(E_A)	0.1065	0.2104	0.0101
		IQR(E_A)	0.1157	0.2839	0.0154
Peaks	4108	Time (s)	0.466	0.998	0.149
		# iterations	24	20	4
		mean(E_A)	0.1108	0.3214	0.0928
		std(E_A)	0.1015	0.1791	0.1027
		median(E_A)	0.0863	0.3107	0.0649
		IQR(E_A)	0.1106	0.1892	0.0934
Lion	33369	Time (s)	3.278	14.791	1.444
		# iterations	22	20	5
		mean(E_A)	0.1285	0.1857	0.0938
		std(E_A)	0.1062	0.1298	0.0981
		median(E_A)	0.1050	0.1612	0.0640
		IQR(E_A)	0.1299	0.1934	0.0982
Niccolò da Uzzano	25900	Time (s)	2.469	9.211	2.020
		# iterations	22	20	8
		mean(E_A)	0.1282	0.1400	0.0737
		std(E_A)	0.1101	0.0970	0.1461
		median(E_A)	0.1037	0.1266	0.0369
		IQR(E_A)	0.1293	0.1250	0.0589
Max Planck	26452	Time (s)	3.035	9.762	2.021
		# iterations	26	20	9
		mean(E_A)	0.1223	0.1075	0.0754
		std(E_A)	0.1015	0.1078	0.1839
		median(E_A)	0.0991	0.0786	0.0333
		IQR(E_A)	0.1243	0.1174	0.0540
Human face	33369	Time (s)	1.700	3.844	0.793
		# iterations	30	20	6
		mean(E_A)	0.1511	0.0830	0.0612
		std(E_A)	0.1386	0.0824	0.1438
		median(E_A)	0.1174	0.0613	0.0291
		IQR(E_A)	0.1507	0.0830	0.0481

Table 7.4: The performance of our DEM algorithm compared with the state-of-the-art nonlinear parameterization algorithms for computing area-preserving parameterizations. For the optimal mass transport (OMT) map [151] and the scalable locally injective map (SLIM) [107], the default parameter settings in their implementation are used: The error threshold for OMT is set to be 0.0001, and the number of iterations for SLIM is set to be 20.

7.3.4 On the use and construction of the sea

Note that most prior parameterization methods do not involve the construction of a sea surrounding the given surface. In our proposed DEM algorithm, the sea is useful not only for aiding the density-equalization process but also for analyzing the physical phenomenon of density propagation around the region of interest.

Here, we consider the deformation of the sea from a physical point of view. Let r be the distance of a tracer particle at the sea from the origin before the deformation, and $\Delta r = r_{\text{final}} - r$ be the change in the distance of it under the density-equalization process. We analyze the density propagation under the proposed DEM algorithm by tracking Δr for all vertices at the sea. From the log–log plots Δr against r outside the unit circle shown in Figure 7.20, it can be observed that $\Delta r \propto r^{-2}$ at the outer part of the sea. In other words, the effect of density diffusion on the displacement of particles at the sea decays quadratically. From an algorithmic point of view, this observation suggests that the construction of a coarser sea at the outermost part by reflection does not affect the accuracy of the density-equalizing map.

We now compare our reflection-based construction of an adaptive sea and other standard construction approaches. Here, we replace our adaptive sea with a sea consisting of uniformly spaced nodes with various choices of spacing, and analyze the performance of the density-equalization algorithm.

More specifically, we consider three choices of node spacing at the sea. From Table 7.5, it can be observed that the mapping results with our adaptive sea is as accurate as those with a uniform dense

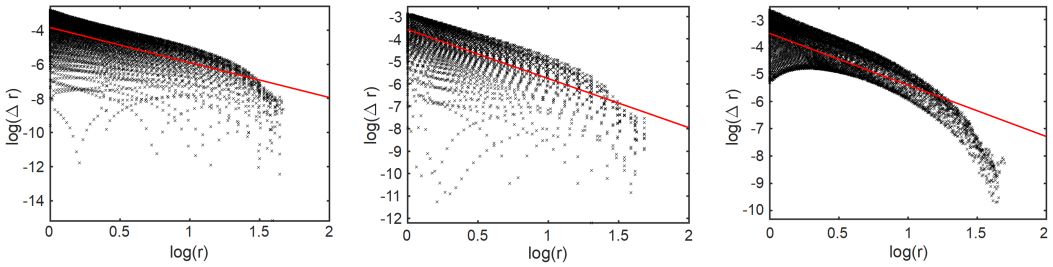

Figure 7.20: The log–log plot of the displacement of tracer particles at the sea under our density-equalizing map (DEM) algorithm. r is the distance of a point at the sea from the origin, and Δr is the change in the distance under the density-equalizing map. Each cross represents a node at the sea, and the red line is the least-squares line. Here we only consider the region outside the unit circle and hence the x-axis starts from 0. Left: The square example. Middle: The hexagon example. Right: The human face example.

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

Table 7.5: The performance of the density-equalizing maps with our adaptive sea and a uniformly sea. The point spacing at the coarse, moderate, and dense level are respectively $5a$, $3a$ and a, where a is the average spacing of the vertices of the initial flattening maps. For a fair comparison, the overall size of the uniform sea is set to be 5, which is consistent with our choice of $\eta = 5$ in our reflection-based construction. d_{bdy} is defined by $\frac{\text{Given population}}{\text{Final area}}$ at the boundary elements of the surfaces under the density-equalizing maps.

sea consisting of over 10 times of nodes, and the computational time is over 90% shorter. While the computation with a coarse sea is faster than that with our adaptive sea, the accuracy of the mapping

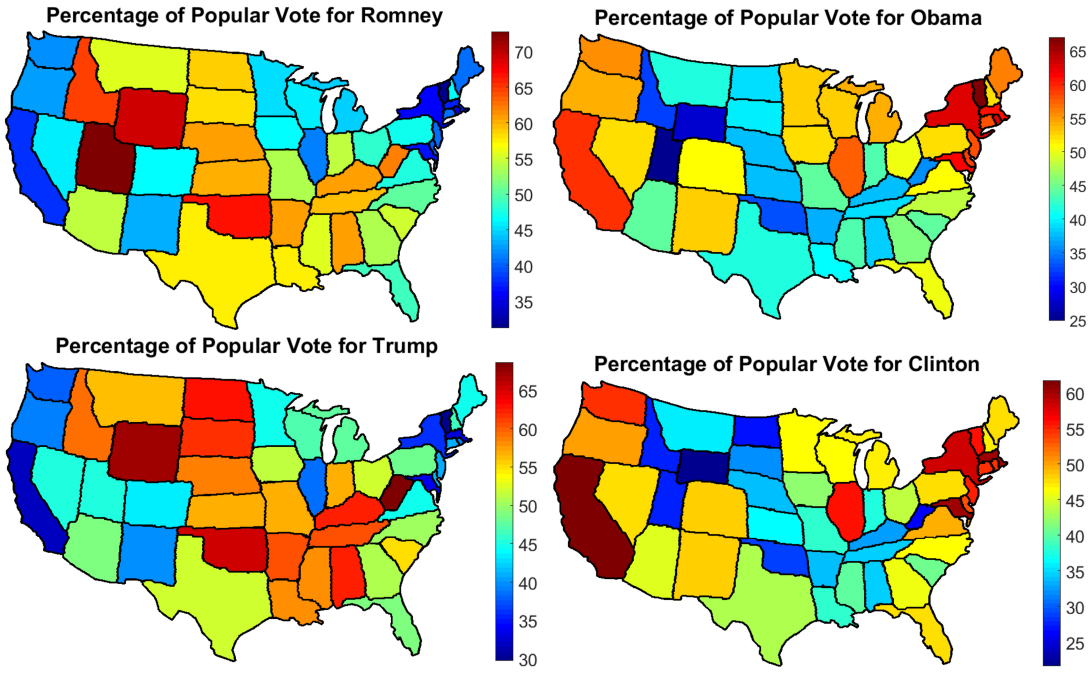

Figure 7.21: Percentage of popular vote in each state visualized on density-equalizing US maps (only including the contiguous 48 states) in the 2012 and 2016 US presidential elections. For enhancing the visual quality, the triangulations are not shown.

results is much lower. The experiments suggest that our reflection-based sea construction is capable of accelerating the computation without sacrificing the accuracy.

7.4 Applications

Below, we discuss two applications of our proposed density-equalizing mapping algorithm.

7.4.1 Data visualization

Similar to GN, our proposed algorithm can be utilized for data visualization. Here we consider visualizing the percentage of popular vote for the Republican party and the Democratic party in the

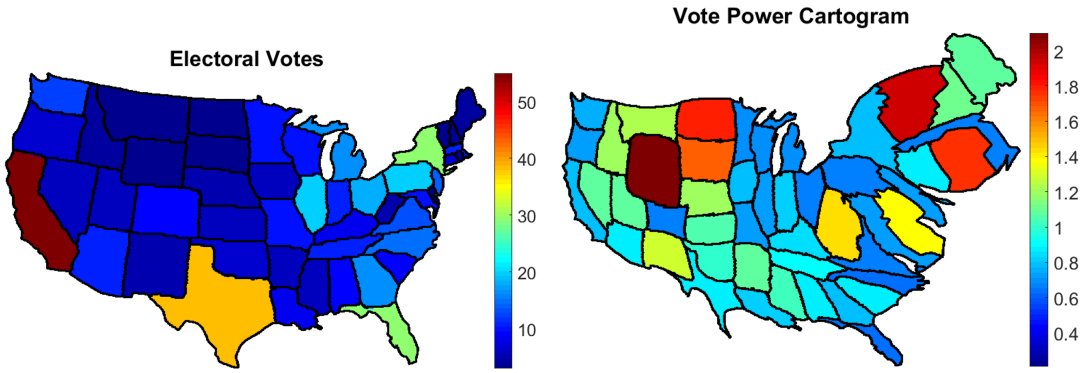

Figure 7.22: Visualizing the vote power of different states (only including the contiguous 48 states). Left: The normal US map color coded by the number of electoral votes. Right: The area cartogram generated by DEM that accurately reflects the vote power of all states.

contiguous 48 states in the 2012 and 2016 US presidential elections. The population on each state on a triangulated US map is set to be the percentage of popular vote obtained by the two parties. From the density-equalizing mapping results shown in Figure 7.21, it can be observed that the east coast and west coast are significantly shrunk for the Republican party. This reflects the relatively low percentage of popular vote obtained at those regions. By contrast, the corresponding regions are significantly enlarged in the mapping results for the Democratic party, which reflects the relative high percentage of popular vote there. Comparing the 2012 and the 2016 mapping results, we observe that the area of California becomes more extreme in the 2016 maps when compared to those in the 2012 maps. More specifically, it has further shrunk on the Trump map while further expanded on the Clinton map. The area of West Virginia has reduced in the map for Clinton when compared to that for Obama, while it has increased in the map for Trump when compared to that for Romney.

One can also use our DEM algorithm to analyze the electoral college system for the US

presidential elections. Noticing that the number of electoral votes is different at different states, we evaluate the vote power of each popular vote at each state by

$$\text{Vote power} = \frac{\frac{\text{Number of electoral votes}}{\text{Number of popular votes}}}{\text{mean}\left(\frac{\text{Number of electoral votes}}{\text{Number of popular votes}}\right)}. \tag{7.59}$$

To visualize the vote power using our DEM algorithm, we note that this quantity is not related to the area of each state. Therefore, we remove the effect of the original area of each state by running our DEM algorithm with the input population being $\frac{\text{Vote power}}{\text{Area of the state}}$ on the contiguous US map. The resulting area cartogram then solely reflects the vote power (Figure 7.22). When compared to the ordinary contiguous US map, the area cartogram produced by DEM provides a more intuitive view of the vote power. For instance, it can be observed that Wyoming is much bigger than Texas in the area cartogram as the vote power there is significantly larger. The two examples show that our DEM algorithm is useful for data visualization.

7.4.2 Adaptive surface remeshing

In our DEM algorithm, the input population affects the size of different regions in the resulting density-equalizing maps, with regions corresponding to a higher population magnified and those corresponding to a lower population shrunk. Using this property, we can utilize our DEM algorithm for adaptive surface remeshing.

More specifically, let S be a simply-connected open surface to be remeshed. We first compute the density-equalizing map $f : S \to \mathbb{C}$ based on a given population. Then, we consider a set of

Figure 7.23: Remeshing a human face. Top left: The input human face. Top middle: The remeshing result via our DEM algorithm. Top right: The remeshing result via the free-boundary conformal parameterization [35]. Bottom left: The density-equalizing map by our DEM algorithm. Bottom right: The free-boundary conformal parameterization [35].

uniformly distributed points $\mathcal{P}$ on the mapping result and triangulate the points. Denote the triangulation by $\mathcal{T}$. Finally, using the inverse mapping f^{-1}, we interpolate $\mathcal{P}$ onto S and the resulting mesh $(f^{-1}(\mathcal{P}), \mathcal{T})$ is a remeshed version of S.

Note that we can increase the level of detail at a region of S by setting a larger population there. As the region is enlarged in the density-equalizing mapping result and $\mathcal{P}$ is uniformly distributed, more points will lie on that part. Therefore, more points will be mapped onto that particular region of S under the inverse mapping.

Figure 7.23 shows the results of remeshing a triangulated human face via our DEM algorithm and the free-boundary conformal parameterization [35]. For our algorithm, we set the population to be the triangle area of the original mesh. It can be observed that the eyes and the nose of the human face are enlarged in our density-equalizing mapping result, while those features are shrunk in the conformal parameterization. Therefore, in the remeshing results via conformal parameterization, the representation of the nose is poor. By contrast, the remeshing result via our DEM algorithm consists of points which are more evenly distributed on the surface. This demonstrates the strength of our algorithm in surface remeshing.

7.5 Discussion

In this work, we have proposed two density-equalizing mapping algorithms (DEM and SPDEM) for simply-connected open surfaces, with numerical experiments showing the effectiveness of them for different applications.

As the discretization accuracy of the cotangent Laplacian is affected by the mesh regularity, a possible method for further improving the accuracy of the density-equalizing maps is to introduce an extra step of recomputing a Delaunay triangulation at every iteration in Algorithm 7.5. Since

the density values are vertex-based, changing the triangulation will not lead to any ambiguity in the computation. Also, as the deformation is based on $\mathbf{r}_n = \mathbf{r}_{n-1} - \delta t(\nabla\rho)_n^{\mathcal{V}}/\rho_n^{\mathcal{V}}$, one may prevent the occurrence of fold-overs by changing the timestep δt adaptively throughout the iterations.

While the proposed algorithms focus on simply-connected open surfaces in $\mathbb{R}^3$, they can be naturally extended for mapping general surfaces. For instance, one can compute density-equalizing maps of multiply-connected surfaces by filling up the holes and treating them as the sea. The entire surface then will become simply-connected and hence we can apply the proposed algorithms. Similarly, we can handle multiple disconnected surfaces by connecting them using a sea.

Where there is matter, there is geometry.

Johannes Kepler

8

Area-preserving map of 3D carotid models

Atherosclerosis is a focal disease with plaques occurring at bends and bifurcations (BFs) of the carotid artery predominantly, causing ischemic stroke[44]. It is therefore important to monitor the local changes in the vessel-wall-plus-plaque thickness (VWT) of carotid artery, which is defined as the pointwise distance between the

lumen-intima boundary (LIB) and the media-adventitia boundary (MAB)[22], for aiding the development of sensitive biomarkers that can identify high-risk patients with rapid plaque progression in a shorter time frame. Instead of visualizing the VWT-Change distribution for an individual patient on a three-dimensional (3D) surface, it is more preferable to have a flattened representation of the carotid artery surface so that clinicians can examine the distribution systematically without having to rotate and interact with the 3D surface[12]. Also, since the geometry of the carotid artery surfaces is highly subject-specific, having a standardized two-dimensional (2D) template facilitates quantitative local comparisons of the VWT-Change distributions among subjects.

Chiu *et al.*[23] developed the arc-length scaling (ALS) mapping method for flattening the 3D carotid surfaces onto a standardized 2D non-convex L-shaped domain (Figure 8.1), with the external carotid artery (ECA) (i.e. the left branch above the bifurcation point) excluded as plaques at ECA are not directly related to stroke. The ALS method has been applied for the development of sensitive biomarkers in clinical trials evaluating the effect of atorvastatin[23,20] and B Vitamins[19]. However, the ALS method does not minimize any local geometric distortion by the flattening procedure. It is well-known that surface flattening always introduces distortions in either angle or area (or both) unless the Gaussian curvature of the surface is zero[48].

Conformal (angle-preserving) flattening methods have been proposed for tubular surfaces[152,3,62,152], with the local geometry of the surfaces taken into account. However, these methods produce 2D maps with shapes depending on the geometry of the input surfaces, making the flattened domain subject-specific. Antiga and Steinman[3] developed a method for producing a

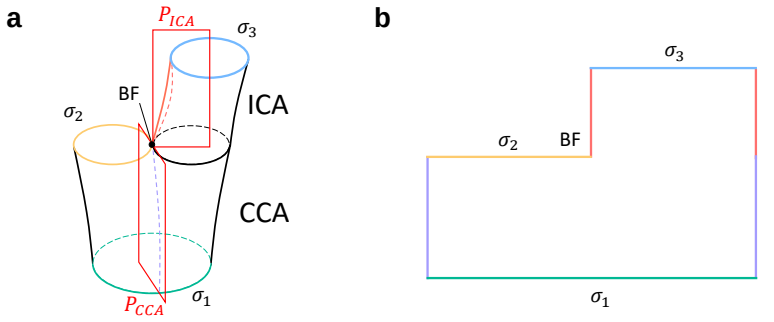

Figure 8.1: An illustration of the 2D arc-length scaling (ALS) map[23]. **a**, A carotid surface is first cut by two planes, denoted by P_{ICA} and P_{CCA}. **b**, The surface is then unfolded to a 2D L-shaped non-convex domain. The arc-length of transverse contours segmented from 2D transverse images is rescaled such that all vertices on the carotid surface correspond to uniformly spaced grid points on the 2D domain. The bifurcation (BF) is mapped to the non-convex corner of the domain, and the carotid boundaries σ_1, σ_2, σ_3 are mapped to the horizontal boundaries.

standardized 2D map by decomposing the common, internal and external carotid arteries (CCA, ICA and ECA) into three topological cylinders, with each of them subsequently flattened onto the 2D plane by solving a Dirichlet problem[2]. However, the flattened maps for the CCA, ICA and ECA are generated independently and displayed in three discontinuous sections, which hampers clinical interpretation of the VWT-Change map at the bifurcation and the carotid bulb.

