



---

*Research article*

## Riemannian gradient descent for spherical area-preserving mappings

Marco Sutti<sup>1,\*</sup> and Mei-Heng Yueh<sup>2,\*</sup>

<sup>1</sup> Mathematics Division, National Center for Theoretical Sciences, Taipei, Taiwan

<sup>2</sup> Department of Mathematics, National Taiwan Normal University, Taipei, Taiwan

\* **Correspondence:** Email: [msutti@ncts.ntu.edu.tw](mailto:msutti@ncts.ntu.edu.tw), [yue@ntnu.edu.tw](mailto:yue@ntnu.edu.tw).

**Abstract:** We propose a new Riemannian gradient descent method for computing spherical area-preserving mappings of topological spheres using a Riemannian retraction-based framework with theoretically guaranteed convergence. The objective function is based on the stretch energy functional, and the minimization is constrained on a power manifold of unit spheres embedded in three-dimensional Euclidean space. Numerical experiments on several mesh models demonstrate the accuracy and stability of the proposed framework. Comparisons with three existing state-of-the-art methods for computing area-preserving mappings demonstrate that our algorithm is both competitive and more efficient. Finally, we present a concrete application to the problem of landmark-aligned surface registration of two brain models.

**Keywords:** stretch-energy functional; area-preserving mapping; Riemannian optimization; Riemannian gradient descent; matrix manifolds

**Mathematics Subject Classification:** 68U05, 65K10, 65D18, 65D19

---

### 1. Introduction

This paper proposes a new method for computing spherical area-preserving mappings of topological spheres using a Riemannian optimization framework.

Area-preserving mappings can serve as parameterization of surfaces and have been applied to the field of computer graphics and medical imaging [6]. State-of-the-art methods for the computation of area-preserving parameterizations include diffusion-based methods [8], optimal mass transportation-based methods [18], and nonlinear optimization methods [20]. It is worth noting that most previous works consider area-preserving parameterization from simply connected open surfaces to planar regions. For the spherical area-preserving mapping of genus-zero closed surfaces, recently developed methods include the adaptive area-preserving parameterization method [7], the spherical optimal transportation mapping [10], and the stretch energy minimization method [21].

Riemannian optimization generalizes unconstrained, continuous optimization (defined on Euclidean space) following the same principle that Riemannian geometry is a generalization of Euclidean space. In traditional optimization methods, nonlinear matrix manifolds rely on the Euclidean vector space structure. In contrast, in the Riemannian optimization framework, algorithms and convergence analysis are formulated based on the language of differential geometry, and numerical linear algebra techniques are then used for implementation. Riemannian optimization focuses on matrix manifolds, which are manifolds in the sense of classical Riemannian geometry that admit a natural representation of their elements in the form of matrices. Retraction mappings play a vital role in this framework, as they are used to turn objective functions defined in Euclidean space into functions defined on an abstract manifold so that constraints are taken into account explicitly. A vast body of literature on the theory and implementation of Riemannian optimization now exists. Some of the earliest references for this field, especially for line-search methods, go back to Luenberger [15], Gabay [14], and Udriște [19]. More recent literature that encompasses the findings and developments of the last three decades is [2, 5, 12].

Our Riemannian gradient descent (RGD) method involves classical components from computational geometry (simplicial surfaces and mappings) and Riemannian optimization (line search, retractions, and Riemannian gradients). To the best of the authors' knowledge, this is the first time this approach has been used in computational geometry. In this paper, we minimize the normalized stretch energy [20] for simplicial mappings subject to the constraint that the image of the mapping belongs to a power manifold of  $n$  unit spheres embedded in  $\mathbb{R}^3$ .

### 1.1. Contributions

In this paper, we propose an RGD method to compute spherical area-preserving mappings on a power manifold of unit spheres. In particular, the main contributions are as follows:

- (i) We combine the tools from the Riemannian optimization framework and the components from computational geometry to propose an RGD method for computing spherical area-preserving mappings of topological spheres.
- (ii) We explore two different line-search strategies: one using MATLAB's `fminbnd`, and the other using the quadratic/cubic interpolant approximation from [11, §6.3.2].
- (iii) We conduct extensive numerical experiments on several mesh models to demonstrate the accuracy and efficiency of the algorithm.
- (iv) We demonstrate the competitiveness and efficiency of our algorithm over three state-of-the-art methods for computing area-preserving mappings.
- (v) We show that our algorithm is stable with respect to small perturbations in the initial mesh model.
- (vi) We apply the algorithm to the concrete application of landmark-aligned surface registration between two human brain models.

### 1.2. Outline of the paper

The remaining part of this paper is organized as follows. Section 2 introduces the main concepts on simplicial surfaces and mappings and presents the formulation of the objective function. Section 3

provides some preliminaries on the Riemannian optimization framework, briefly recalls the geometry of the unit sphere, and then describes the tools needed to perform optimization on the power manifold. Section 4 is about the RGD method and reports known convergence results. Section 5 discusses the extensive numerical experiments to evaluate our algorithm in terms of accuracy and efficiency, and provides a concrete application. Finally, we wrap our paper with conclusions and a future outlook in Section 6. The appendices contain other details about the calculations of the area of the simplicial mapping, the Hessian of the objective function, and the line-search procedure used in the RGD method.