As areal and volumetric measurements of plaques are related to the stroke risk of patients[126,145], area-preserving flattening methods are more preferable. However, while some area-preserving carotid flattening methods have been developed[152,21] and applied to clinical studies[43,78], these methods result in 2D maps with shapes depending on the geometry of the input surfaces. The inability of these methods in producing a standardized 2D template makes them not suitable for quantitative analysis of VWT-Change.

In Chapter 7, we developed two surface density-equalizing map methods (DEM and SPDEM) for flattening simply-connected open surfaces based on density diffusion[52]. In particular, our

SPDEM method allows for the computation of area-preserving maps of surfaces onto a standardized convex planar domain such as a disk or a rectangle. However, the carotid mapping problem involves a non-convex target 2D domain, for which the bijectivity of SPDEM is not guaranteed. To overcome this issue, in this chapter we develop a novel method called the *density-equalizing reference map* (DERM) for computing area-preserving carotid flattening maps onto the 2D non-convex L-shaped domain. The proposed method extends the diffusion-based formulation and further combines it with the reference map technique[71,141,114] in solid mechanics. With the bijectivity ensured and the area distortion minimized, the flattening maps produced by our method provide an accurate and consistent collective representation of carotid surfaces, thereby enabling unbiased quantitative comparisons of the extent of carotid diseases among patients in population studies.

8.1 The reference map technique

Rycroft, Kamrin, and coworkers[71,141,114] developed the *reference map technique* (RMT) for simulating large-strain solid mechanics. The RMT uses the reference map as the basis for a fully Eulerian formulation, which provides a simple method to describe arbitrary deformations of a body in $\mathbb{R}^d$.

Suppose the material initially located at the position $\mathbf{X}$ of the body is moved to the position $\mathbf{x}$ at time t. The deformation can then be described by the *motion function* $\mathbf{x}(\mathbf{X}, t)$, which keeps track of the motion of the material initially at $\mathbf{X}$. The *reference map* is defined as the inverse of the motion

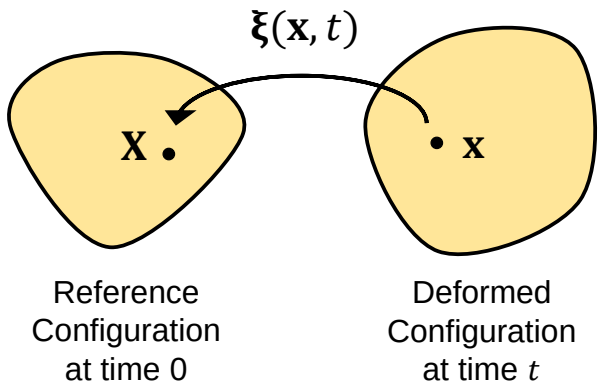

Figure 8.2: An illustration of the reference map. The reference map is the inverse of the motion function, which indicates the reference location of the material occupying $\mathbf{x}$ at time t.

function $\mathbf{X} = \xi(\mathbf{x}, t)$, which can be regarded as a vector field in the deformed body indicating the reference location at time 0 of the material occupying the position $\mathbf{x}$ at time t (Figure 8.2). In particular, $\xi(\mathbf{x}, 0) = \mathbf{x}$ as the initial configuration is undeformed. Now, note that for any tracer particle, the reference location of it is the same at all time t under the deformation. Therefore, we have

$$\dot{\xi} = 0, \tag{8.1}$$

which yields the advection equation

$$\frac{\partial \xi}{\partial t} + \mathbf{v} \cdot \nabla \xi = 0. \tag{8.2}$$

Solving the above equation as $t \to \infty$, we obtain the final reference map $\xi(\mathbf{x}_{\text{final}}, \infty)$ which gives the reference location of the material occupying the final position $\mathbf{x}_{\text{final}}$ of the deformed body. In the

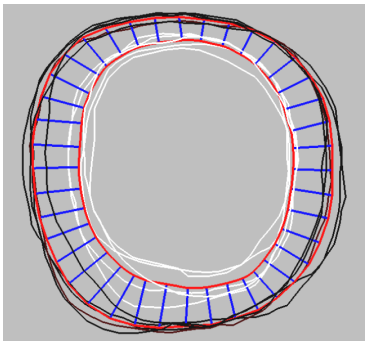

Figure 8.3: Reconstruction of the carotid surfaces from the ultrasound images. The white and black contours represent repeated segmentations of the lumen-intima boundary (LIB) and media-adventitia boundary (MAB) respectively, and the red contours represent the mean LIB and MAB. Each blue line connects a pair of correspondence points matched using the symmetric correspondence algorithm [22], and the local vessel-wall-plus-plaque thickness (VWT) is given by the distance between each correspondence pair.

2D case, we can obtain $\mathbf{x}_{\text{final}}$ from ξ by tracking the contour lines of the x- and y- coordinates of ξ.

8.2 Materials and methods

8.2.1 Study subjects and 3D ultrasound image acquisition

Here we consider ten subjects with carotid atherosclerosis recruited from The Premature Atherosclerosis Clinic and the Stroke Prevention Clinic at the London Health Science Center, London, Canada and the Stroke Prevention & Atherosclerosis Research Center, Robarts Research Institute, London, Canada. 20 carotid images were obtained by scanning each subject at baseline and two weeks later using a 3D ultrasound carotid imaging system [80]. All ten patients have stable atherosclerosis, and no physiological changes were expected to increase the inter-scan variability of VWT.

3D carotid surfaces were reconstructed from the ultrasound images as described in Refs. [21,19,25]

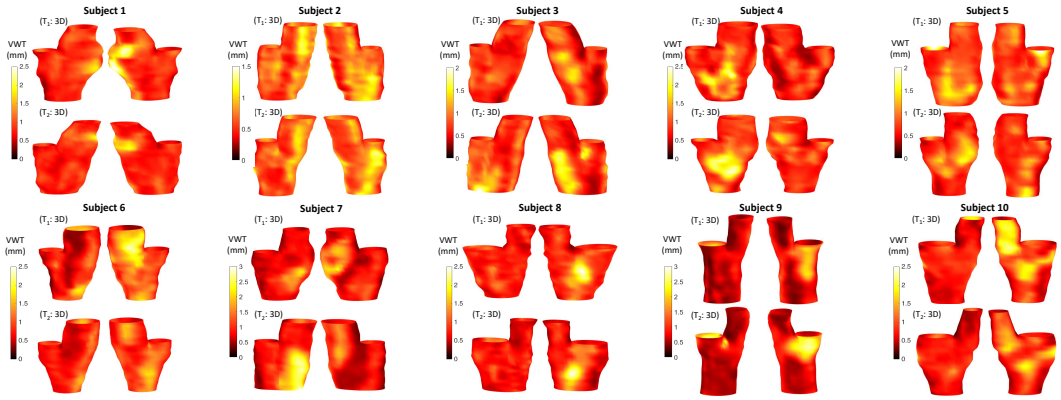

Figure 8.4: The 20 3D carotid surfaces color-coded and superimposed by their VWT distributions (with both front and back views). The models at baseline and follow-up are labeled as T_1 and T_2 respectively.

(Figure 8.3). The VWT was measured by computing the distance between each pair of corresponding points (i.e. the length of blue lines in Figure 8.3), and the 3D VWT map was constructed by superimposing the pointwise VWT on the MAB surface. The 20 resulting carotid surfaces are shown in Figure 8.4.

8.2.2 2D arc-length scaling (ALS) map

Here we briefly describe the ALS method[23] for flattening a carotid surface onto a 2D non-convex L-shaped domain. The input surface is first translated and rotated such that the bifurcation (BF) is at the origin, the longitudinal direction of the common carotid artery (CCA) is aligned with the z-axis, and the internal carotid artery (ICA) is located at the upper half space. The surface is then cut by two planes (Figure 8.1**a**) and unfolded to a 2D L-shaped non-convex domain (Figure 8.1**b**), with the ICA and the CCA respectively mapped to the top part and the bottom part of the planar domain. The method then rescales the arc-length of transverse contours segmented from 2D

transverse images resliced from 3D ultrasound images, so that the vertices on the input surface correspond to uniformly spaced grid points on the L-shaped domain. This results in a standardized 2D template for analyzing the carotid surfaces.

However, the ALS method does not minimize any local geometric distortion produced by the flattening process. In the following section, we further deform the ALS mapping result and generate an area-preserving 2D carotid template.

8.2.3 Area-preserving map via density-equalizing reference map (DERM)

Denote the 2D L-shaped domain obtained using the ALS method by D. Here, we deform D based on a prescribed density distribution. Denote the density at the location $\mathbf{x}$ on D at time t as $\rho(\mathbf{x}, t)$. We set $\rho(\mathbf{x}, 0)$ to be the area of each face of the carotid surface in order to achieve area-preservation under a diffusion-based deformation.

The following more general version of the diffusion process with diffusivity κ is considered:

$$\frac{\partial \rho}{\partial t} = \nabla \cdot (\kappa \nabla \rho) = \kappa \Delta \rho + \nabla \kappa \cdot \nabla \rho. \tag{8.3}$$

Note that here we introduce an additional differentiable function κ for handling the non-convex corner of the L-shaped domain. More specifically, to regularize the deformation around the non-convex corner, we slow down the diffusion process there by setting κ to be such that $\kappa \ll 1$ around

the non-convex corner and $\kappa \approx 1$ at the regions distant from it:

$$\kappa(x, y) = 1 - \left(1 - \frac{1}{\sqrt{a}}\right) e^{-\frac{(x-p)^2+(y-q)^2}{\sqrt{a}}}, \tag{8.4}$$

where (p, q) are the coordinates of the bifurcation point on D and a is the total area of D. Note that $\nabla\kappa$ in Eq. (8.3) can be expressed explicitly by taking partial derivatives on Eq. (8.4). On the boundary edges of D, we enforce the no-flux boundary condition $\mathbf{n} \cdot \nabla\rho = 0$ where $\mathbf{n}$ is the unit outward normal. This ensures that the diffusion occurs within D, and hence the subsequent diffusion-based deformation will not change the overall shape of it. The velocity field (7.4) is then given by

$$\mathbf{v}(\mathbf{x}, t) = -\frac{\kappa\nabla\rho}{\rho}. \tag{8.5}$$

Now, we treat D as a solid body and consider its deformation under the velocity field $\mathbf{v}(\mathbf{x}, t)$ induced by the density gradient. The reference map $\xi(\mathbf{x}, t)$ can be obtained by solving the advection equation

$$\frac{\partial\xi}{\partial t}(\mathbf{x}, t) + \mathbf{v}(\mathbf{x}, t) \cdot \nabla\xi(\mathbf{x}, t) = 0. \tag{8.6}$$

As $t \to \infty$, $\rho(\mathbf{x}, t)$ is equalized over D and the associated reference map field $\xi(\mathbf{x}_{\text{final}}, \infty)$ is a density-equalizing reference map. Denote the VWT at the location $\mathbf{X}$ on the initial 2D ALS map as $T(\mathbf{X})$. The VWT on the final area-preserving map is given by $T(\xi(\mathbf{X}, \infty))$. In other words, we can obtain the final area-preserving map $\mathbf{x}_{\text{final}}$ by considering the contour lines of constant x- and y- coordinates of ξ.

In the discrete case, suppose we use a rectangular grid consisting of $M \times N$ grid points with grid spacing h in both the x- and y-directions to represent D with the top left empty space of it included (which is just for simplicity of the discretization and can be omitted in the computations). Let the coordinates of the grid points be (ih, jh), where $0 \leq i \leq M-1$ and $0 \leq j \leq N-1$. We discretize the diffusion equation (8.3), the velocity field (8.5) and the advection equation (8.6) and solve them iteratively. Denote the step size as δt and the density at the grid point (ih, jh) at the n-th step as $\rho^n_{i,j}$. We can similarly write the discrete version of $\mathbf{v}$ and $\mathbf{X}$.

Note that κ and its derivatives κ_x, κ_y can be easily discretized. The diffusion equation (8.3) is solved by the implicit Euler method, with the central difference scheme used for approximating $\nabla\rho$:

$$\begin{aligned}\frac{\rho^n_{i,j} - \rho^{n-1}_{i,j}}{\delta t} =& \kappa_{i,j}\frac{\rho^n_{i+1,j} + \rho^n_{i-1,j} + \rho^n_{i,j+1} + \rho^n_{i,j-1} - 4\rho^n_{i,j}}{h^2} \\ &+ (\kappa_x)_{i,j}\frac{\rho^n_{i+1,j} - \rho^n_{i-1,j}}{2h} + (\kappa_y)_{i,j}\frac{\rho^n_{i,j+1} - \rho^n_{i,j-1}}{2h}.\end{aligned} \tag{8.7}$$

To enforce the the no-flux boundary condition for the diffusion equation, we use the following ghost node approach. At the four rectangular boundaries $(x, y) = (0, jh), ((M-1)h, jh), (ih, 0), (0, (N-1)h)$ where $0 \leq i \leq M-1$ and $0 \leq j \leq N-1$, and the two L-shaped boundaries $(x, y) = (p, jh), (ih, q)$ where $0 \leq ih \leq p$ and $0 \leq jh \leq q$, we suitably replace the terms $\rho^n_{i-1,j}, \rho^n_{i+1,j}, \rho^n_{i,j-1}, \rho^n_{i,j+1}$ on the right hand side in Eq. (8.7) by $\rho^n_{i,j}$ such that there is no density flux orthogonal to the boundaries, thereby maintaining the L-shape throughout the density-equalization process. By representing ρ^n as a column vector of size $MN \times 1$,we can simplify Eq. (8.7) as $\rho^n = A^{-1}\rho^{n-1}$, where A is an $MN \times MN$ matrix with

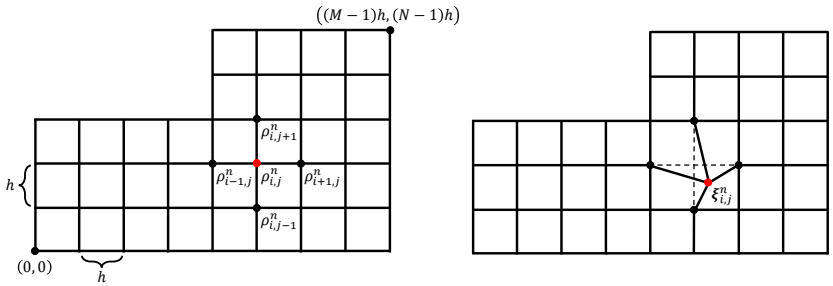

Figure 8.5: The discretization of the proposed DERM method. On a grid with spacing h, we update the density $\rho^n_{i,j}$ at the grid point (ih, jh) at the n-th step by solving the diffusion equation (8.7). $\rho^n_{i,j}$ depends on $\rho^n_{i\pm1,j}, \rho^n_{i,j\pm1}$ at the neighboring grid points at the n-th step, $\rho^{n-1}_{i,j}$ at (ih, jh) at the $(n-1)$-th step, as well as κ and its derivatives at (ih, jh). The density flux causes a velocity field (8.8), and we update the reference map $\xi^n_{i,j}$ using the advection equation (8.9).

$A = I - \delta t(\kappa\Delta + K_x + K_y)$, $K_x + K_y$ being the matrix representation of $\nabla\kappa\cdot\nabla$. A is a sparse matrix independent of n, and hence it is needed to be computed only once throughout the iterations.

As for the discretization of the velocity field (8.5), we again use the central difference scheme:

$$\begin{cases} (\mathbf{v}_x)^n_{i,j} = -\kappa_{i,j}\dfrac{\rho^n_{i+1,j} - \rho^n_{i-1,j}}{2h\rho^n_{i,j}}, \\ (\mathbf{v}_y)^n_{i,j} = -\kappa_{i,j}\dfrac{\rho^n_{i,j+1} - \rho^n_{i,j-1}}{2h\rho^n_{i,j}}. \end{cases} \tag{8.8}$$

We solve the advection equation (8.6) for updating $\xi(\mathbf{x}, t)$ using the following upwind method:

$$\frac{\xi^n_{i,j} - \xi^{n-1}_{i,j}}{\delta t} = \begin{cases} -(\mathbf{v}_x)^n_{i,j}\dfrac{\xi^{n-1}_{i,j} - \xi^{n-1}_{i-1,j}}{h} - (\mathbf{v}_y)^n_{i,j}\dfrac{\xi^{n-1}_{i,j} - \xi^{n-1}_{i,j-1}}{h} & \text{if } (\mathbf{v}_x)^n_{i,j} > 0 \text{ and } (\mathbf{v}_y)^n_{i,j} > 0, \\ -(\mathbf{v}_x)^n_{i,j}\dfrac{\xi^{n-1}_{i+1,j} - \xi^{n-1}_{i,j}}{h} - (\mathbf{v}_y)^n_{i,j}\dfrac{\xi^{n-1}_{i,j} - \xi^{n-1}_{i,j-1}}{h} & \text{if } (\mathbf{v}_x)^n_{i,j} \le 0 \text{ and } (\mathbf{v}_y)^n_{i,j} > 0, \\ -(\mathbf{v}_x)^n_{i,j}\dfrac{\xi^{n-1}_{i,j} - \xi^{n-1}_{i-1,j}}{h} - (\mathbf{v}_y)^n_{i,j}\dfrac{\xi^{n-1}_{i,j+1} - \xi^{n-1}_{i,j}}{h} & \text{if } (\mathbf{v}_x)^n_{i,j} > 0 \text{ and } (\mathbf{v}_y)^n_{i,j} \le 0, \\ -(\mathbf{v}_x)^n_{i,j}\dfrac{\xi^{n-1}_{i+1,j} - \xi^{n-1}_{i,j}}{h} - (\mathbf{v}_y)^n_{i,j}\dfrac{\xi^{n-1}_{i,j+1} - \xi^{n-1}_{i,j}}{h} & \text{if } (\mathbf{v}_x)^n_{i,j} \le 0 \text{ and } (\mathbf{v}_y)^n_{i,j} \le 0. \end{cases} \tag{8.9}$$

Figure 8.5 shows a schematic for the discretization. By solving (8.7), (8.8), (8.9) iteratively until

the density is fully equalized on D, we obtain the desired density-equalizing reference map ξ. Denote $\xi = (\xi_1, \xi_2)$. We obtain the associated area-preserving map $\mathbf{x}_{\text{final}}$ as follows. For every grid point (ih, jh) on the ALS map, where $i = 0, 1, \cdots, M-1$ and $j = 0, 1, \cdots, N-1$, the corresponding point of it in $\mathbf{x}_{\text{final}}$ is given by the intersection of the contour lines $\xi_1 = ih$ and $\xi_2 = jh$. In the discrete case, each contour line is represented as a piecewise linear curve. Therefore, to check whether a line segment $\{(x_1^k, y_1^k), (x_1^{k+1}, y_1^{k+1})\}$ of a ξ_1-contour intersects with another line segment $\{(x_2^l, y_2^l), (x_2^{l+1}, y_2^{l+1})\}$ of a ξ_2-contour, we solve the following system of four linear equations in four unknowns x^*, y^*, t_1, t_2:

$$\begin{cases} (x_1^{k+1} - x_1^k)t_1 &= x^* - x_1^k, \\ (y_1^{k+1} - y_1^k)t_1 &= y^* - y_1^k, \\ (x_2^{l+1} - x_2^l)t_2 &= x^* - x_2^l, \\ (y_2^{l+1} - y_2^l)t_2 &= y^* - y_2^l. \end{cases} \tag{8.10}$$

The two line segments intersect at (x^*, y^*) if and only if $t_1, t_2 \in [0, 1]$. Hence, we can obtain the area-preserving map $\mathbf{x}_{\text{final}}$ by tracking the intersection points of all pairwise contour lines.

To choose a suitable step size δt, note that the implicit Euler scheme for the diffusion equation is unconditionally stable. By performing a dimensional analysis on Eq. (7.3), we can see that an appropriate dimension of δt is (length)^2. Also, note that δt should be independent of the magnitude of ρ as the density diffusion process is invariant under uniform rescaling of ρ. Therefore, we set

$$\delta t = \frac{\text{std}(\rho)}{\text{mean}(\rho)} \times ac, \tag{8.11}$$

where c is a dimensionless constant. The convergence criterion is set to be $\frac{\mathrm{sd}(\rho^n)}{\mathrm{mean}(\rho^n)} \leq \varepsilon$, where ε is the error threshold. Algorithm 8.1 summarizes the proposed density-equalizing reference map (DERM) method for computing area-preserving carotid mapping.

Algorithm 8.1: Area-preserving carotid mapping via density-equalizing reference map (DERM)

Input: A carotid surface S, the error threshold ε, the maximum number of iterations $n_{\max}$.

Output: An area-preserving map $\mathbf{x}_{\text{final}}$ on the 2D non-convex L-shaped domain.

1 Use the ALS method[23] to compute an initial map $f : S \to \mathbb{R}^2$ onto the 2D L-shaped domain;
2 Set the density ρ as the area of every face of S;
3 Set $\delta t = \frac{\mathrm{std}(\rho)}{\mathrm{mean}(\rho)} \times ac$;
4 Compute $A = I - \delta t(\kappa\Delta + K_x + K_y)$;
5 Set $n = 0$ and $\rho^0 = \rho$;
6 **repeat**
7 Solve $\rho^{n+1} = A^{-1}\rho^n$;
8 Compute the velocity field $\mathbf{v}$ using Eq. (8.8);
9 Update the reference map ξ using Eq. (8.9);
10 Update $n = n + 1$;
11 **until** $\frac{sd(\rho^n)}{mean(\rho^n)} \leq \varepsilon$ *or* $n \geq n_{\max}$;
12 Obtain the desired map $\mathbf{x}_{\text{final}}$ by tracking the intersections of the contour lines $\xi_1 = ih$ and $\xi_2 = jh$ of $\xi = (\xi_1, \xi_2)$ for all i, j;

Recall that the method by Gastner and Newman (GN)[52] iteratively solves Eq. (7.3) with uniform diffusivity, obtains the velocity field (7.4), and tracks the displacement of every tracer particle by Eq. (7.5). To track the displacement of a tracer, an interpolation of the velocity field at its current location is required at every iteration. By contrast, DERM keeps track of the reference map field by Eq. (8.6), which is fully Eulerian. Therefore, no interpolation is needed throughout the iterations.

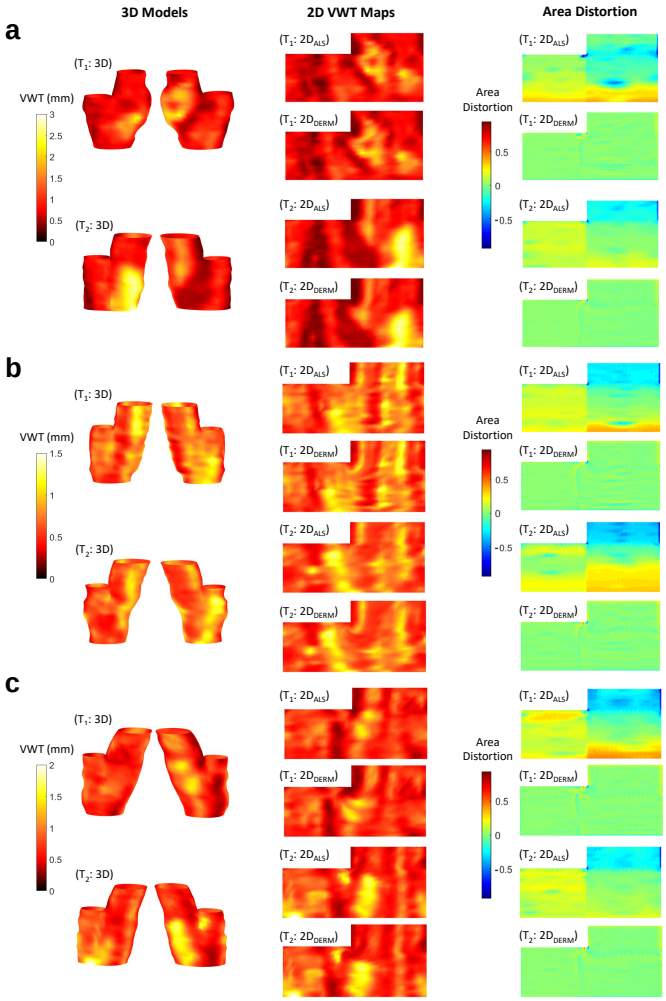

Figure 8.6: Carotid flattening by ALS[23] and our area-preserving DERM method. a-c, The results for three subjects. The carotid models constructed from the baseline and follow-up 3D ultrasound images for each subject are denoted by T_1 and T_2 respectively. The left column shows the 3D carotid surfaces color-coded and superimposed by their vessel-wall-plus-plaque thickness (VWT) distributions (with both front and back views). The middle column shows the 2D maps generated by ALS and DERM. The right column shows the area distortion of the flattening maps.

8.3 Results

The ALS mapping method is implemented in C++. The proposed DERM method is also implemented in C++ with OpenMP parallelization (with grid spacing $h = 1$, maximum number of

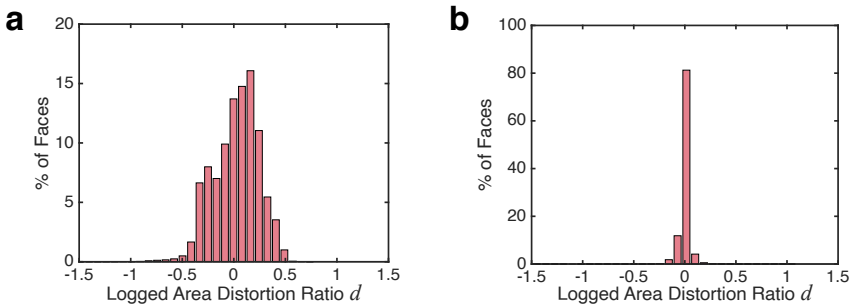

Figure 8.7: Histograms of the logged area distortion ratio d. d is evaluated on the 150360 quadrilateral faces (7518 per carotid model $\times$ 20 carotid models) in the ALS maps and the area-preserving DERM maps. **a**, The histogram for ALS. **b**, The histogram for DERM.

iterations $n_{\max} = 500$, step size constant $c = 0.01$, and error threshold $\varepsilon = 10^{-3}$). The biconjugate gradient stabilized method (`BiCGSTAB`) in the C++ library Eigen is used for solving the sparse linear systems. The experiments are performed on a PC with an Intel i7-6700K quad-core processor and 16 GB RAM. For each arterial model, the ALS initialization takes 1 second and DERM takes 8 seconds.

Figure 8.6 (left) shows the front and back views of six carotid surfaces at baseline and follow-up from three subjects. The mapping results by ALS and DERM for three subjects are shown in Figure 8.6 (middle). To quantitatively assess the area distortion of the two methods, we compute the following logged area distortion ratio associated with each quadrilateral face of the models:

$$d = \log_e \frac{\text{Area of the face on the flattened map}}{\text{Area of the face on the 3D carotid model}}. \tag{8.12}$$

Note that d represents enlargement and shrinkage in an equal magnitude. For a perfectly area-preserving map, we have $d = 0$ for all faces. A positive d at a local region indicates that the

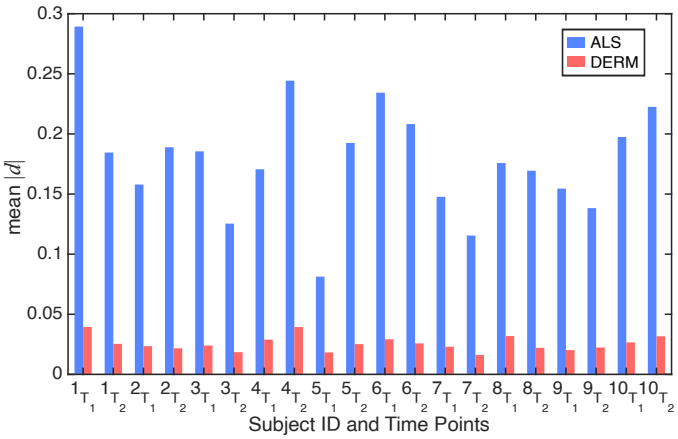

Figure 8.8: Average absolute area distortion mean$(|d|)$ for ALS and DERM for the 20 carotid surfaces. For each subject, the carotid models at baseline and follow-up are denoted by T_1 and T_2 respectively.

region is enlarged on the 2D domain, while a negative d indicates that the region is shrunken. The symmetric property of d makes it easier to interpret than the linear ratio. Figure 8.6 (right) shows the 2D maps with d color-coded and superimposed, from which it can be observed that the area distortion exhibited in the ALS initialization is effectively corrected by DERM. Figure 8.7 shows the histograms of d for ALS and DERM evaluated on the 150360 quadrilateral elements of the 20 carotid surfaces (7518 per surface $\times$ 20 surfaces). The histogram for DERM is much more concentrated at $d = 0$, which indicates that the area distortion associated with DERM is smaller than that associated with ALS. We further calculate the average value of $|d|$ over the 2D maps produced by ALS and DERM for each of the 20 carotid models (Figure 8.8). The result shows that DERM reduces the average absolute area distortion by over 80%, and a two-sample t-test shows that the reduction is significant ($P < 10^{-6}$).

Besides minimizing the area distortion, DERM is advantageous in preventing overlaps in the 2D

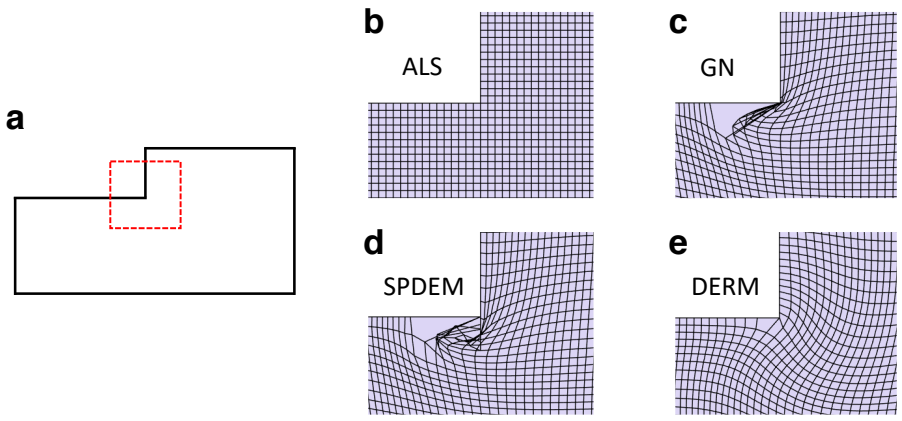

Figure 8.9: Comparison between the flattening maps produced by four methods. a, We consider zooming into the non-convex corner of the 2D L-shaped domain. **b-e**, The mapping results at that region produced by ALS [23], GN [52], SPDEM (Chapter 7) and DERM. Our method is advantageous in adjusting for the geometric variability of the carotid surfaces without causing any overlaps.

non-convex domain. Here, we compare DERM with ALS, GN and SPDEM. For both GN and SPDEM, we set the density to be the face area for computing area-preserving flattening maps and enforce the no-flux boundary condition for preserving the L-shape throughout the iterations. By zooming into the non-convex corner of the 2D maps produced by the four methods (Figure 8.9), we observe that GN and SPDEM result in overlaps around the non-convex corner due to the large density gradient between the ICA and CCA, while DERM is free of overlaps because of the use of a non-constant diffusion coefficient κ in the diffusion process. We quantify the total overlapping area A_{overlap} for each of the 20 carotid models and for each method by taking the difference between the sum of the area of all quadrilateral faces in the 2D map and the area enclosed by the boundary of the L-shaped domain. The mean and the standard deviation of A_{overlap} for GN and SPDEM are 2.1 ± 1.3 and 2.8 ± 2.1 mm^2 respectively. By contrast, $A_{\text{overlap}} = 0$ for DERM for all surfaces.

The average area distortion mean$(|d|)$ and the total overlapping area A_{overlap} for the four

Carotid model	ALS [23]		GN [52]		SPDEM (Ch. 7)		DERM	
	mean($\lvert d \rvert$)	A_{overlap}	mean($\lvert d \rvert$)	A_{overlap}	mean($\lvert d \rvert$)	A_{overlap}	mean($\lvert d \rvert$)	A_{overlap}
Subject 1 Time 1	0.29	0	0.15	4.27	0.04	8.88	0.04	0
Subject 1 Time 2	0.18	0	0.07	1.97	0.03	4.57	0.03	0
Subject 2 Time 1	0.16	0	0.06	2.47	0.02	3.55	0.02	0
Subject 2 Time 2	0.19	0	0.06	2.43	0.03	3.03	0.02	0
Subject 3 Time 1	0.19	0	0.08	2.87	0.02	4.52	0.02	0
Subject 3 Time 2	0.13	0	0.05	2.18	0.02	3.09	0.02	0
Subject 4 Time 1	0.17	0	0.05	1.29	0.02	1.65	0.03	0
Subject 4 Time 2	0.24	0	0.09	3.01	0.03	4.82	0.04	0
Subject 5 Time 1	0.08	0	0.02	0.36	0.01	0.49	0.02	0
Subject 5 Time 2	0.19	0	0.05	1.89	0.02	2.61	0.03	0
Subject 6 Time 1	0.23	0	0.11	4.58	0.03	5.40	0.03	0
Subject 6 Time 2	0.21	0	0.08	3.19	0.03	3.39	0.03	0
Subject 7 Time 1	0.15	0	0.03	0.83	0.02	1.21	0.02	0
Subject 7 Time 2	0.12	0	0.04	1.49	0.01	2.31	0.02	0
Subject 8 Time 1	0.18	0	0.05	1.65	0.02	1.57	0.03	0
Subject 8 Time 2	0.17	0	0.06	2.97	0.02	1.57	0.02	0
Subject 9 Time 1	0.15	0	0.02	0	0.02	0.22	0.02	0
Subject 9 Time 2	0.14	0	0.02	0	0.02	0	0.02	0
Subject 10 Time 1	0.20	0	0.04	1.87	0.02	1.01	0.03	0
Subject 10 Time 2	0.22	0	0.08	3.32	0.03	1.91	0.03	0

Table 8.1: The area distortion mean($\lvert d \rvert$) and the total overlapping area A_{overlap} for ALS [23], GN [52], SPDEM (Chapter 7), and the proposed DERM method for the 20 carotid models.

methods (ALS, GN, SPDEM, and DERM) are recorded in Table 8.1. It can be observed that DERM achieves a reduction in area distortion comparable to SPDEM and outperforms it in terms of the bijectivity.

After assessing the area-preservation and bijectivity of DERM, we consider the improvement of the accuracy of the plaque size representation in the 2D carotid template produced by DERM. As the latest European Society of Hypertension (ESH) and European Society of Cardiology (ESC) hypertension guidelines [89] lists intima–media thickness (IMT) $>$ 0.9 mm as a risk factor of asymptomatic organ damage, we define the plaque regions in a carotid model to be the regions with VWT $>$ 0.9 mm (see Figure 8.10 for the binary 2D VWT maps constructed for two carotid models by ALS and DERM, with white and black representing the plaque regions and the background respectively).