### 1.3. Notation

In this section, we list the notations and symbols adopted in order of appearance in the paper. Symbols specific to a particular section are usually not included in this list.

$\tau$	A triangular face
$ \tau $	The area of the triangle $\tau$
$\mathcal{M}$	Simplicial surface
$\mathcal{V}(\mathcal{M})$	Set of vertices of $\mathcal{M}$
$\mathcal{F}(\mathcal{M})$	Set of faces of $\mathcal{M}$
$\mathcal{E}(\mathcal{M})$	Set of edges of $\mathcal{M}$
$v_i, v_j, v_k$	Vertices of a triangular face
$f$	Simplicial mapping
$\mathbf{f}$	Representative matrix of $f$
$\mathbf{f}_\ell$	Coordinates of a vertex $f(v_\ell)$
$\text{vec}$	Column-stacking vectorization operator
$S^2$	Unit sphere in $\mathbb{R}^3$
$(S^2)^n$	Power manifold of $n$ unit spheres in $\mathbb{R}^3$
$E_A(f)$	The authalic energy
$E_S(f)$	The stretch energy
$L_S(f)$	Weighted Laplacian matrix
$\omega_S$	Modified cotangent weights
$\mathcal{A}(f)$	Area of the image of $f$
$E(f)$	The normalized stretch energy
$T_{\mathbf{x}}S^2$	Tangent space to $S^2$ at $\mathbf{x}$
$P_{T_{\mathbf{x}}S^2}$	Orthogonal projector onto the tangent space to $S^2$ at $\mathbf{x}$
$P_{T_{\mathbf{f}_\ell}(S^2)^n}$	Orthogonal projector onto the tangent space to $(S^2)^n$ at $\mathbf{f}_\ell$
$R$	Retraction mapping
$\nabla E(f)$	Euclidean gradient of $E(f)$
$\text{grad } E(f)$	Riemannian gradient of $E(f)$

## 2. Simplicial surfaces, mappings, and objective function

We introduce the simplicial surfaces and mappings in subsection 2.1 and the objective function in subsection 2.2.

### 2.1. Simplicial surfaces and mappings

A simplicial surface  $\mathcal{M}$  is the underlying set of a simplicial 2-complex  $\mathcal{K}(\mathcal{M}) = \mathcal{F}(\mathcal{M}) \cup \mathcal{E}(\mathcal{M}) \cup \mathcal{V}(\mathcal{M})$  composed of vertices

$$\mathcal{V}(\mathcal{M}) = \left\{ v_\ell = (v_\ell^1, v_\ell^2, v_\ell^3)^\top \in \mathbb{R}^3 \right\}_{\ell=1}^n,$$

oriented triangular faces

$$\mathcal{F}(\mathcal{M}) = \left\{ \tau_\ell = [v_{i_\ell}, v_{j_\ell}, v_{k_\ell}] \mid v_{i_\ell}, v_{j_\ell}, v_{k_\ell} \in \mathcal{V}(\mathcal{M}) \right\}_{\ell=1}^m,$$

and undirected edges

$$\mathcal{E}(\mathcal{M}) = \left\{ [v_i, v_j] \mid [v_i, v_j, v_k] \in \mathcal{F}(\mathcal{M}) \text{ for some } v_k \in \mathcal{V}(\mathcal{M}) \right\}.$$

A simplicial mapping  $f: \mathcal{M} \rightarrow \mathbb{R}^3$  is a particular type of piecewise affine mapping with the restriction mapping  $f|_\tau$  being affine, for every  $\tau \in \mathcal{F}(\mathcal{M})$ . We denote

$$\mathbf{f}_\ell := f(v_\ell) = (f_\ell^1, f_\ell^2, f_\ell^3)^\top, \text{ for every } v_\ell \in \mathcal{V}(\mathcal{M}).$$

The mapping  $f$  can be represented as a matrix

$$\mathbf{f} = \begin{bmatrix} \mathbf{f}_1^\top \\ \vdots \\ \mathbf{f}_n^\top \end{bmatrix} = \begin{bmatrix} f_1^1 & f_1^2 & f_1^3 \\ \vdots & \vdots & \vdots \\ f_n^1 & f_n^2 & f_n^3 \end{bmatrix} =: [\mathbf{f}^1 \quad \mathbf{f}^2 \quad \mathbf{f}^3], \quad (2.1)$$

or a vector

$$\text{vec}(\mathbf{f}) = \begin{bmatrix} \mathbf{f}^1 \\ \mathbf{f}^2 \\ \mathbf{f}^3 \end{bmatrix}.$$

### 2.2. The objective function

The authalic energy for simplicial mappings  $f: \mathcal{M} \rightarrow \mathbb{R}^3$  is defined as [20]

$$E_A(f) = E_S(f) - \mathcal{A}(f),$$

where  $E_S$  is the stretch energy defined as

$$E_S(f) = \frac{1}{2} \text{vec}(\mathbf{f})^\top (I_3 \otimes L_S(f)) \text{vec}(\mathbf{f}),$$