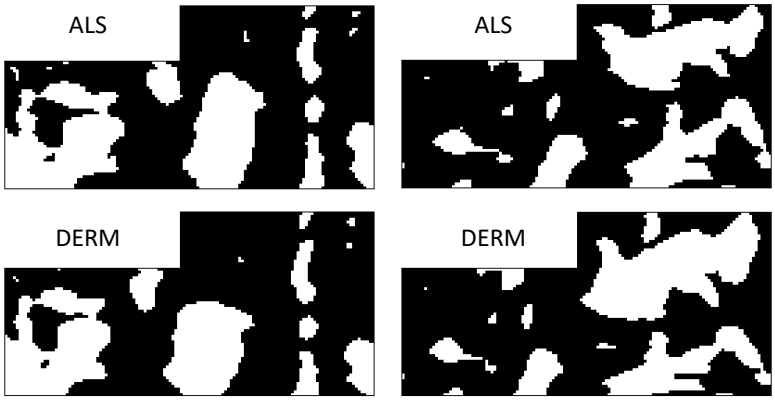

Figure 8.10: Binary 2D VWT maps by ALS and DERM, with white representing the plaque regions (i.e. with VWT > 0.9 mm) and black otherwise. Each column corresponds to one carotid model.

To evaluate the accuracy of the plaque area change for each patient from baseline to follow-up exhibited in the 2D VWT maps produced by ALS and DERM, we consider the plaque area change error associated with the two methods:

$$e_{\Delta \mathrm{PA}_i} = \Delta \mathrm{PA}_i - \Delta \mathrm{PA}_{3\mathrm{D}}, \tag{8.13}$$

where $i \in \{\mathrm{ALS}, \mathrm{DERM}\}$ represents the mapping method (ALS or DERM), $\Delta \mathrm{PA}_i$ is plaque area change estimated using method i and $\Delta \mathrm{PA}_{3\mathrm{D}}$ is the plaque area change obtained on the 3D VWT map. Computing the plaque area change errors for all 10 subjects, we have $|e_{\Delta \mathrm{PA}_{\mathrm{ALS}}}| = 13.8 \pm 24.9\ \mathrm{mm}^2$ and $|e_{\Delta \mathrm{PA}_{\mathrm{DERM}}}| = 7.2 \pm 12.9\ \mathrm{mm}^2$. The mean error is reduced by 48% using DERM, and the difference is statistically significant ($P = 0.01$). This suggests that DERM is advantageous in reducing the error in plaque size representation.

Note that the location and the size of plaques are the two major determinants of stroke

risk [106,126,145]. A previous study has shown that ALS provides good anatomic correspondence for carotid arteries [18]. While we have shown that DERM significantly reduces the error in plaque region area estimation when compared to ALS, one may ask whether the anatomic correspondence would be negatively affected under DERM. As no physiological change is expected between the two scanning sessions for the ten subjects, we assess the anatomic correspondence in terms of the inter-scan variability by computing ΔVWT between the 2D maps of the two sessions. More specifically, we compare ΔVWTs associated with ALS and DERM based on the following three parameters:

- Percentage of vertices with $\Delta\text{VWT} \leq 0.35$ mm for each subject.
- Visual comparison between the mean ΔVWT maps generated by ALS and DERM for the ten subjects.
- The subject-based mean $|\Delta\text{VWT}|$ for each of the ten subjects.

Here, we choose 0.35 mm as the threshold because the axial, lateral and elevational resolutions of the 3D imaging system are 0.6, 0.8 and 2 mm repsectively at the depth of 40 mm, which is approximately the depth of the carotid artery [14]. The average in-plane resolution is 0.7 mm, and half of which (0.35 mm) can be considered as a small change. For ALS the percentage is $80.7\% \pm 6.7\%$, and for DERM the percentage is $80.9\% \pm 6.4\%$. A two-sample paired *t*-test shows no significant difference between them ($P = 0.85$). Figure 8.11 shows the mean ΔVWT maps generated by ALS and DERM for the ten subjects, from which it can be observed that the two ΔVWT patterns are similar. As for the subject-based mean $|\Delta\text{VWT}|$, the two-sample paired *t*-test shows no significant difference between the results produced by ALS and DERM ($P = 0.80$). The above experiments

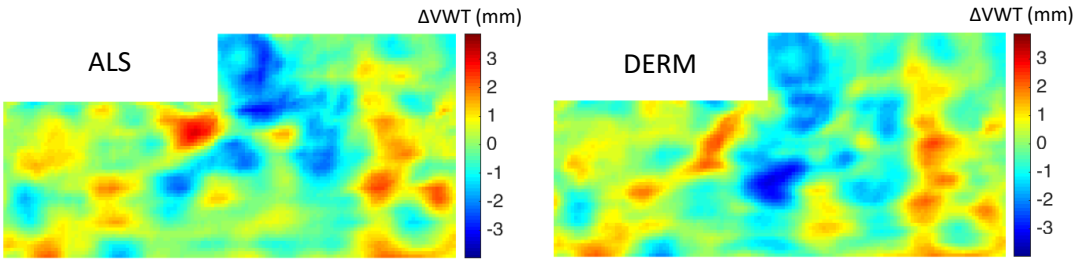

Figure 8.11: The average ΔVWT map between the baseline and the follow-up generated by ALS and DERM for the ten subjects.

suggest that DERM effectively improves the accuracy of plaque area change estimation without compromising inter-scan reproducibility, thereby facilitating the collective analysis of the location and the size of plaques on the standardized 2D carotid template.

8.4 Discussion

One limitation of GN and SPDEM is the lack of bijectivity for non-convex 2D domains. Our proposed DERM method overcomes this limitation by allowing for a non-constant diffusivity that effectively regularizes the density gradient around the non-convex corner of the 2D L-shaped domain. Another limitation of the two methods is that they track the deformation of each node on the domain individually without considering the neighboring spatial information, and hence a large deformation will result in an unsmooth mapping. By integrating the reference map technique (RMT) with the density-equalization process, DERM is capable of generating a smooth and area-preserving flattening map.

Although the carotid surfaces in our experiments are generated from 3D carotid ultrasound images, the proposed DERM method is equally applicable to any imaging modality that allows the

segmentation of the LIB and MAB. Also, while we have primarily focused on carotid mapping, the proposed DERM method can be applied for flattening other organs such as brain ventricles and kidneys [32] for facilitating the interpretation of spatial distributions on the surfaces. The resulting standardized 2D template will allow for unbiased quantitative comparisons of the organs.

One should always generalize.

Carl Jacobi

9

Volumetric density-equalizing map

After studying the use of the density-equalizing maps for mapping surfaces onto 2D domains, it is natural to ask if it is possible to extend density-equalizing maps to 3D. In this chapter, we develop a novel method called the *volumetric density-equalizing reference map* (VDERM) that produces volumetric deformations based on a prescribed density distribution. Our

method combines and extends the method by Gastner and Newman (GN)[52] and the reference map technique (RMT)[71,141,114]. The proposed method can be utilized for volumetric data visualization, shape modeling, adaptive remeshing etc. To the best of our knowledge, this is the first work on higher dimensional density-equalizing maps. Furthermore, all prior works on density-equalizing maps only focused on the use of them for producing a single final output. With the observation that density-equalizing map is a continuous deformation, we introduce the first use of density-equalizing map for time-dependent applications such as shape morphing.

9.1 Formulation

9.1.1 Iterative scheme for volumetric deformation

Let $D \subset \mathbb{R}^3$ be a rectangular solid domain discretized as a $L \times M \times N$ 3D grid with grid spacing h in all the x-, y- and z-directions. The coordinates of the grid points are given by (ih, jh, kh), where $0 \leq i \leq L-1, 0 \leq j \leq M-1, 0 \leq k \leq N-1$. Let $\rho^0 = \rho^0(\mathbf{x})$ be a prescribed density defined in D, with discretization given by $\rho^0_{i,j,k} = \rho^0(ih, jh, kh)$ for all i, j, k. Denote $\rho(\mathbf{x}, t)$ as the density at time t, with $\rho(\mathbf{x}, 0) = \rho^0(\mathbf{x})$. The diffusion of ρ follows from the diffusion equation

$$\frac{\partial \rho}{\partial t}(\mathbf{x}, t) = \Delta \rho(\mathbf{x}, t). \tag{9.1}$$

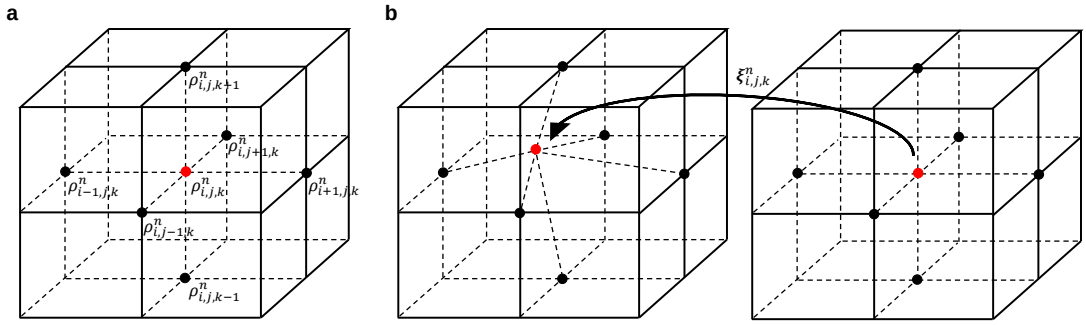

Figure 9.1: A illustration of density diffusion and reference map update in our proposed volumetric density-equalizing reference map method. a, The diffusion of ρ is computed iteratively by solving Eq. (9.2) based on the six neighboring nodes. **b**, The density gradient is then used for updating ξ via Eq. (9.7).

We discretize Eq. (9.1) using the backward Euler scheme (Figure 9.1):

$$\frac{\rho^n_{i,j,k} - \rho^{n-1}_{i,j,k}}{\delta t} = \frac{1}{h^2}\left(\rho^n_{i+1,j,k} + \rho^n_{i-1,j,k} + \rho^n_{i,j+1,k} + \rho^n_{i,j-1,k} + \rho^n_{i,j,k+1} + \rho^n_{i,j,k-1} - 6\rho^n_{i,j,k}\right), \tag{9.2}$$

where δt is the timestep to be determined, and $n = t/\delta t$ is the iteration number. We further simplify Eq. (9.2) as

$$\rho^n = (I - \delta t \Delta)^{-1} \rho^{n-1}, \tag{9.3}$$

where Δ is an $LMN \times LMN$ seven-point Laplacian stencil. Here, $I - \delta t \Delta$ is a sparse symmetric positive definite matrix independent of n. Hence, after setting up the matrix $A = I - \delta t \Delta$ based on the backward Euler scheme (9.2) and preconditioning the system once, we can efficiently solve Eq. (9.3) at each iteration with both the matrix A and the preconditioner reused.

The density gradient of ρ induces a velocity field

$$\mathbf{v}(\mathbf{x}, t) = -\frac{\nabla\rho(\mathbf{x}, t)}{\rho(\mathbf{x}, t)}, \tag{9.4}$$

which can be discretized using the following central difference scheme:

$$\begin{cases} (\mathbf{v}_x)^n_{i,j,k} = -\dfrac{\rho^n_{i+1,j,k} - \rho^n_{i-1,j,k}}{2h\rho^n_{i,j,k}}, \\ (\mathbf{v}_y)^n_{i,j,k} = -\dfrac{\rho^n_{i,j+1,k} - \rho^n_{i,j-1,k}}{2h\rho^n_{i,j,k}}, \\ (\mathbf{v}_z)^n_{i,j,k} = -\dfrac{\rho^n_{i,j,k+1} - \rho^n_{i,j,k-1}}{2h\rho^n_{i,j,k}}. \end{cases} \tag{9.5}$$

We then track the deformation of D under the velocity field using the reference map $\xi(\mathbf{x}, t)$, discretized as $\xi^n_{i,j,k} = \xi((ih, jh, kh), n\delta t)$. ξ can be computed by solving the advection equation

$$\frac{\partial\xi}{\partial t}(\mathbf{x}, t) - \frac{\nabla\rho(\mathbf{x}, t)}{\rho(\mathbf{x}, t)} \cdot \nabla\xi(\mathbf{x}, t) = 0, \tag{9.6}$$

which we discretize using the second-order upwind scheme:

$$\frac{\xi^n_{i,j,k} - \xi^{n-1}_{i,j,k}}{\delta t} = d_x + d_y + d_z, \tag{9.7}$$

where

$$d_x = \begin{cases} (\mathbf{v}_x)^n_{i,j,k} d^-_x & \text{if } (\mathbf{v}_x)^n_{i,j,k} > 0, \\ (\mathbf{v}_x)^n_{i,j,k} d^+_x & \text{if } (\mathbf{v}_x)^n_{i,j,k} \leq 0, \end{cases} \tag{9.8}$$

$$d_y = \begin{cases} (\mathbf{v}_y)^n_{i,j,k} d_y^- & \text{if } (\mathbf{v}_y)^n_{i,j,k} > 0, \\ (\mathbf{v}_y)^n_{i,j,k} d_y^+ & \text{if } (\mathbf{v}_y)^n_{i,j,k} \le 0, \end{cases} \tag{9.9}$$

$$d_z = \begin{cases} (\mathbf{v}_z)^n_{i,j,k} d_z^- & \text{if } (\mathbf{v}_z)^n_{i,j,k} > 0, \\ (\mathbf{v}_z)^n_{i,j,k} d_z^+ & \text{if } (\mathbf{v}_z)^n_{i,j,k} \le 0, \end{cases} \tag{9.10}$$

with $d_x^-, d_x^+, d_y^-, d_y^+, d_z^-, d_z^+$ being the three-point finite-difference discretization of the spatial derivative $\nabla \xi$:

$$\begin{cases} d_x^- &= \frac{3\xi^{n-1}_{i,j,k} - 4\xi^{n-1}_{i-1,j,k} + \xi^{n-1}_{i-2,j,k}}{2h}, \\ d_x^+ &= \frac{-\xi^{n-1}_{i+2,j,k} + 4\xi^{n-1}_{i+1,j,k} - 3\xi^{n-1}_{i,j,k}}{2h}, \\ d_y^- &= \frac{3\xi^{n-1}_{i,j,k} - 4\xi^{n-1}_{i,j-1,k} + \xi^{n-1}_{i,j-2,k}}{2h}, \\ d_y^+ &= \frac{-\xi^{n-1}_{i,j+2,k} + 4\xi^{n-1}_{i,j+1,k} - 3\xi^{n-1}_{i,j,k}}{2h}, \\ d_z^- &= \frac{3\xi^{n-1}_{i,j,k} - 4\xi^{n-1}_{i,j,k-1} + \xi^{n-1}_{i,j,k-2}}{2h}, \\ d_z^+ &= \frac{-\xi^{n-1}_{i,j,k+2} + 4\xi^{n-1}_{i,j,k+1} - 3\xi^{n-1}_{i,j,k}}{2h}. \end{cases} \tag{9.11}$$

The above discretization works for $2 \le i \le L-1, 2 \le j \le M-1, 2 \le k \le N-1$. For $i = 1, L-2$, we use the two-point difference

$$\begin{cases} d_x^- &= \frac{\xi^{n-1}_{i,j,k} - \xi^{n-1}_{i-1,j,k}}{h}, \\ d_x^+ &= \frac{\xi^{n-1}_{i+1,j,k} - \xi^{n-1}_{i,j,k}}{h}. \end{cases} \tag{9.12}$$

Similarly, for $j = 1, M-2$, we use

$$\begin{cases} d_y^- &= \frac{\xi_{i,j,k}^{n-1} - \xi_{i,j-1,k}^{n-1}}{h}, \\ d_y^+ &= \frac{\xi_{i,j+1,k}^{n-1} - \xi_{i,j,k}^{n-1}}{h}, \end{cases} \tag{9.13}$$

and for $k = 1, N-2$ we use

$$\begin{cases} d_z^- &= \frac{\xi_{i,j,k}^{n-1} - \xi_{i,j,k-1}^{n-1}}{h}, \\ d_z^+ &= \frac{\xi_{i,j,k+1}^{n-1} - \xi_{i,j,k}^{n-1}}{h}. \end{cases} \tag{9.14}$$

To choose a suitable timestep δt, we note that Eq. (9.2) is unconditionally stable. A necessary condition for the convergence of the second-order upwind scheme (9.7) is given by the Courant-Friedrichs-Lewy (CFL) condition [31], which states that the numerical domain of dependence must include the physical domain of dependence. Consider multiplying δt on both sides in Eq. (9.7) and making $\xi_{i,j,k}^n$ as the subject. The remaining terms involving $\xi_{i,j,k}^{n-1}$ are

$$\begin{aligned} &\xi_{i,j,k}^{n-1} - \delta t \left(\frac{3\xi_{i,j,k}^{n-1} |(\mathbf{v}_x)_{i,j,k}^n|}{2h} + \frac{3\xi_{i,j,k}^{n-1} |(\mathbf{v}_y)_{i,j,k}^n|}{2h} + \frac{3\xi_{i,j,k}^{n-1} |(\mathbf{v}_z)_{i,j,k}^n|}{2h} \right) \\ =&\xi_{i,j,k}^{n-1} \left(1 - \delta t \left(\frac{3|(\mathbf{v}_x)_{i,j,k}^n|}{2h} + \frac{3|(\mathbf{v}_y)_{i,j,k}^n|}{2h} + \frac{3|(\mathbf{v}_z)_{i,j,k}^n|}{2h} \right) \right). \end{aligned} \tag{9.15}$$

Here we have the absolute signs because for a negative $(\mathbf{v}_x)_{i,j,k}^n (\mathbf{v}_y)_{i,j,k}^n$, or $(\mathbf{v}_z)_{i,j,k}^n$, the corresponding d_x^+, d_y^+, or d_z^+ involves a negative sign in the coefficient of $\xi_{i,j,k}^{n-1}$. Now, as the full numerical domain of dependence of the scheme must contain the physical domain of dependence,

the coefficient of $\xi_{i,j,k}^{n-1}$ must satisfy the following condition for all i, j, k, n:

$$0 \leq 1 - \delta t \left(\frac{3|(\mathbf{v}_x)_{i,j,k}^n|}{2h} + \frac{3|(\mathbf{v}_y)_{i,j,k}^n|}{2h} + \frac{3|(\mathbf{v}_z)_{i,j,k}^n|}{2h} \right) \leq 1, \tag{9.16}$$

which implies that

$$\delta t \left(\frac{3|(\mathbf{v}_x)_{i,j,k}^n|}{2h} + \frac{3|(\mathbf{v}_y)_{i,j,k}^n|}{2h} + \frac{3|(\mathbf{v}_z)_{i,j,k}^n|}{2h} \right) \leq 1. \tag{9.17}$$

As diffusion smooths out sharp gradients in density, the magnitude of the velocity field decreases as n increases and we have

$$\delta t \leq \frac{2h}{3 \max_{i,j,k}(|(\mathbf{v}_x)_{i,j,k}^0| + |(\mathbf{v}_y)_{i,j,k}^0| + |(\mathbf{v}_z)_{i,j,k}^0|)}. \tag{9.18}$$

Therefore, we can take the upper bound and set the timestep δt as follows:

$$\delta t = \frac{2h}{3 \max_{i,j,k}(|(\mathbf{v}_x)_{i,j,k}^0| + |(\mathbf{v}_y)_{i,j,k}^0| + |(\mathbf{v}_z)_{i,j,k}^0|)}. \tag{9.19}$$

As $t \to \infty$, the density $\rho(\mathbf{x}, t)$ is equalized over the entire domain D, and hence the associated reference map $\xi_{\text{final}}(\mathbf{x}) = \xi(\mathbf{x}, \infty)$ is a density-equalizing reference map. In the discrete case, we use the following convergence criterion:

$$\frac{\|\rho^n - \rho^{n-1}\|_2}{\text{mean}(\rho^{n-1})} \leq \varepsilon, \tag{9.20}$$

where ε is the error threshold. The proposed algorithm is summarized in Algorithm 9.1.

Algorithm 9.1: Volumetric density-equalizing reference map (VDERM)

Input: A volumetric domain $D \subset \mathbb{R}^3$ of size $L \times M \times N$, a prescribed density ρ^0, the error threshold ε, and the maximum number of iterations $n_{\max}$.

Output: A density-equalizing reference map ξ_{final}.

Set $\delta t = \frac{2h}{3 \max_{i,j,k}(|(\mathbf{v}_x)^0_{i,j,k}|+|(\mathbf{v}_y)^0_{i,j,k}|+|(\mathbf{v}_z)^0_{i,j,k}|)}$;
Compute $A = I - \delta t \Delta$;
Set $n = 0$;
repeat
    Update $n = n + 1$;
    Solve $\rho^n = A^{-1}\rho^{n-1}$ as discussed in Eq. (9.3);
    Compute the velocity field $\mathbf{v}^n$ using Eq. (9.5);
    Update the reference map ^{n} using Eq. (9.7);
until $\frac{\|\rho^n - \rho^{n-1}\|_2}{mean(\rho^{n-1})} \leq \varepsilon$ *or* $n \geq n_{\max}$;
Obtain $\xi_{\text{final}} = \xi^n$;

The volumetric density-equalizing reference map provides us with the information of the diffusion-based volumetric deformation. In particular, if we denote $\xi_{\text{final}} = (\xi_1, \xi_2, \xi_3)$, one can obtain the forward mapping of the grid points of D by tracking the intersections of the contour planes $\xi_1 = ih$, $\xi_2 = jh$ and $\xi_3 = kh$. More specifically, after computing the density-equalizing reference map, we build an interpolant for each of the x-, y-, and z-coordinates of the undeformed grid using (ξ_1, ξ_2, ξ_3), where the ξ values are the abscissa and the x, y, z values are the ordinate. In other words, we have three interpolant functions $I_1(x, y, z), I_2(x, y, z), I_3(x, y, z)$ where

$$\begin{cases} I_1(\xi_1(ih, jh, kh), \xi_2(ih, jh, kh), \xi_3(ih, jh, kh)) = ih, \\ I_2(\xi_1(ih, jh, kh), \xi_2(ih, jh, kh), \xi_3(ih, jh, kh)) = jh, \\ I_3(\xi_1(ih, jh, kh), \xi_2(ih, jh, kh), \xi_3(ih, jh, kh)) = kh, \end{cases} \tag{9.21}$$

for all i, j, k. The three interpolants allow us to compute a forward deformation of the given domain. In practice, we we compute a Delaunay triangulation of the given points and then perform a linear interpolation in each tetrahedral element. It is noteworthy that throughout the density-equalizing iterations it suffices to work on a grid, and the interpolation is only needed once at the end. On the contrary, a direct extension of the original density-equalizing map [52] would involve interpolating the velocity field at every iteration. Therefore, the use of the reference map technique here is advantageous.

9.1.2 Boundary conditions

Note that boundary conditions are needed for solving Eq. (9.2). Three possible boundary conditions are discussed below.

No-flux boundary condition

To keep the boundary shape of the 3D domain unchanged, we enforce the following no-flux boundary condition

$$\mathbf{n} \cdot \nabla \rho = 0 \tag{9.22}$$

at the boundaries ∂D, where $\mathbf{n}$ is the outward unit normal. This condition ensures that the density gradient at the boundaries is zero in the normal direction, thereby keeping all the boundary planes to be planar throughout the density-equalization process.

In the discrete case, the above no-flux boundary condition can be incorporated in the diffusion

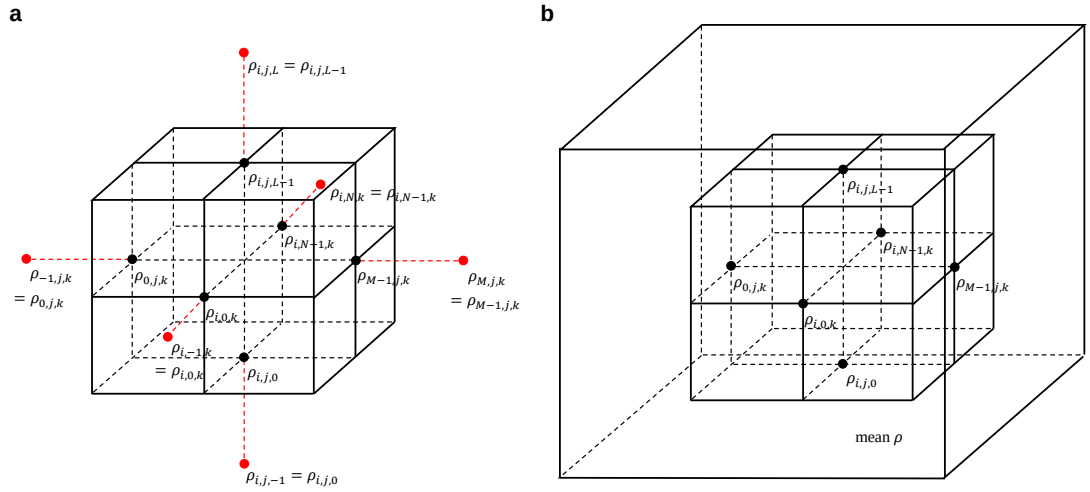

Figure 9.2: Three possible choices of boundary conditions. a, The no-flux boundary condition can be imposed by placing a ghost node (red) at the neighborhood of each boundary node on ∂D, with the density being the same as that at the boundary node. No density flux will be orthogonal to the boundary faces. **b**, The free boundary condition can be imposed by creating a larger solid domain $\widetilde{D}$ and setting the density at $\widetilde{D} \setminus D$ (the "sea") to be the average value of ρ. This ensures that the original boundary ∂D can deform freely.

equation (9.2) using the following ghost node approach. At the six boundary faces $i = 0, i = L - 1$, $j = 0, j = M - 1, k = 0, k = N - 1$, we replace the terms $\rho^n_{i-1,j,k}, \rho^n_{i+1,j,k}, \rho^n_{i,j-1,k}, \rho^n_{i,j+1,k}, \rho^n_{i,j,k-1}$, $\rho^n_{i,j,k+1}$ respectively on the right hand side of Eq. (9.2) by $\rho^n_{i,j,k}$. Then, there will be no density flux orthogonal to the six boundary faces (Figure 9.2**a**).

As a consequence, for each boundary node, one of its coordinates will remain unchanged under the update by the advection equation (9.7), while the other coordinates can vary. In other words, the boundary nodes can freely slide along the six boundary faces to achieve density equalization while preserving the rectangular shape of D throughout the iterations.

Free boundary condition

Another possibility is to let the domain deform freely. To achieve this, the "sea" approach in GN[52] can be used. We put the entire solid domain D with the prescribed density ρ^0 inside a larger rectangular solid domain $\widetilde{D}$ and define the density $\widetilde{\rho}^0$ on $\widetilde{D}$ by

$$\widetilde{\rho}^0 = \begin{cases} \rho^0 & \text{on } D, \\ \text{mean}(\rho^0) & \text{on } \widetilde{D} \setminus D. \end{cases} \tag{9.23}$$

We can then impose the no-flux boundary condition on $\partial\widetilde{D}$ and compute the density-equalizing map on $\widetilde{D}$ (Figure 9.2**b**). As there is no boundary condition imposed on ∂D, D can deform freely under the deformation of $\widetilde{D}$ without any components of the reference map ξ on ∂D being fixed. As a remark, setting $\widetilde{\rho}^0 = \text{mean}(\rho^0)$ at the "sea" $\widetilde{D} \setminus D$ prevents D from expanding infinitely.

Mixed boundary condition

One may also combine different boundary conditions to achieve other deformations. For instance, to keep the top and the bottom boundary planes planar while allowing the other boundary planes to deform freely, one can place the $L \times M \times N$ domain in a larger $\tilde{L} \times \tilde{M} \times N$ domain, where $\tilde{L} > L$ and $\tilde{M} > M$. Then, one can use the above-mentioned "sea" approach to enforce the no-flux boundary condition on the z-boundary planes and the free boundary condition on the x- and y-boundary planes. This results in the desired effect.

9.2 Experimental results

For the implementation of the proposed iterative scheme, we use C++ with OpenMP parallelization (with grid size $h = 1$, maximum number of iterations $n_{\max} = 10000$, and error threshold $\varepsilon = 10^{-2}$). The sparse linear systems are solved using the conjugate gradient method (`ConjugateGradient`) in the C++ library Eigen, with the default preconditioner `DiagonalPreconditioner` used. The experiments are performed on a PC with an Intel i7-6700K quad-core processor and 16 GB RAM. The statistics and the visualization are done using MATLAB. In the following, we assess the performance of the proposed method.

Consider a cubic solid D of grid size $L \times M \times N = 32 \times 32 \times 32$ with a smooth input density

$$\rho^0(i,j,k) = 10 + 9.99 \sin\left(\frac{4\pi i}{L-1}\right) \cos\left(\frac{2\pi j}{M-1}\right) \cos\left(\frac{2\pi k}{N-1}\right), \tag{9.24}$$

with $i,j,k = 0,1,\ldots,31$, as shown in Figure 9.3**a**. We compute the volumetric density-equalizing maps with the no-flux boundary condition (Figure 9.3**b**), the free boundary condition (Figure 9.3**c**), and the mixed-boundary condition (Figure 9.3**d**). For the experiment with the free boundary condition imposed, we place the solid in a circumscribed $48 \times 48 \times 48$ cubic grid. For the experiment with the mixed boundary condition imposed, we also make use of a circumscribed $48 \times 48 \times 32$ rectangular grid. It can be observed from Figure 9.3**b** that different regions are enlarged or shrunk according to the prescribed density. Also, the boundary shape of the deformed domain can be effectively controlled by imposing different boundary conditions. For instance, for

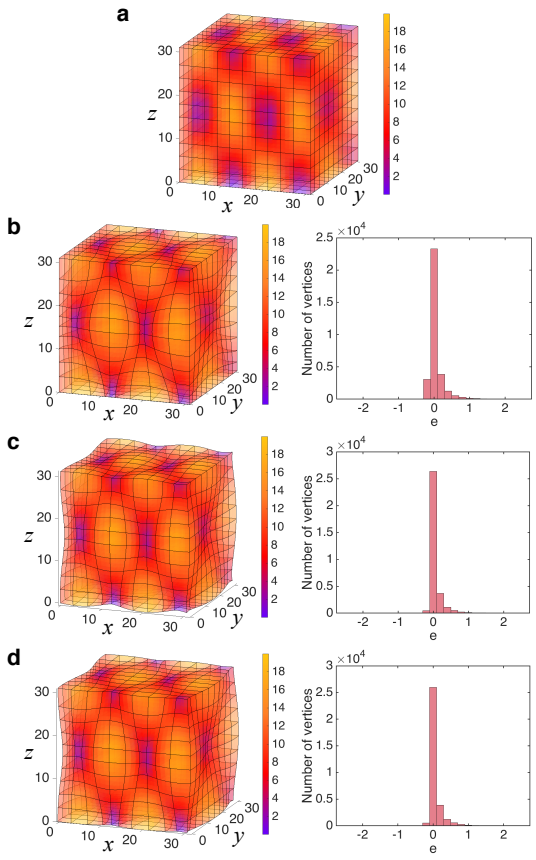

Figure 9.3: An example of volumetric density-equalizing maps on a $32 \times 32 \times 32$ grid, with the prescribed density being a periodic function with multiple peaks. The mapping results are visualized by the deformed contour planes in the x-, y-, and z-directions. **a**, The initial state. **b**, The forward mapping with the no-flux boundary condition and the histogram of the volume-density mismatch error e. **c**, The forward mapping with the free boundary condition and the histogram of e. **d**, The forward mapping with the mixed boundary condition where only the top and bottom boundary planes are enforced to be planar, and the histogram of e. All of them are color-coded with the prescribed density. For better visualization, only some of the contour planes are drawn.

the map with the free boundary condition (Figure 9.3**c**), the domain deforms freely while not maintaining a cubic shape. For the map with the mixed boundary condition (Figure 9.3**d**), the top and the bottom of the solid domain remain to be planar, while the other sides of the boundary are deformed freely.

We assess the accuracy of the proposed method by defining the *volume-density mismatch error* as

$$e(\mathbf{x}) = \log \frac{\det(F(\mathbf{x}))/\iiint_D \det(F(\mathbf{x}))}{\rho^0(\xi_{\text{final}}(\mathbf{x}))/\iiint_D \rho^0}, \tag{9.25}$$

where $F(\mathbf{x}) = \left(\frac{\partial \xi_{\text{final}}}{\partial \mathbf{x}}\right)^{-1}$ is the Jacobian. The integrals $\iiint_D \det(F(\mathbf{x}))$ and $\iiint_D \rho^0$ are computed using the trapezoidal method. In other words, e is the logged ratio of the volumetric scale factor to the prescribed density with a normalization, and it is equal to 0 if and only if the final volume distribution matches the prescribed density distribution perfectly. It can be observed from Figure 9.3 that the histograms of e highly concentrate at 0 regardless of the choice of the boundary conditions. This shows that our method is accurate.

Figure 9.4**a** shows another example of a 3D grid of size $L \times M \times N = 32 \times 32 \times 32$, with the prescribed density being discontinuous:

$$\rho^0(i,j,k) = \begin{cases} 1 & \text{if } i < L/2, j < M/2, k < N/2, \\ 3 & \text{if } i \geq L/2, j < M/2, k < N/2, \\ 5 & \text{if } i < L/2, j \geq M/2, k < N/2, \\ 7 & \text{if } i \geq L/2, j \geq M/2, k < N/2, \\ 9 & \text{if } i < L/2, j < M/2, k \geq N/2, \\ 11 & \text{if } i \geq L/2, j < M/2, k \geq N/2, \\ 13 & \text{if } i < L/2, j \geq M/2, k \geq N/2, \\ 15 & \text{if } i \geq L/2, j \geq M/2, k \geq N/2. \end{cases} \tag{9.26}$$

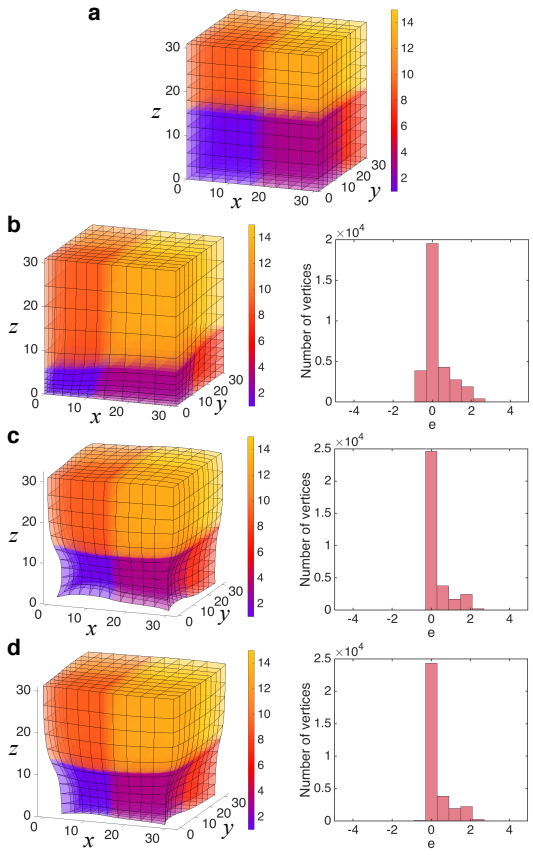

Figure 9.4: Another example of volumetric density-equalizing maps on a $32 \times 32 \times 32$ grid, with eight different density values defined on eight regions. See Figure 9.3 for the description of **a**-**d**.

As shown in Figure 9.4, our VDERM method effectively deforms different regions according to the input density, with different boundary conditions satisfied. While the input density is discontinuous, the histograms of e show that the error is still concentrated at 0. This shows that our method is capable of handling a large wide of input densities.