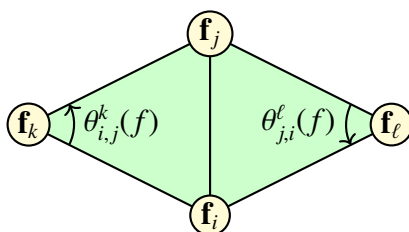
where  $I_3$  is the identity matrix of size 3-by-3,  $\otimes$  denotes the Kronecker product, and  $L_S(f)$  is the weighted Laplacian matrix  $L_S(f)$ . The latter is defined by

$$[L_S(f)]_{i,j} = \begin{cases} -\sum_{[v_i, v_j, v_k] \in \mathcal{F}(\mathcal{M})} [\omega_S(f)]_{i,j,k} & \text{if } [v_i, v_j] \in \mathcal{E}(\mathcal{M}), \\ -\sum_{\ell \neq i} [L_S(f)]_{i,\ell} & \text{if } j = i, \\ 0 & \text{otherwise,} \end{cases} \quad (2.2)$$

in which  $\omega_S(f)$  is the modified cotangent weight defined as

$$[\omega_S(f)]_{i,j,k} = \frac{\cot(\theta_{i,j}^k(f)) |f([v_i, v_j, v_k])|}{2|[v_i, v_j, v_k]|}, \quad (2.3)$$

with  $\theta_{i,j}^k(f)$  being the angle opposite to the edge  $f([v_i, v_j])$  at the point  $f(v_k)$  on the image  $f(\mathcal{M})$ , as illustrated in Figure 1.



**Figure 1.** An illustration of the cotangent weight defined on the image of  $f$ .

It is proved that  $E_A(f) \geq 0$  and the equality holds if and only if  $f$  preserves the area [20, Corollary 3.4]. Due to the optimization process, the image area  $\mathcal{A}(f)$  is not constant, hence we introduce a prefactor  $|\mathcal{M}|/\mathcal{A}(f)$  and define the *normalized stretch energy* as

$$E(f) = \frac{|\mathcal{M}|}{\mathcal{A}(f)} E_S(f). \quad (2.4)$$

To perform numerical optimization via the RGD method, we need to compute the Euclidean gradient. By applying the formula  $\nabla E_S(f) = 2(I_3 \otimes L_S(f)) \text{vec}(\mathbf{f})$  from [20, (3.6)], the gradient of  $E(f)$  can be formulated as

$$\begin{aligned} \nabla E(f) &= \nabla \left( \frac{|\mathcal{M}|}{\mathcal{A}(f)} E_S(f) \right) \\ &= \frac{|\mathcal{M}|}{\mathcal{A}(f)} \nabla E_S(f) + E_S(f) \nabla \frac{|\mathcal{M}|}{\mathcal{A}(f)} \\ &= \frac{2|\mathcal{M}|}{\mathcal{A}(f)} (I_3 \otimes L_S(f)) \text{vec}(\mathbf{f}) - \frac{|\mathcal{M}| E_S(f)}{\mathcal{A}(f)^2} \nabla \mathcal{A}(f). \end{aligned} \quad (2.5)$$

The following proposition gives an explicit formula for calculating  $\nabla \mathcal{A}(f)$ .

**Proposition 2.1** (Formula for  $\nabla \mathcal{A}$ ). *The gradient of  $\mathcal{A}$  can be explicitly formulated as*

$$\nabla \mathcal{A}(f|_\tau) = \frac{|\tau|}{\mathcal{A}(f|_\tau)} \text{vec}(L_S(f|_\tau) \mathbf{f}_\tau). \quad (2.6)$$

*Proof.* By applying the explicit formulas  $E_S(f|_\tau) = \frac{\mathcal{A}(f|_\tau)^2}{|\tau|}$  and  $\nabla E_S(f|_\tau) = 2 \operatorname{vec}(L_S(f|_\tau) \mathbf{f}_\tau)$  from [20], the chain rule yields

$$2 \operatorname{vec}(L_S(f|_\tau) \mathbf{f}_\tau) = \nabla E_S(f|_\tau) = \frac{2\mathcal{A}(f|_\tau)}{|\tau|} \nabla \mathcal{A}(f|_\tau),$$

which is equivalent to (2.6).  $\square$

Other details about the calculation of  $\mathcal{A}(f)$  are reported in Appendix A.

### 3. Riemannian optimization framework and geometry

The *Riemannian optimization framework* [2, 5, 12] solves constrained optimization problems where the constraints have a geometric nature. This approach utilizes the underlying geometric structure of the problems, which allows the constraints to be taken explicitly into account. The optimization variables are constrained to a smooth manifold, and the optimization is performed on that manifold. Typically, the manifolds considered are matrix manifolds, meaning there is a natural representation of their elements in matrix form. In particular, in this paper, the problem is formulated on a power manifold of  $n$  unit spheres embedded in  $\mathbb{R}^3$ , and we use the RGD method for minimizing the cost function (2.4) on this power manifold.