To further examine the performance of VDERM, we consider different resolutions $L \times L \times L$ with the density given by Eq. (9.24). An additional diffusion coefficient $\kappa = L/64$ is introduced on the right-hand side of Eq. (9.1) and Eq. (9.4) to rescale the rate of diffusion proportionally, thereby

No. of vertices ($L \times L \times L$)	**Time (s)**	**# iterations**	**mean($\lvert e \rvert$)**
$8 \times 8 \times 8 = 512$	0.02	105	0.8868
$16 \times 16 \times 16 = 4096$	0.20	323	0.4468
$24 \times 24 \times 24 = 13824$	1.14	504	0.2774
$32 \times 32 \times 32 = 32768$	4.43	637	0.1972
$40 \times 40 \times 40 = 64000$	12.13	727	0.1577
$48 \times 48 \times 48 = 110592$	26.65	784	0.1356
$56 \times 56 \times 56 = 175616$	59.59	816	0.1262
$64 \times 64 \times 64 = 262144$	111.53	856	0.1241

Table 9.1: The performance of the proposed method. The time required for the iterative scheme (with four OpenMP threads used), the number of iterations needed, and the average of the absolute volume-density mismatch error mean($\lvert e \rvert$) are recorded.

ensuring the fairness of the comparison. It can be observed from Table 9.1 that mean($\lvert e \rvert$) decreases linearly with L, which suggests that the accuracy of the density-equalizing reference map increases with the resolution.

9.3 Applications

9.3.1 Volumetric data visualization

The 2D density-equalizing maps have been widely applied to sociological and biological data visualization on planar maps. Analogously, we can apply our VDERM method to data visualization in 3D, as illustrated by two following medical and sociological examples.

Medical data visualization

In neurology, the *cortical homunculus* (also known as the *cortex man*) is a distorted 3D human model with different parts of the body enlarged or shunk for representing the proportion of the

human brain used for processing sensory or motor functions there[91] (Figure 9.5). Our VDERM method is naturally applicable for generating the cortical homunculus. To visualize the spatial acuity for pain in human body, we consider deforming a 3D human body model with the prescribed density being the reciprocal of the 2-point discrimination (2PD) threshold of each part of the body[90] divided by the volume of the part:

$$\rho^0 = \begin{cases} (1/0.6)/\text{Volume of the fingers} & \text{for the fingers,} \\ (1/1.1)/\text{Volume of the hand palms} & \text{for the palms,} \\ (1/2.7)/\text{Volume of the dorsum of the hands} & \text{for the dorsum of the hands,} \\ (1/2.1)/\text{Volume of the arms} & \text{for the arms,} \\ (1/1.2)/\text{Volume of the head} & \text{for the head,} \\ (1/1.4)/\text{Volume of the shoulders} & \text{for the shoulders,} \\ (1/2.5)/\text{Volume of the lower back} & \text{for the lower back,} \\ (1/2.9)/\text{Volume of the thighs} & \text{for the thighs,} \\ (1/3.3)/\text{Volume of the calves} & \text{for the calves,} \\ (1/3.5)/\text{Volume of the dorsum of the foot} & \text{for the dorsum of the foot,} \\ (1/1.3)/\text{Volume of the foot soles} & \text{for the foot soles.} \end{cases} \tag{9.27}$$

For the remaining parts of the human body, we define the density using the closest parts with data available. The region surrounding the human body is set as the "sea" for achieving a free-boundary deformation.

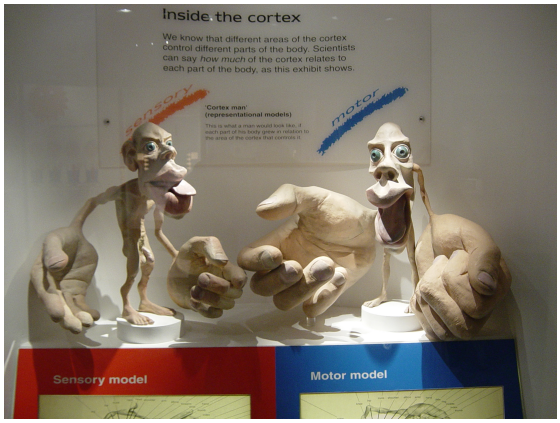

Figure 9.5: Sensory and motor homunculus sculptures at the Museum of Natural History, London, UK. The image is adopted online* under the CC BY-SA 3.0 license.

Figure 9.6 shows the deformed model proposed by VDERM, from which it can be observed that the parts with the highest spatial acuity (the hands) are significantly enlarged. The deformed model is highly similar to the cortical homunculus sculptures in Figure 9.5, except for the regions without available acuity data (such as the mouth/lips).

Sociological data visualization

It is also possible to visualize sociological data using VDERM. It is well known that the ticket price of different flight classes can be significantly different due to the different levels of service and accommodation. Here, we apply DERM for visualizing the ticket price of a round-trip direct flight between New York (JFK) and Hong Kong (HKG) (departure date: March 1, 2020; returning date: March 8, 2020; data retrieved on December 3, 2019 from the American Airlines website†). Both the departure flight and the returning flight are operated on a Boeing 777-300ER aircraft, which

*https://en.wikipedia.org/wiki/Cortical_homunculus
†https://www.aa.com

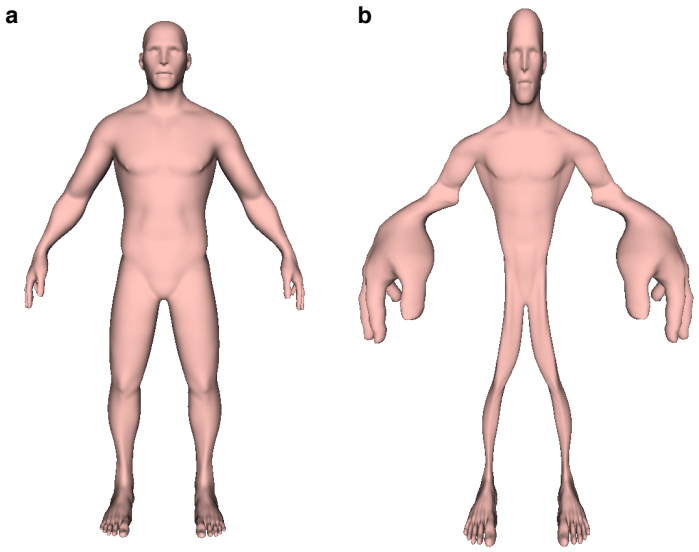

Figure 9.6: A cortical homunculus produced by our VDERM method. a, The input 3D human body model. **b**, The deformed model highlighting the spatial acuity of pain in human body [90].

typically consists of 6 first class seats, 53 business class seats, 34 premium economy class seats and 182 economy class seats[‡]. The ticket prices are $16039, $6922, $2542, and $941 for a first, business, premium economy, and economy class seat respectively.

We first apply VDERM to deform a 3D model of the Boeing 777-300ER aircraft (Figure 9.7**a**) with the density being the ticket price divided by the volume of each cabin:

$$\rho^0 = \begin{cases} 16039/\text{Volume of the first class cabin} & \text{for the first class cabin,} \\ 6922/\text{Volume of the business class cabin} & \text{for the business class cabin,} \\ 2542/\text{Volume of the premium economy class cabin} & \text{for the premium economy class cabin,} \\ 941/\text{Volume of the economy class cabin} & \text{for the economy class cabin.} \end{cases} \tag{9.28}$$

[‡] https://www.seatguru.com

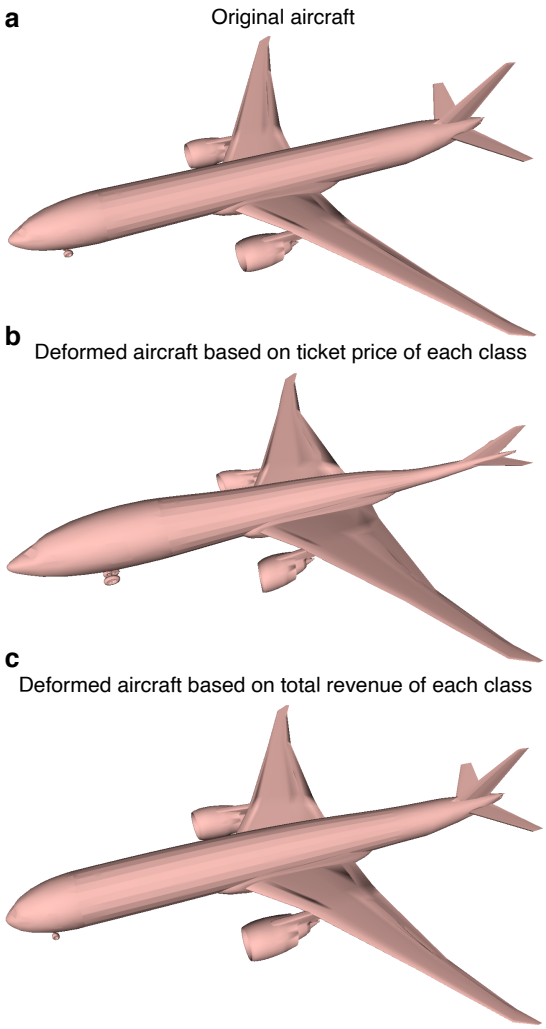

Figure 9.7: Visualizing the economics of airline class using VDERM. a, A 3D model of the Boeing 777-300ER aircraft. **b**, The deformed model produced by VDERM based on the ticket price for different travel classes. **c**, The deformed aircraft produced by VDERM based on the total revenue for each travel class.

Again, we set the surrounding region as the "sea" to allow for a free-boundary deformation. Here we remark that the dividing the ticket price by the volume of the cabin ensures that the ticket price ratio equals the ratio of the volume of the entire cabins. The resulting deformed model (Figure 9.7**b**) shows a significant enlargement at the front of the aircraft and a shrinkage at the end of it, reflecting

the large ticket price difference between the seats at the first class and the economy class.

We then estimate the total revenue of each ticket class by multiplying the ticket price by the number of seats, which gives \$96234, \$366866, \$86428, and \$171262 for the four classes respectively. We apply VDERM with the following density:

$$\rho^0 = \begin{cases} 96234/\text{Volume of the first class cabin} & \text{for the first class cabin,} \\ 366866/\text{Volume of the business class cabin} & \text{for the business class cabin,} \\ 86428/\text{Volume of the premium economy class cabin} & \text{for the premium economy class cabin,} \\ 171262/\text{Volume of the economy class cabin} & \text{for the economy class cabin.} \end{cases} \tag{9.29}$$

From the resulting deformed aircraft shown in Figure 9.7c, it can be observed that the economy class cabin is slightly shrunk, while the business class cabin is slightly expanded. Overall, the deformed aircraft is not significantly different from the input model, which suggests that the total revenue of each travel class is in fact similar.

9.3.2 Deformation-based shape modeling

The proposed VDERM method can also be utilized for deformation-based shape modeling. More specifically, to deform a mesh in $\mathbb{R}^3$, we can prescribe a density function on an underlying 3D grid and compute the volumetric density-equalizing map of it. The deformation of the grid then induces a deformation of the mesh.

We first consider deforming a dragon model adapted from The Stanford 3D Scanning

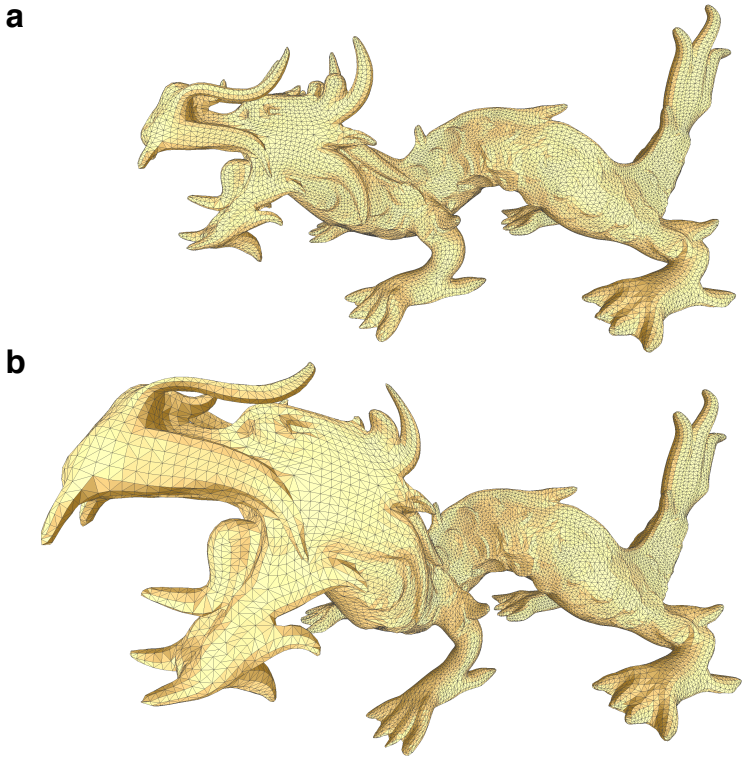

Figure 9.8: Deforming a dragon model using VDERM. a, The input dragon model. **b**, The deformed dragon with its head enlarged.

Repository[§] (Figure 9.8) to enlarge its head while maintaining the shape of its body. We use an underlying 3D grid of size $32 \times 32 \times 32$ and the following density:

$$\rho^0 = \begin{cases} 10 & \text{around the head of the dragon,} \\ 1 & \text{elsewhere.} \end{cases} \tag{9.30}$$

It can be observed that the head of the dragon is magnified under the deformation, and the body shape is basically unchanged.

[§] http://graphics.stanford.edu/data/3Dscanrep/

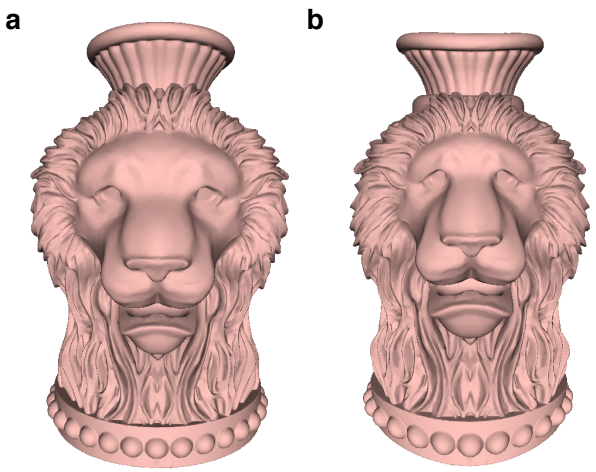

Figure 9.9: Deforming a lion vase model using VDERM. a, The input lion vase. **b**, The deformed lion vase with the facial expression changed.

We now consider another experiment on deforming a lion vase model adapted from the 3D Segmentation Benchmark[¶] (Figure 9.9). This time, we change the facial expression of the lion by setting a smaller density value around its forehead and a larger density value around its chin. It can be observed that facial expression of the lion is effectively changed under the deformation. The two examples demonstrate the effectiveness of VDERM for deformation-based shape modeling.

9.3.3 Shape morphing

As our proposed method deforms the underlying grid continuously using an iterative scheme, we can utilize the intermediate states of the deformation for producing a continuous change of the 3D model. As shown in Figure 9.10, our method is capable of changing changing the facial expression of the lion vase using a diffusion-based deformation. This shows that our method is advantageous

[¶] http://193.48.251.101/3dsegbenchmark/bust.html

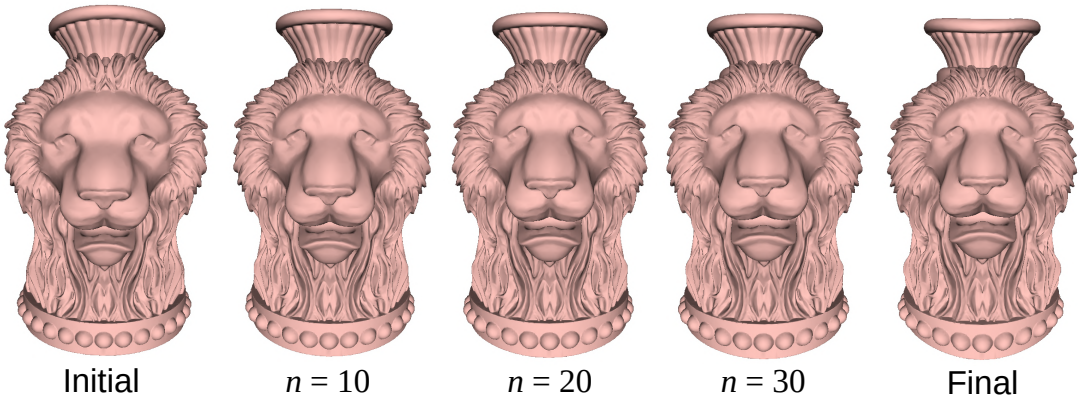

Figure 9.10: A continuous change in the facial expression produced using DERM.

for object morphing.

9.3.4 Adaptive remeshing

Remeshing aims at improving the discretization of meshes. Suppose we would like to construct a tetrahedral mesh for a genus-0 closed surface, with the mesh density of certain prescribed regions desired to be higher. By applying VDERM on the input domain D with a larger density ρ^0 at those regions, we obtain a volumetric density-equalizing reference map ξ_{final}. We can then construct a uniform tetrahedral mesh in the deformed domain using prior methods such as DistMesh [105], and map the uniform mesh back to D using ξ_{final}. This results in more tetrahedral elements being mapped to the regions with a larger prescribed density ρ^0, and hence we obtain a tetrahedral mesh with the desired mesh density.

Figure 9.11**a** shows an example with the central part of the solid desired to be with a higher mesh density. The remeshed solid is achieved by applying our VDERM method with the following

density:

$$\rho^0 = \begin{cases} 10 & \text{around the center of the domain,} \\ 1 & \text{elsewhere.} \end{cases} \tag{9.31}$$

Figure 9.11**b** shows another example, with the input density given by Eq. (9.26). Our approach produces eight different tetrahedral mesh densities at the eight corners. Figure 9.11**c** shows a more complicated example, with the mesh density inside a π shape being much higher than the other regions of the domain. This is achieved by applying VDERM with ρ^0 being 10 inside the π shape and 1 otherwise. The three examples demonstrate the effectiveness of our method for adaptive remeshing.

We remark that DistMesh [105] also allows for adaptive remeshing, by specifying an edge length function expression that yields the target adaptive resolution. However, to achieve complicated effects such as the example shown in Figure 9.11**c**, it may be very difficult to specify an appropriate edge length function expression. By contrast, our approach combining both DistMesh and VDERM allows us to precisely control element volume by exactly specifying the volumes of mesh elements, so that such effects can be easily achieved. This shows the advantage of our approach for generating meshes with added precision in certain regions for numerical computations.