Traditional optimization methods rely on the Euclidean vector space structure. For instance, the steepest descent method for minimizing a function  $g: \mathbb{R}^n \rightarrow \mathbb{R}$  updates the current iterate  $\mathbf{x}_k$  by moving in the direction  $\mathbf{d}_k$  of the anti-gradient of  $g$ , by a step size  $\alpha_k$  chosen according to an appropriate line-search rule. However, on a nonlinear manifold, the vector addition  $\mathbf{x}_k + \alpha_k \mathbf{d}_k$  does not make sense in general due to the manifold curvature. We need the notion of tangent vectors and their length to generalize the steepest descent direction to a Riemannian manifold. The retraction mapping is also a vital tool in the Riemannian optimization framework. It is a mapping from the tangent space to the manifold, used to transform objective functions defined in Euclidean space into functions defined on a manifold while explicitly taking the constraints into account.

Similarly to the line-search method in Euclidean space, a line-search method in the Riemannian framework determines at a current iterate  $\mathbf{x}_k$  on a manifold  $M$  a search direction  $\xi$  on the tangent space  $T_{\mathbf{x}_k}M$ . The next iterate  $\mathbf{x}_{k+1}$  is then determined by a line search along a curve  $\alpha \mapsto R_{\mathbf{x}_k}(\alpha \xi)$  where  $R_{\mathbf{x}}: T_{\mathbf{x}}M \rightarrow M$  is the retraction mapping. The procedure is then repeated for  $\mathbf{x}_{k+1}$  taking the role of  $\mathbf{x}_k$ . Similarly to optimization methods in Euclidean space, search directions can be the negative of the Riemannian gradient, leading to the Riemannian steepest descent method. Other choices of search directions lead to other methods, e.g., Riemannian versions of the trust-region method [1] or BFGS [17].

In what follows, we introduce some fundamental geometry concepts used in Riemannian optimization, which are necessary to formulate the RGD method for our problems. We first briefly recall the geometry of the unit sphere  $S^2$  embedded in  $\mathbb{R}^3$ , and then we switch to the power manifold of  $n$  unit spheres, denoted by  $(S^2)^n$ .

#### 3.1. Geometry of the unit sphere $S^2$

The unit sphere  $S^2$  is a Riemannian submanifold of  $\mathbb{R}^3$  defined as

$$S^2 = \{\mathbf{x} \in \mathbb{R}^3: \mathbf{x}^\top \mathbf{x} = 1\}.$$

The Riemannian metric (inner product) on the sphere is inherited from the embedding space  $\mathbb{R}^3$ , i.e.,

$$\langle \xi, \eta \rangle_{\mathbf{x}} = \xi^{\top} \eta, \quad \xi, \eta \in T_{\mathbf{x}}S^2,$$

where  $T_{\mathbf{x}}S^2$  is the tangent space to  $S^2$  at  $\mathbf{x} \in S^2$ , defined as the set of all vectors orthogonal to  $\mathbf{x}$  in  $\mathbb{R}^3$ , i.e.,

$$T_{\mathbf{x}}S^2 = \{\mathbf{z} \in \mathbb{R}^3 : \mathbf{x}^{\top} \mathbf{z} = 0\}.$$

The projector  $P_{T_{\mathbf{x}}S^2}$  from  $\mathbb{R}^3$  onto the tangent space  $T_{\mathbf{x}}S^2$  is a mapping  $P_{T_{\mathbf{x}}S^2}: \mathbb{R}^3 \rightarrow T_{\mathbf{x}}S^2$ , defined by

$$P_{T_{\mathbf{x}}S^2}(\mathbf{z}) = (I_3 - \mathbf{x}\mathbf{x}^{\top})\mathbf{z}. \quad (3.1)$$

In the following, points on the unit sphere are denoted by  $\mathbf{f}_{\ell}$  (the vertices of the simplicial mapping  $f$ ), and tangent vectors are represented by  $\xi_{\ell}$ .

### 3.2. Geometry of the power manifold $(S^2)^n$

We aim to minimize the function  $E(f) = E(\mathbf{f}_1, \dots, \mathbf{f}_n)$  (2.4), where each  $\mathbf{f}_{\ell}$ ,  $\ell = 1, \dots, n$ , lives on the same manifold  $S^2$ . This leads us to consider the *power manifold* of  $n$  unit spheres

$$(S^2)^n = \underbrace{S^2 \times S^2 \times \dots \times S^2}_{n \text{ times}},$$

with the metric of  $S^2$  extended elementwise. In the remaining part of this section, we present the tools from Riemannian geometry needed to generalize gradient descent to this manifold. The projector onto the tangent space is used to compute the Riemannian gradient, as it will be explained in Section 4. The projection onto the power manifold turns points of  $\mathbb{R}^{n \times 3}$  into points of  $(S^2)^n$ . Finally, the retraction turns an objective function defined on  $\mathbb{R}^{n \times 3}$  into an objective function defined on the manifold  $(S^2)^n$ .

#### 3.2.1. Projection onto the tangent space

As seen above, the projector onto the tangent space to the unit sphere at the point  $\mathbf{x}$  is given by (3.1). Here, the points are denoted by  $\mathbf{f}_{\ell} \in \mathbb{R}^3$ ,  $\ell = 1, \dots, n$ , so we write

$$P_{T_{\mathbf{f}_{\ell}}S^2} = I_3 - \mathbf{f}_{\ell}\mathbf{f}_{\ell}^{\top}.$$

It clearly changes for every vertex  $\mathbf{f}_{\ell}$ . The projector from  $\mathbb{R}^{n \times 3}$  onto the tangent space at  $\mathbf{f}$  to the power manifold  $(S^2)^n$  is a mapping

$$P_{T_{\mathbf{f}}(S^2)^n}: \mathbb{R}^{n \times 3} \rightarrow T_{\mathbf{f}}(S^2)^n,$$

and can be represented by a block diagonal matrix of size  $3n \times 3n$ , i.e.,

$$P_{T_{\mathbf{f}}(S^2)^n} := \text{blkdiag}(P_{T_{\mathbf{f}_1}S^2}, P_{T_{\mathbf{f}_2}S^2}, \dots, P_{T_{\mathbf{f}_n}S^2}) = \begin{bmatrix} P_{T_{\mathbf{f}_1}S^2} & & & \\ & P_{T_{\mathbf{f}_2}S^2} & & \\ & & \ddots & \\ & & & P_{T_{\mathbf{f}_n}S^2} \end{bmatrix}. \quad (3.2)$$

In practice, we never actually create this matrix. Instead, we implement an efficient version using vectorized operations (MATLAB's `bsxfun`).

### 3.2.2. Projection onto $(S^2)^n$

The projection of a single vertex  $\mathbf{f}_\ell$  from  $\mathbb{R}^3$  to the unit sphere  $S^2$  is given by the normalization

$$\tilde{\mathbf{f}}_\ell = \frac{\mathbf{f}_\ell}{\|\mathbf{f}_\ell\|_2}.$$

Hence, the projection of the whole of  $\mathbf{f}$  onto the power manifold  $(S^2)^n$  is given by

$$P_{(S^2)^n}: \mathbb{R}^{n \times 3} \rightarrow (S^2)^n,$$

defined by

$$\mathbf{f} \mapsto \tilde{\mathbf{f}} := \text{diag}\left(\frac{1}{\|\mathbf{f}_1\|_2}, \frac{1}{\|\mathbf{f}_2\|_2}, \dots, \frac{1}{\|\mathbf{f}_n\|_2}\right) [\mathbf{f}_1 \ \mathbf{f}_2 \ \dots \ \mathbf{f}_n]^\top.$$

Again, this representative matrix is only shown for illustrative purposes; in the actual implementation, we use row-wise normalization of  $\mathbf{f}$ .

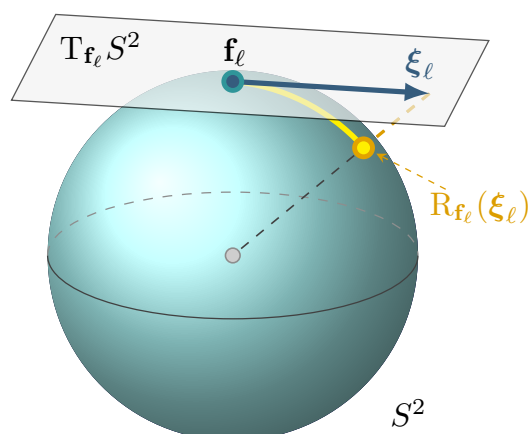
### 3.2.3. Retraction onto $(S^2)^n$

A retraction is a mapping from the tangent space to the manifold used to turn tangent vectors into points on the manifold and functions defined on the manifold into functions defined on the tangent space; see [2, 3] for more details.

The retraction of a single tangent vector  $\xi_\ell$  from  $T_{\mathbf{f}_\ell} S^2$  to  $S^2$  is a mapping  $R_{\mathbf{f}_\ell}: T_{\mathbf{f}_\ell} S^2 \rightarrow S^2$ , defined by [2, Example 4.1.1]

$$R_{\mathbf{f}_\ell}(\xi_\ell) = \frac{\mathbf{f}_\ell + \xi_\ell}{\|\mathbf{f}_\ell + \xi_\ell\|}.$$

Figure 2 provides an illustration of the retraction from  $T_{\mathbf{f}_\ell} S^2$  to  $S^2$ .



**Figure 2.** An illustration of the retraction mapping on the unit sphere  $S^2$ .

For the power manifold  $(S^2)^n$ , the retraction of all the tangent vectors  $\xi_\ell$ ,  $\ell = 1, \dots, n$ , is a mapping

$$R_{\mathbf{f}}: T_{\mathbf{f}}(S^2)^n \rightarrow (S^2)^n,$$



defined by

$$[\xi_1 \ \cdots \ \xi_n]^\top \mapsto \text{diag}\left(\frac{1}{\|\mathbf{f}_1 + \xi_1\|_2}, \dots, \frac{1}{\|\mathbf{f}_n + \xi_n\|_2}\right) [\mathbf{f}_1 + \xi_1 \ \cdots \ \mathbf{f}_n + \xi_n]^\top. \quad (3.3)$$

Again, this retraction is implemented by row-wise normalization of  $\mathbf{f} + \xi$ .