9.4 Discussion

Our proposed VDERM method is the first work on the density-equalizing maps in 3D for volumetric deformations. We have demonstrated the effectiveness of the method for volumetric data visualization, deformation-based shape modeling and remeshing. The intermediate process can also

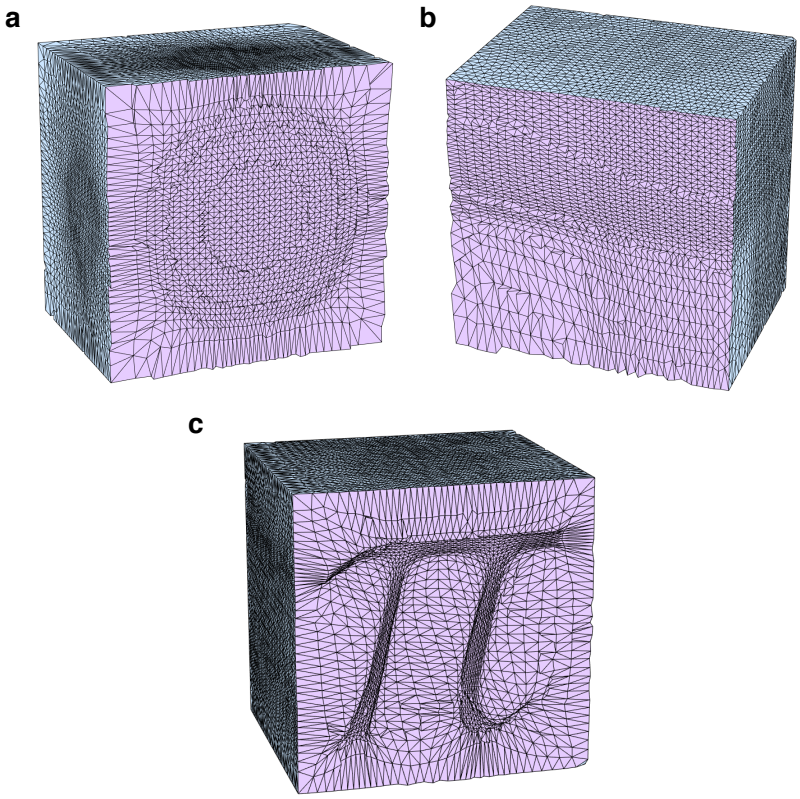

Figure 9.11: Remeshing results produced by our VDERM method. Each example is displayed in a cross-sectional view, with the surface colored in blue and the interior tetrahedral mesh elements colored in purple. **a**, A cube with a higher tetrahedral mesh density at the center. **b**, A cube with eight different tetrahedral mesh densities at eight regions. **c**, A cube with a higher tetrahedral mesh density inside a π shape.

be utilized for object morphing. Overall, our novel 3D generalization opens up a wide range of new applications of the density-equalizing maps.

10

Conclusion

In this thesis, we have developed mathematical approaches for advancing our understanding of mechanical metamaterials, biological shapes, and computational mappings using discrete and continuous geometry.

By proposing novel frameworks for the design of kirigami structures with prescribed geometric, topological and physical properties, we have provided a class of new ways to connect shape and

function of mechanical metamaterials. A natural next step is to explore how we can couple our kirigami design frameworks with their counterparts in origami [40,17] to obtain more flexible deployable structures. We have taken the first step of connecting the two design problems locally at the growth front of origami and kirigami structures in our additive design approaches, and it will be interesting to see how our methods can complement other recent approaches in the design and fabrication of physical kirigami [108,111,7,134,150] and origami [146,116,47,143,121,115] structures. Also, with the increasing interest in 3D metamaterial design [102,65,70], another natural future direction is to extend our design frameworks for creating deployable structures using 3D solids instead of flat, thin sheets. While we have extended our hierarchical construction method for the topological control of 3D structural assemblies, the extension of the geometric design approaches has yet to be established.

We have also studied growth and form using advanced computational tools involving geometry and mechanics. These tools allow us to analyze a wide range of complex biological shapes in nature, with varying size, dimension, and curvature. While our quasi-conformal mapping method enables us to assess the change in the local size, eccentricity and orientation in a mapping between two shapes, it is limited by the boundary correspondence prescribed. A natural next step is to explore how we can remove the requirement and produce quasi-conformal flows that simulate and explain the growth of shapes, possibly by solving some time-dependent PDEs related to the Beltrami equation such as the Beltrami holomorphic flow [87,88] or some variants of the Ricci flow [67,148]. Another possible future direction is to combine our geometric tools with topological methods and assess the interplay between gene and shape.

Using the physical principle of diffusion, we have developed several mapping algorithms for two-

and three-dimensional objects with applications to computer graphics, geometry processing, as well as sociological and medical data visualization. While our methods effectively produce deformations related to the area or volume of the mesh elements, they do not allow for extra effects such as rotation, shear or landmark-matching. A possible future direction is to extend our methods by incorporating other deformation energies[63] or landmark-matching approaches[69,86,81], which would allow us to create a wider range of shape deformations.

More broadly, the three major fields covered in this thesis are closely related: Nature inspires engineering; engineering realizes computation; and computation quantifies nature. Our works fuse ideas and tools from all these fields, yielding important insight into interdisciplinary mathematical sciences.

References

[1] Alim, K., Armon, S., Shraiman, B. I., & Boudaoud, A. (2016). Leaf growth is conformal. *Physical Biology*, 13(5), 05LT01.

[2] Angenent, S., Haker, S., Tannenbaum, A., & Kikinis, R. (1999). On the Laplace–Beltrami operator and brain surface flattening. *IEEE Transactions on Medical Imaging*, 18, 700–711.

[3] Antiga, L. & Steinman, D. A. (2004). Robust and objective decomposition and mapping of bifurcating vessels. *IEEE Transactions on Medical Imaging*, 23, 704–713.

[4] Barnette, A. R., Neil, J. J., Kroenke, C. D., Griffith, J. L., Epstein, A. A., Bayly, P. V., Knutsen, A. K., & Inder, T. E. (2009). Characterization of brain development in the ferret via mri. *Pediatric Research*, 66(1), 80–84.

[5] Beg, M. F., Miller, M. I., Trouvé, A., & Younes, L. (2005). Computing large deformation metric mappings via geodesic flows of diffeomorphisms. *International Journal of Computer Vision*, 61(2), 139–157.

[6] Benjamin, E. J. (2017). Heart disease and stroke statistics–2017 update: a report from the american heart association. *Circulation*, 135, e146–e603.

[7] Bertoldi, K., Vitelli, V., Christensen, J., & van Hecke, M. (2017). Flexible mechanical metamaterials. *Nature Reviews Materials*, 2(11), 1–11.

[8] Blees, M. K., Barnard, A. W., Rose, P. A., Roberts, S. P., McGill, K. L., Huang, P. Y., Ruyack, A. R., Kevek, J. W., Kobrin, B., Muller, D. A., et al. (2015). Graphene kirigami. *Nature*, 524(7564), 204.

[9] Bonacum, J., O'Grady, P. M., Kambysellis, M., & DeSalle, R. (2005). Phylogeny and age of diversification of the planitibia species group of the hawaiian drosophila. *Molecular Phylogenetics and Evolution*, 37(1), 73–82.

[10] Bonet, J. & Burton, A. J. (1998). A simple average nodal pressure tetrahedral element for incompressible and nearly incompressible dynamic explicit applications. *Communications in Numerical Methods in Engineering*, 14(5), 437–449.

[11] Bookstein, F. L. (1989). Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(6), 567–585.

[12] Borkin, M., Gajos, K., Peters, A., Mitsouras, D., Melchionna, S., Rybicki, F., Feldman, C., & Pfister, H. (2011). Evaluation of artery visualizations for heart disease diagnosis. *IEEE Transactions on Visualization and Computer Graphics*, 17, 2479–2488.

[13] Bossavit, A. (1988). Whitney forms: A class of finite elements for three-dimensional computations in electromagnetism. *IEE Proceedings A (Physical Science, Measurement and Instrumentation, Management and Education, Reviews)*, 135(8), 493–500.

[14] Browne, J. E., Watson, A. J., Gibson, N. M., Dudley, N. J., & Elliott, A. T. (2004). Objective measurements of image quality. *Ultrasound in Medicine & Biology*, 30, 229–237.

[15] Celli, P., McMahan, C., Ramirez, B., Bauhofer, A., Naify, C., Hofmann, D., Audoly, B., & Daraio, C. (2018). Shape-morphing architected sheets with non-periodic cut patterns. *Soft Matter*, 14(48), 9744–9749.

[16] Chen, B. G.-g., Liu, B., Evans, A. A., Paulose, J., Cohen, I., Vitelli, V., & Santangelo, C. D. (2016). Topological mechanics of origami and kirigami. *Physical Review Letters*, 116(13), 135501.

[17] Chen, S. & Mahadevan, L. (2019). Rigidity percolation and geometric information in floppy origami. *Proceedings of the National Academy of Sciences*, 116(17), 8119–8124.

[18] Chen, Y. & Chiu, B. (2016). Correspondence optimization in 2D standardized carotid wall thickness map by description length minimization: A tool for increasing reproducibility of 3D ultrasound-based measurements. *Medical Physics*, 43, 6474–6490.

[19] Cheng, J., Ukwatta, E., Shavakh, S., Chow, T. W. S., Parraga, G., Spence, J. D., & Chiu, B. (2017). Sensitive three-dimensional ultrasound assessment of carotid atherosclerosis by

weighted average of local vessel wall and plaque thickness change. *Medical Physics*, 44, 5280–5292.

[20] Chiu, B., Chen, W., & Cheng, J. (2016). Concise biomarker for spatial-temporal change in three-dimensional ultrasound measurement of carotid vessel wall and plaque thickness based on a graph-based random walk framework: towards sensitive evaluation of response to therapy. *Computers in Biology and Medicine*, 79, 149–162.

[21] Chiu, B., Egger, M., Spence, J. D., Parraga, G., & Fenster, A. (2008a). Area-preserving flattening maps of 3D ultrasound carotid arteries images. *Medical Image Analysis*, 12, 676–688.

[22] Chiu, B., Egger, M., Spence, J. D., Parraga, G., & Fenster, A. (2008b). Quantification of carotid vessel wall and plaque thickness change using 3D ultrasound images. *Medical Physics*, 35, 3691–3710.

[23] Chiu, B., Li, B., & Chow, T. W. S. (2013a). Novel 3D ultrasound image-based biomarkers based on a feature selection from a 2D standardized vessel wall thickness map: a tool for sensitive assessment of therapies for carotid atherosclerosis. *Physics in Medicine & Biology*, 58, 5959–5982.

[24] Chiu, B., Ukwatta, E., Shavakh, S., & Fenster, A. (2013b). Quantification and visualization of carotid segmentation accuracy and precision using a 2D standardized carotid map. *Physics in Medicine & Biology*, 58, 3671–3703.

[25] Choi, G. P. T., Chen, Y., Lui, L. M., & Chiu, B. (2017). Conformal mapping of carotid vessel wall and plaque thickness measured from 3D ultrasound images. *Medical & Biological Engineering & Computing*, 55, 2183–2195.

[26] Choi, G. P.-T. & Lui, L. M. (2018). A linear formulation for disk conformal parameterization of simply-connected open surfaces. *Advances in Computational Mathematics*, 44(1), 87–114.

[27] Choi, P. T., Lam, K. C., & Lui, L. M. (2015). FLASH: Fast landmark aligned spherical harmonic parameterization for genus-0 closed brain surfaces. *SIAM Journal on Imaging Sciences*, 8(1), 67–94.

[28] Choi, P. T. & Lui, L. M. (2015). Fast disk conformal parameterization of simply-connected open surfaces. *Journal of Scientific Computing*, 65(3), 1065–1090.

[29] Christensen, G. E., Rabbitt, R. D., & Miller, M. I. (1996). Deformable templates using large deformation kinematics. *IEEE Transactions on Image Processing*, 5(10), 1435–1447.

[30] Colizza, V., Barrat, A., Barthélemy, M., & Vespignani, A. (2006). The role of the airline transportation network in the prediction and predictability of global epidemics. *Proceedings of the National Academy of Sciences*, 103(7), 2015–2020.

[31] Courant, R., Friedrichs, K., & Lewy, H. (1928). Über die partiellen differenzengleichungen der mathematischen physik. *Mathematische Annalen*, 100(1), 32–74.

[32] Davies, R. H., Twining, C. J., Cootes, T. F., Waterton, J. C., & Taylor, C. J. (2002). 3D statistical shape models using direct optimisation of description length. *European Conference on Computer Vision*, (pp. 3–20).

[33] de Goes, F., Desbrun, M., & Tong, Y. (2016). Vector field processing on triangle meshes. *ACM SIGGRAPH 2016 Courses*, (pp. 1–49).

[34] Dehn, M. (1916). Über die starrheit konvexer polyeder. *Mathematische Annalen*, 77(4), 466–473.

[35] Desbrun, M., Meyer, M., & Alliez, P. (2002). Intrinsic parameterizations of surface meshes. *Computer Graphics Forum*, 21(3), 209–218.

[36] Dieleman, P., Vasmel, N., Waitukaitis, S., & van Hecke, M. (2020). Jigsaw puzzle design of pluripotent origami. *Nature Physics*, 16, 63–68.

[37] do Carmo, M. P. (1976). *Differential geometry of curves and surfaces*. Prentice-Hall.

[38] Dorling, D. (1996). *Area cartograms: Their use and creation, concepts and techniques in modern geography*. Institute of British Geographers.

[39] Dorling, D., Newman, M., & Barford, A. (2010). *The atlas of the real world: mapping the way we live*. Thames & Hudson.

[40] Dudte, L. H., Vouga, E., Tachi, T., & Mahadevan, L. (2016). Programming curvature using origami tessellations. *Nature Materials*, 15(5), 583.

[41] Edelsbrunner, H. & Waupotitsch, R. (1997). A combinatorial approach to cartograms. *Computational Geometry*, 7(5-6), 343–360.

[42] Edwards, K. A., Doescher, L. T., Kaneshiro, K. Y., & Yamamoto, D. (2007). A database of wing diversity in the hawaiian drosophila. *PLoS One*, 2(5).

[43] Egger, M., Chiu, B., Spence, J. D., Fenster, A., & Parraga, G. (2008). Mapping spatial and temporal changes in carotid atherosclerosis from three-dimensional ultrasound images. *Ultrasound in Medicine & Biology*, 34, 64–72.

[44] Eicke, B. M., Lorentz, J. V., & Paulus, W. (1995). Embolus detection in different degrees of carotid disease. *Neurological Tesearch*, 17, 181–184.

[45] Ericson, C. (2004). *Real-time collision detection*. CRC Press.

[46] Filipov, E. T., Paulino, G. H., & Tachi, T. (2016). Origami tubes with reconfigurable polygonal cross-sections. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 472(2185), 20150607.

[47] Filipov, E. T., Tachi, T., & Paulino, G. H. (2015). Origami tubes assembled into stiff, yet reconfigurable structures and metamaterials. *Proceedings of the National Academy of Sciences*, 112(40), 12321–12326.

[48] Floater, M. S. & Hormann, K. (2005). Surface parameterization: a tutorial and survey. In *Advances in Multiresolution for Heometric Modelling* (pp. 157–186).

[49] Fox, J. G. & Marini, R. P. (2014). *Biology and Diseases of the Ferret*. John Wiley & Sons.

[50] Fruchart, M., Zhou, Y., & Vitelli, V. (2020). Dualities and non-abelian mechanics. *Nature*, 577, 636–640.

[51] Gardiner, F. P. & Lakic, N. (2000). *Quasiconformal Teichmüller theory*. American Mathematical Soc.

[52] Gastner, M. T. & Newman, M. E. J. (2004). Diffusion-based method for producing density-equalizing maps. *Proceedings of the National Academy of Sciences*, 101, 7499–7504.

[53] Gatt, R., Mizzi, L., Azzopardi, J. I., Azzopardi, K. M., Attard, D., Casha, A., Briffa, J., & Grima, J. N. (2015). Hierarchical auxetic mechanical metamaterials. *Scientific Reports*, 5, 8395.

[54] Gleditsch, K. S. & Ward, M. D. (2006). Diffusion and the international context of democratization. *International Organization*, 60(4), 911–933.

[55] Gower, J. C. (1975). Generalized procrustes analysis. *Psychometrika*, 40(1), 33–51.

[56] Grima, J. N., Alderson, A., & Evans, K. E. (2004). Negative poisson's ratios from rotating rectangles. *Computational Methods in Science and Technology*, 10(2), 137–145.

[57] Grima, J. N., Alderson, A., & Evans, K. E. (2005). Auxetic behaviour from rotating rigid units. *Physica Status Solidi (b)*, 242(3), 561–575.

[58] Grima, J. N. & Evans, K. E. (2000). Auxetic behavior from rotating squares. *Journal of Materials Science Letters*.

[59] Grünbaum, B. & Shephard, G. C. (1987). *Tilings and patterns*. Freeman.

[60] Gu, X., Wang, Y., Chan, T. F., Thompson, P. M., & Yau, S.-T. (2004). Genus zero surface conformal mapping and its application to brain surface mapping. *IEEE Transactions on Medical Imaging*, 23(8), 949–958.

[61] Guest, S. (2006). The stiffness of prestressed frameworks: a unifying approach. *International Journal of Solids and Structures*, 43(3-4), 842–854.

[62] Haker, S., Angenent, S., Tannenbaum, A., & Kikinis, R. (2000). Nondistorting flattening maps and the 3-d visualization of colon ct images. *IEEE Transactions on Medical Imaging*, 19, 665–670.

[63] Hildebrandt, K., Schulz, C., Tycowicz, C. v., & Polthier, K. (2011). Interactive surface modeling using modal analysis. *ACM Transactions on Graphics*, 30(5), 1–11.

[64] Hormann, K., Lévy, B., & Sheffer, A. (2007). Mesh parameterization: Theory and practice. *Proceeding of ACM SIGGRAPH 2007 Courses*, 1, 1–122.