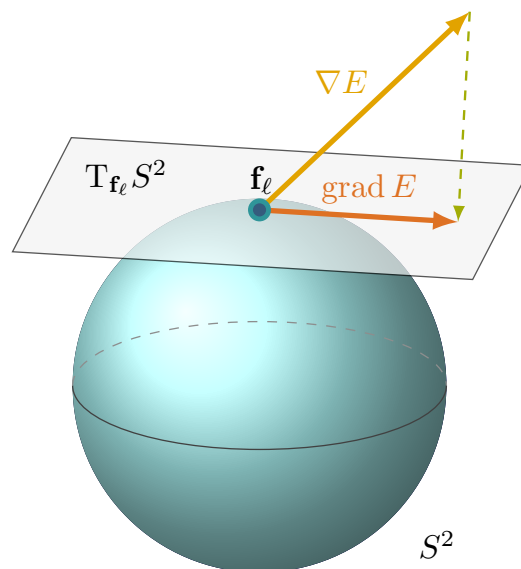
#### 4. Riemannian gradient descent method

We are now in the position of introducing the RGD method. In this section, we will first provide the formula for the Riemannian gradient on the power manifold  $(S^2)^n$ . Then, we will explain the RGD method by providing its pseudocode, and finally, we will recall the known theoretical results that ensure the convergence of RGD.

The Riemannian gradient of the objective function  $E$  in (2.4) is given by the projection onto  $T_{\mathbf{f}}(S^2)^n$  of the Euclidean gradient of  $E$ , namely,

$$\text{grad } E(f) = P_{T_{\mathbf{f}}(S^2)^n}(\nabla E(f)), \quad (4.1)$$

where  $\nabla E$  is explicitly formulated in (2.5). This is always the case for embedded submanifolds; see [2, §3.6.1]. Figure 3 illustrates the difference between the Euclidean and the Riemannian gradient for one point on the unit sphere.



**Figure 3.** Illustration of the difference between Euclidean and Riemannian gradient. The dashed green line represents the projection step.

Given an initial iterate  $f^{(0)} \in (S^2)^n$ , the RGD algorithm generates a sequence of iterates  $\{f^{(k)}\}$  as follows. At each iteration  $k = 0, 1, 2, \dots$ , it chooses a search direction  $\mathbf{d}^{(k)} = -\text{grad } E(f^{(k)})$  in the tangent space  $T_{f^{(k)}}(S^2)^n$  such that the sequence  $\{\mathbf{d}^{(k)}\}$  is gradient related. Then, the new point  $f^{(k+1)}$  is chosen such that it satisfies the sufficient decrease condition

$$E(f^{(k)}) - E(f^{(k+1)}) \geq c \left( E(f^{(k)}) - E(\mathbf{R}_{f^{(k)}}(\alpha_k \mathbf{d}^{(k)})) \right), \quad (4.2)$$

where  $c > 0$  and  $\alpha_k$  is the step size for the given  $\mathbf{d}^{(k)}$ .

The RGD method on the power manifold  $(S^2)^n$  is summarized in Algorithm 1. It has been adapted from [2, p. 63]. In practice, in our numerical experiments of Section 5, the initial mapping  $\mathbf{f}^{(0)} \in (S^2)^n$  is computed by applying the fixed-point iteration (FPI) method of [21] until the first increase in energy is detected; see subsection 5.3, Algorithm 2. The line-search procedure used in line 7 is described in detail in Appendix C.

---

**Algorithm 1:** The RGD method on  $(S^2)^n$ .

---

```

1 Given objective function  $E$ , power manifold  $(S^2)^n$ , initial iterate  $\mathbf{f}^{(0)} \in (S^2)^n$ , projector  $P_{T_{\mathbf{f}}(S^2)^n}$ 
   from  $\mathbb{R}^{n \times 3}$  to  $T_{\mathbf{f}}(S^2)^n$ , retraction  $R_{\mathbf{f}}$  from  $T_{\mathbf{f}}(S^2)^n$  to  $(S^2)^n$ ;
   Result: Sequence of iterates  $\{f^{(k)}\}$ .
2  $k \leftarrow 0$ ;
3 while  $f^{(k)}$  does not sufficiently minimize  $E$  do
4   Compute the Euclidean gradient of the objective function  $\nabla E(f^{(k)})$  (2.5);
5   Compute the Riemannian gradient as  $\text{grad } E(f^{(k)}) = P_{T_{\mathbf{f}^{(k)}}(S^2)^n}(\nabla E(f^{(k)}))$ ;
6   Choose the anti-gradient direction  $\mathbf{d}^{(k)} = -\text{grad } E(f^{(k)})$ ;
7   Use a line-search procedure to compute a step size  $\alpha_k > 0$  that satisfies the sufficient
   decrease condition (4.2); see Appendix C;
8   Set  $\mathbf{f}^{(k+1)} = R_{\mathbf{f}^{(k)}}(\alpha_k \mathbf{d}^{(k)})$ ;
9    $k \leftarrow k + 1$ ;
10 end while

```

---

### Known convergence results

The RGD method has theoretically guaranteed convergence. For the reader's convenience, we report the two main results on the convergence of RGD. The first result is about the convergence of Algorithm 1 to critical points of the objective function. The second result is about the convergence of RGD to a local minimizer with a line-search technique.

**Theorem 4.1** ([2, Theorem 4.3.1]). *Let  $\{f^{(k)}\}$  be an infinite sequence of iterates generated by Algorithm 1. Then, every accumulation point  $f^{(*)}$  of  $\{f^{(k)}\}$  is a critical point of the cost function  $E$ .*

**Remark 4.1.** *We are implicitly saying that a sequence can have more than one accumulation point; for example, from a sequence  $\{f^{(k)}\}$ , we may extract two subsequences such that they have two distinct accumulation points.*

The proof of Theorem 4.1 can be done by contradiction, but it remains pretty technical, so we refer the interested reader to [2, p. 65]. It should be pointed out that Theorem 4.1 only guarantees the convergence to critical points. It does not tell us anything about their nature, i.e., whether the critical points are local minimizers, local maximizers, or saddle points. However, if the smallest eigenvalue of the Hessian of  $E$  at  $f^{(*)}$ ,  $\lambda_{H,\min} > 0$ , then the critical point  $f^{(*)}$  is a local minimizer of  $E$ . Under this assumption (i.e., that  $\lambda_{H,\min} > 0$ ), [2, §4.5.2] gives an asymptotic convergence bound for Algorithm 1.

Indeed, the following result uses the smallest and largest eigenvalues of the Hessian of the objective function at a critical point  $f^{(*)}$ .

**Theorem 4.2** ([2, Theorem 4.5.6]). *Let  $\{f^{(k)}\}$  be an infinite sequence of iterates generated by Algorithm 1, converging to a point  $f^{(*)}$ . (By Theorem 4.1,  $f^{(*)}$  is a critical point of  $E$ .) Let  $\lambda_{H,\min}$  and  $\lambda_{H,\max}$  be the smallest and largest eigenvalues of the Hessian of  $E$  at  $f^{(*)}$ . Assume that  $\lambda_{H,\min} > 0$  (hence  $f^{(*)}$  is a local minimizer of  $E$ ). Then, given  $r$  in the interval  $(r_*, 1)$  with  $r_* = 1 - \min\left(2\sigma\bar{\alpha}\lambda_{H,\min}, 4\sigma(1 - \sigma)\beta\frac{\lambda_{H,\min}}{\lambda_{H,\max}}\right)$ , there exists an integer  $K \geq 0$  such that*

$$E(f^{(k+1)}) - E(f^{(*)}) \leq (r + (1 - r)(1 - c))(E(f^{(k)}) - E(f^{(*)})), \quad (4.3)$$

for all  $k \geq K$ , where  $c$  is the parameter in (4.2) of Algorithm 1. Note that  $0 < r_* < 1$  since  $\beta, \sigma \in (0, 1)$ .

As noted in [2, p. 71], in typical cases of Algorithm 1, the constant  $c$  in the descent condition (4.2) equals 1, hence (4.3) reduces to  $E(f^{(k+1)}) - E(f^{(*)}) \leq r(E(f^{(k)}) - E(f^{(*)}))$ .

## 5. Numerical experiments

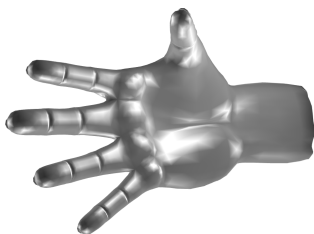
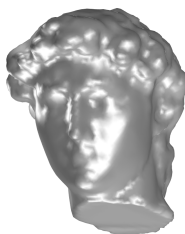
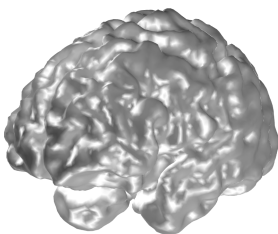

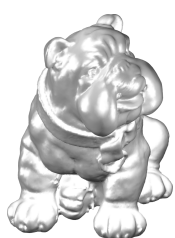





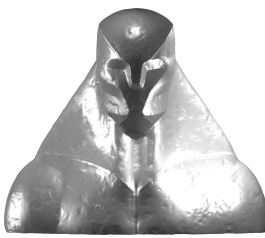

In this section, we demonstrate the convergence behavior of the RGD method using twelve mesh models and two line-search techniques. We present numerical results in tables and provide convergence plots. The computational time cost of the RGD in every table does not include the time cost for the initial mapping by the FPI method. In subsection 5.2, we introduce a correction for bijectivity, which helps to unfold the folding triangles. We then compare RGD to the FPI method of [21] in subsection 5.3 and the adaptive area-preserving parameterization method of [7] in subsection 5.4. In subsection 5.5, we compare our RGD to the spherical optimal transportation mapping proposed by Cui et al. [10]. Then, in subsection 5.6, we show that the algorithm is robust to noise by starting the algorithm from an initial guess with small perturbations. Finally, in subsection 5.7, we apply our algorithm to the concrete application of surface registration of two brain models.

We conducted all our experiments on a laptop Lenovo ThinkPad T460s, with Windows 10 Pro and MATLAB R2021a installed, with Intel Core i7-6600 CPU, 20GB RAM, and Mesa Intel HD Graphics 520. The benchmark triangular mesh models used in our numerical experiments are shown in Figure 4, arranged per increasing number of vertices and faces, from the top left to bottom right.