[65] Iniguez-Rabago, A., Li, Y., & Overvelde, J. T. B. (2019). Exploring multistability in prismatic metamaterials through local actuation. *Nature Communications*, 10(1), 1–10.

[66] Isobe, M. & Okumura, K. (2016). Initial rigid response and softening transition of highly stretchable kirigami sheet materials. *Scientific Reports*, 6, 24758.

[67] Jin, M., Kim, J., Luo, F., & Gu, X. (2008). Discrete surface Ricci flow. *IEEE Transactions on Visualization and Computer Graphics*, 14(5), 1030–1043.

[68] Jones, G. W. & Mahadevan, L. (2013). Planar morphometry, shear and optimal quasi-conformal mappings. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 469(2153), 20120653.

[69] Joshi, S. C. & Miller, M. I. (2000). Landmark matching via large deformation diffeomorphisms. *IEEE Transactions on Image Processing*, 9(8), 1357–1370.

[70] Kadic, M., Milton, G. W., van Hecke, M., & Wegener, M. (2019). 3d metamaterials. *Nature Reviews Physics*, 1(3), 198–210.

[71] Kamrin, K., Rycroft, C. H., & Nave, J.-C. (2012). Reference map technique for finite-strain elasticity and fluid–solid interaction. *Journal of the Mechanics and Physics of Solids*, 60, 1952–1969.

[72] Kane, C. L. & Lubensky, T. C. (2014). Topological boundary modes in isostatic lattices. *Nature Physics*, 10(1), 39–45.

[73] Keim, D. A., North, S. C., & Panse, C. (2004). Cartodraw: A fast algorithm for generating contiguous cartograms. *IEEE Transactions on Visualization and Computer Graphics*, 10(1), 95–110.

[74] Keim, D. A., Panse, C., & North, S. C. (2005). Medial-axis-based cartograms. *IEEE Computer Graphics and Applications*, 25(3), 60–68.

[75] Kolken, H. M. A. & Zadpoor, A. A. (2017). Auxetic mechanical metamaterials. *RSC Advances*, 7(9), 5111–5129.

[76] Konaković, M., Crane, K., Deng, B., Bouaziz, S., Piker, D., & Pauly, M. (2016). Beyond developable: computational design and fabrication with auxetic materials. *ACM Transactions on Graphics (TOG)*, 35(4), 89.

[77] Konaković-Luković, M., Panetta, J., Crane, K., & Pauly, M. (2018). Rapid deployment of curved surfaces via programmable auxetics. *ACM Transactions on Graphics*, 37(4), 106.

[78] Krasinski, A., Chiu, B., Spence, J. D., Fenster, A., & Parraga, G. (2009). Three-dimensional ultrasound quantification of intensive statin treatment of carotid atherosclerosis. *Ultrasound in Medicine & Biology*, 35, 1763–1772.

[79] Kurtek, S. A., Srivastava, A., & Wu, W. (2011). Signal estimation under random time-warpings and nonlinear signal alignment. *Advances in Neural Information Processing Systems*, (pp. 675–683).

[80] Landry, A., Ainsworth, C., Blake, C., Spence, J. D., & Fenster, A. (2007). Manual planimetric measurement of carotid plaque volume using three-dimensional ultrasound imaging. *Medical Physics*, 34, 1496–1505.

[81] Lee, Y. T., Lam, K. C., & Lui, L. M. (2016). Landmark-matching transformation with large deformation via n-dimensional quasi-conformal maps. *Journal of Scientific Computing*, 67(3), 926–954.

[82] Liu, L., Wang, D., Wong, K. L., & Wang, Y. (2011). Stroke and stroke care in china: huge burden, significant workload, and a national priority. *Stroke*, 42, 3651–3654.

[83] Lubbers, L. A. & van Hecke, M. (2019). Excess floppy modes and multibranched mechanisms in metamaterials with symmetries. *Physical Review E*, 100(2), 021001.

[84] Lubensky, T. C., Kane, C. L., Mao, X., Souslov, A., & Sun, K. (2015). Phonons and elasticity in critically coordinated lattices. *Reports on Progress in Physics*, 78(7), 073901.

[85] Lui, L. M., Gu, X., & Yau, S.-T. (2015). Convergence of an iterative algorithm for teichmüller maps via harmonic energy optimization. *Mathematics of Computation*, 84(296), 2823–2842.

[86] Lui, L. M., Lam, K. C., Yau, S.-T., & Gu, X. (2014). Teichmüller mapping (t-map) and its applications to landmark matching registration. *SIAM Journal on Imaging Sciences*, 7(1), 391–426.

[87] Lui, L. M., Wong, T. W., Thompson, P., Chan, T., Gu, X., & Yau, S.-T. (2010). Shape-based diffeomorphic registration on hippocampal surfaces using Beltrami holomorphic flow. *International Conference on Medical Image Computing and Computer-Assisted Intervention*, (pp. 323–330).

[88] Lui, L. M., Wong, T. W., Zeng, W., Gu, X., Thompson, P. M., Chan, T. F., & Yau, S.-T. (2012). Optimization of surface registrations using Beltrami holomorphic flow. *Journal of Scientific Computing*, 50(3), 557–585.

[89] Mancia, G., Fagard, R., Narkiewicz, K., Redon, J., Zanchetti, A., Boehm, M., Christiaens, T., Cifkova, R., De Backer, G., Dominiczak, A., et al. (2013). 2013 ESH/ESC guidelines for the management of arterial hypertension: the task force for the management of arterial hypertension of the European Society of Hypertension (ESH) and of the European Society of Cardiology (ESC). *Journal of Hypertension*, 31, 1281–1357.

[90] Mancini, F., Bauleo, A., Cole, J., Lui, F., Porro, C. A., Haggard, P., & Iannetti, G. D. (2014). Whole-body mapping of spatial acuity for pain and touch. *Annals of Neurology*, 75(6), 917–924.

[91] Marieb, E. N. & Hoehn, K. (2007). *Human anatomy & physiology*. Pearson education.

[92] Meng, T. W., Choi, G. P.-T., & Lui, L. M. (2016). Tempo: feature-endowed teichmuller extremal mappings of point clouds. *SIAM Journal on Imaging Sciences*, 9(4), 1922–1962.

[93] Mislove, A., Lehmann, S., Ahn, Y.-Y., Onnela, J.-P., & Rosenquist, J. N. (2011). Understanding the demographics of twitter users. *Fifth International AAAI Conference on Weblogs and Social Media*, (pp. 554–557).

[94] Mitchison, G. (2016). Conformal growth of arabidopsis leaves. *Journal of Theoretical Biology*, 408, 155–166.

[95] Mitschke, H., Robins, V., Mecke, K., & Schröder-Turk, G. E. (2013). Finite auxetic deformations of plane tessellations. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 469(2149), 20120465.

[96] Morrison, G. & Mahadevan, L. (2011). Asymmetric network connectivity using weighted harmonic averages. *EPL (Europhysics Letters)*, 93(4), 40002.

[97] Morrison, G. & Mahadevan, L. (2012). Discovering communities through friendship. *PLoS One*, 7(7).

[98] Moshe, M., Esposito, E., Shankar, S., Bircan, B., Cohen, I., Nelson, D. R., & Bowick, M. J. (2019). Kirigami mechanics as stress relief by elastic charges. *Physical Review Letters*, 122(4), 048001.

[99] Neal, J., Takahashi, M., Silva, M., Tiao, G., Walsh, C. A., & Sheen, V. L. (2007). Insights into the gyrification of developing ferret brain by magnetic resonance imaging. *Journal of Anatomy*, 210(1), 66–77.

[100] Neville, R. M., Scarpa, F., & Pirrera, A. (2016). Shape morphing kirigami mechanical metamaterials. *Scientific Reports*, 6, 31067.

[101] Nijhout, H. F., Cinderella, M., & Grunert, L. W. (2014). The development of wing shape in lepidoptera: mitotic density, not orientation, is the primary determinant of shape. *Evolution & Development*, 16(2), 68–77.

[102] Overvelde, J. T. B., Weaver, J. C., Hoberman, C., & Bertoldi, K. (2017). Rational design of reconfigurable prismatic architected materials. *Nature*, 541(7637), 347.

[103] Pan, R. K., Kaski, K., & Fortunato, S. (2012). World citation and collaboration networks: uncovering the role of geography in science. *Scientific Reports*, 2, 902.

[104] Parchem, R. J., Perry, M. W., & Patel, N. H. (2007). Patterns on the insect wing. *Current Opinion in Genetics & Development*, 17(4), 300–308.

[105] Persson, P.-O. & Strang, G. (2004). A simple mesh generator in MATLAB. *SIAM Review*, 46(2), 329–345.

[106] Prabhakaran, S., Rundek, T., Ramas, R., Elkind, M. S., Paik, M. C., Boden-Albala, B., & Sacco, R. L. (2006). Carotid plaque surface irregularity predicts ischemic stroke: the northern manhattan study. *Stroke*, 37, 2696–2701.

[107] Rabinovich, M., Poranne, R., Panozzo, D., & Sorkine-Hornung, O. (2017). Scalable locally injective mappings. *ACM Transactions on Graphics (TOG)*, 36(4), 1.

[108] Rafsanjani, A. & Bertoldi, K. (2017). Buckling-induced kirigami. *Physical Review Letters*, 118(8), 084301.

[109] Rafsanjani, A., Jin, L., Deng, B., & Bertoldi, K. (2019). Propagation of pop ups in kirigami shells. *Proceedings of the National Academy of Sciences*, 116(17), 8200–8205.

[110] Rafsanjani, A. & Pasini, D. (2016). Bistable auxetic mechanical metamaterials inspired by ancient geometric motifs. *Extreme Mechanics Letters*, 9, 291–296.

[111] Rafsanjani, A., Zhang, Y., Liu, B., Rubinstein, S. M., & Bertoldi, K. (2018). Kirigami skins make a simple soft actuator crawl. *Science Robotics*, 3(15), eaar7555.

[112] Reuter, M., Biasotti, S., Giorgi, D., Patanè, G., & Spagnuolo, M. (2009). Discrete Laplace–Beltrami operators for shape analysis and segmentation. *Computers & Graphics*, 33(3), 381–390.

[113] Rolland-Lagan, A.-G., Remmler, L., & Girard-Bock, C. (2014). Quantifying shape changes and tissue deformation in leaf development. *Plant Physiology*, 165(2), 496–505.

[114] Rycroft, C. H., Wu, C.-H., Yu, Y., & Kamrin, K. (2018). Reference map technique for incompressible fluid-structure interaction. *arXiv preprint, arXiv:1810.03015*.

[115] Santangelo, C. D. (2017). Extreme mechanics: self-folding origami. *Annual Review of Condensed Matter Physics*, 8, 165–183.

[116] Schenk, M. & Guest, S. D. (2013). Geometry of miura-folded metamaterials. *Proceedings of the National Academy of Sciences*, 110(9), 3276–3281.

[117] Shan, S., Kang, S. H., Zhao, Z., Fang, L., & Bertoldi, K. (2015). Design of planar isotropic negative poisson's ratio structures. *Extreme Mechanics Letters*, 4, 96–102.

[118] Sheffer, A., Praun, E., Rose, K., et al. (2007). Mesh parameterization methods and their applications. *Foundations and Trends in Computer Graphics and Vision*, 2(2), 105–171.

[119] Shyu, T. C., Damasceno, P. F., Dodd, P. M., Lamoureux, A., Xu, L., Shlian, M., Shtein, M., Glotzer, S. C., & Kotov, N. A. (2015). A kirigami approach to engineering elasticity in nanocomposites through patterned defects. *Nature Materials*, 14(8), 785.

[120] Silverberg, J. L., Evans, A. A., McLeod, L., Hayward, R. C., Hull, T., Santangelo, C. D., & Cohen, I. (2014). Using origami design principles to fold reprogrammable mechanical metamaterials. *Science*, 345(6197), 647–650.

[121] Silverberg, J. L., Na, J.-H., Evans, A. A., Liu, B., Hull, T. C., Santangelo, C. D., Lang, R. J., Hayward, R. C., & Cohen, I. (2015). Origami structures with a critical transition to bistability arising from hidden degrees of freedom. *Nature Materials*, 14(4), 389–393.

[122] Smart, I. H. & McSherry, G. M. (1986a). Gyrus formation in the cerebral cortex in the ferret. i. description of the external changes. *Journal of Anatomy*, 146, 141.

[123] Smart, I. H. & McSherry, G. M. (1986b). Gyrus formation in the cerebral cortex of the ferret. ii. description of the internal histological changes. *Journal of Anatomy*, 147, 27.

[124] Smith, J. & Schaefer, S. (2015). Bijective parameterization with free boundaries. *ACM Transactions on Graphics (TOG)*, 34(4), 1–9.

[125] Smith, R. S., Kenny, C. J., Ganesh, V., Jang, A., Borges-Monroy, R., Partlow, J. N., Hill, R. S., Shin, T., Chen, A. Y., Doan, R. N., et al. (2018). Sodium channel scn3a (nav1. 3) regulation of human cerebral cortical folding and oral motor development. *Neuron*, 99(5), 905–913.

[126] Spence, J. D., Eliasziw, M., DiCicco, M., Hackam, D. G., Galil, R., & Lohmann, T. (2002). Carotid plaque area: a tool for targeting and evaluating vascular preventive therapy. *Stroke*, 33, 2916–2922.

[127] Srivastava, A. & Klassen, E. P. (2016). *Functional and shape data analysis*. Heidelberg: Springer.

[128] Srivastava, A., Wu, W., Kurtek, S., Klassen, E., & Marron, J. S. (2011). Registration of functional data using fisher-rao metric. *arXiv preprint, arXiv:1103.3817*.

[129] Stavric, M. & Wiltsche, A. (2019). Geometrical elaboration of auxetic structures. *Nexus Network Journal*, 21(1), 79–90.

[130] Stomakhin, A., Howes, R., Schroeder, C., & Teran, J. M. (2012). Energetically consistent invertible elasticity. *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, (pp. 25–32).

[131] Strebel, K. (1978). On quasiconformal mappings of open riemann surfaces. *Commentarii Mathematici Helvetici*, 53(1), 301–321.

[132] Su, K., Cui, L., Qian, K., Lei, N., Zhang, J., Zhang, M., & Gu, X. D. (2016). Area-preserving mesh parameterization for poly-annulus surfaces based on optimal mass transportation. *Computer Aided Geometric Design*, 46, 76–91.

[133] Sun, K., Souslov, A., Mao, X., & Lubensky, T. C. (2012). Surface phonons, elastic response, and conformal invariance in twisted kagome lattices. *Proceedings of the National Academy of Sciences*, 109(31), 12369–12374.

[134] Sussman, D. M., Cho, Y., Castle, T., Gong, X., Jung, E., Yang, S., & Kamien, R. D. (2015). Algorithmic lattice kirigami: A route to pluripotent materials. *Proceedings of the National Academy of Sciences*, 112(24), 7449–7453.

[135] Tallinen, T., Chung, J. Y., Biggins, J. S., & Mahadevan, L. (2014). Gyrification from constrained cortical expansion. *Proceedings of the National Academy of Sciences*, 111(35), 12667–12672.

[136] Tallinen, T., Chung, J. Y., Rousseau, F., Girard, N., Lefèvre, J., & Mahadevan, L. (2016). On the growth and form of cortical convolutions. *Nature Physics*, 12(6), 588.

[137] Tang, Y., Lin, G., Yang, S., Yi, Y. K., Kamien, R. D., & Yin, J. (2017). Programmable kiri-kirigami metamaterials. *Advanced Materials*, 29(10), 1604262.

[138] Tang, Y. & Yin, J. (2017). Design of cut unit geometry in hierarchical kirigami-based auxetic metamaterials for high stretchability and compressibility. *Extreme Mechanics Letters*, 12, 77–85.

[139] Thompson, D. W. (1917). *On growth and form*. Cambridge University Press.

[140] Tutte, W. T. (1963). How to draw a graph. *Proceedings of the London Mathematical Society*, 3(1), 743–767.

[141] Valkov, B., Rycroft, C. H., & Kamrin, K. (2015). Eulerian method for multiphase interactions of soft solid bodies in fluids. *Journal of Applied Mechanics*, 82, 041011.

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

[143] Waitukaitis, S., Menaut, R., Chen, B. G.-g., & van Hecke, M. (2015). Origami multistability: From single vertices to metasheets. *Physical Review Letters*, 114(5), 055503.

[144] Wake, D. B. & Vredenburg, V. T. (2008). 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), 11466–11473.

[145] Wannarong, T., Parraga, G., Buchanan, D., Fenster, A., House, A. A., Hackam, D. G., & Spence, J. D. (2013). Progression of carotid plaque volume predicts cardiovascular events. *Stroke*, 44, 1859–1865.

[146] Wei, Z. Y., Guo, Z. V., Dudte, L., Liang, H. Y., & Mahadevan, L. (2013). Geometric mechanics of periodic pleated origami. *Physical Review Letters*, 110(21), 215501.

[147] Whitney, H. (1957). *Geometric integration theory*. Princeton University Press.

[148] Yang, Y.-L., Guo, R., Luo, F., Hu, S.-M., & Gu, X. (2009). Generalized discrete Ricci flow. *Computer Graphics Forum*, 28(7), 2005–2014.

[149] Zelditch, M. L., Swiderski, D. L., & Sheets, H. D. (2012). *Geometric morphometrics for biologists: a primer*. Academic Press.

[150] Zhang, Y., Yan, Z., Nan, K., Xiao, D., Liu, Y., Luan, H., Fu, H., Wang, X., Yang, Q., Wang, J., et al. (2015). A mechanically driven form of kirigami as a route to 3D mesostructures in micro/nanomembranes. *Proceedings of the National Academy of Sciences*, 112(38), 11757–11764.

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

[152] Zhu, L., Haker, S., & Tannenbaum, A. (2005). Flattening maps for the visualization of multibranched vessels. *IEEE Transactions on Medical Imaging*, 24, 191–198.

[153] Zou, G., Hu, J., Gu, X., & Hua, J. (2011). Authalic parameterization of general surfaces using lie advection. *IEEE Transactions on Visualization and Computer Graphics*, 17(12), 2005–2014.