### 5.1. Convergence behavior of RGD

To provide the RGD method with a good initial mapping, we first apply the FPI method of [21], described in subsection 5.3. Specifically, we apply the FPI until the first increase in energy occurs. We adopted two different line-search strategies: one that uses MATLAB's `fminbnd` and another that uses the quadratic/cubic interpolant of [11, §6.3.2], described in Appendix C.

In all the experiments, we monitor the authalic energy  $E_A(f) := E_S(f) - \mathcal{A}(f)$  defined in subsection 2.2 instead of the normalized stretch energy (2.4) because when  $E_A = 0$ , we know from the theory that  $f$  is an area-preserving mapping. Strictly speaking, in practice, we never obtain a mapping that exactly preserves the area, but we obtain a mapping that is area-distortion minimizing, since  $E_A$  is never identically zero. We also monitor the ratio between the standard deviation and the mean of the area ratio. This quantity has been considered in [8]. Finally, the computational time is always reported in seconds, and #Fs denotes the number of folding triangles.

Model Name	Right Hand	David Head	Cortical Surface	Bull
# Faces	8,808	21,338	30,000	34,504
# Vertices	4,406	10,671	15,002	17,254
				
Model Name	Bulldog	Lion Statue	Gargoyle	Max Planck
# Faces	99,590	100,000	100,000	102,212
# Vertices	49,797	50,002	50,002	51,108
				
Model Name	Bunny	Chess King	Art Statuette	Bimba
# Faces	111,364	263,712	895,274	1,005,146
# Vertices	55,684	131,858	447,639	502,575
				

**Figure 4.** The benchmark triangular mesh models used in this paper.

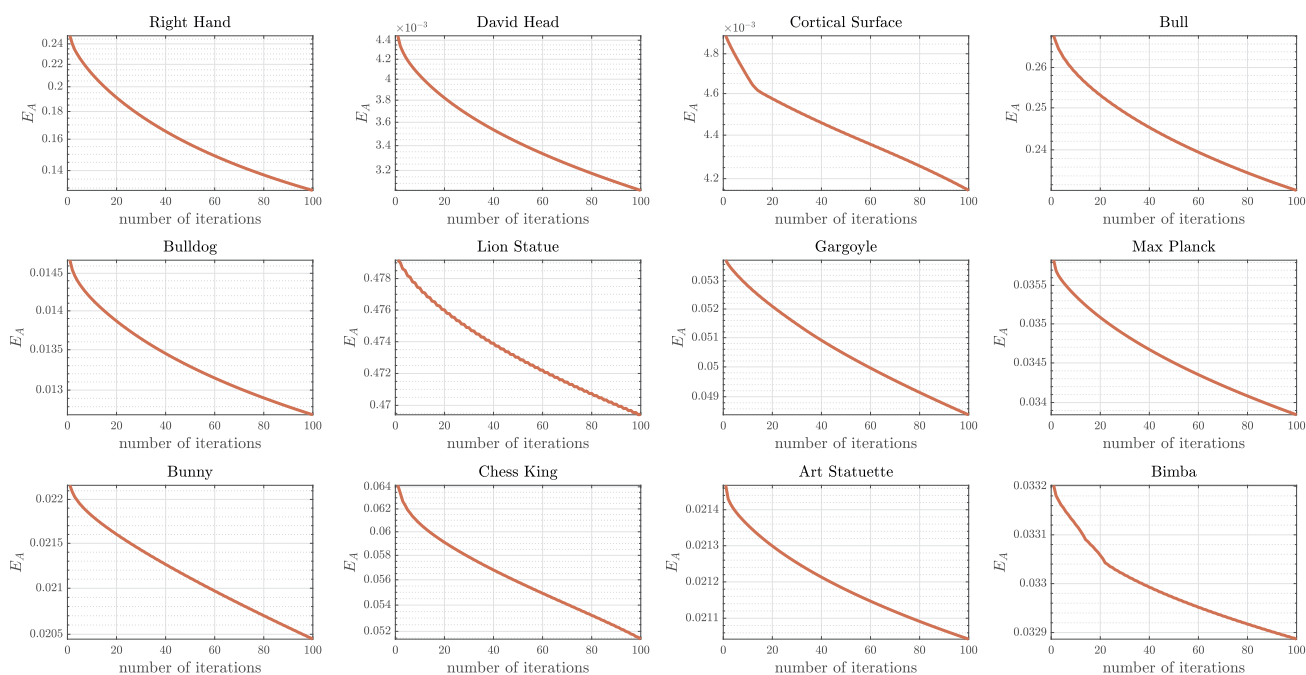
Tables 1 and 2 report the numerical results for RGD for minimizing the normalized stretch energy  $E$ , when run for a fixed maximum number of iterations (10 on the left and 100 on the right). Table 1 is for the RGD that uses the `fminbnd` line-search strategy, while Table 2 is for the RGD that uses the quadratic/cubic interpolant from [11, §6.3.2]. Similarly to the tables, Figures 5 and 6 illustrate the convergence behavior of RGD in the same setting when run for 100 iterations.

**Table 1.** RGD for minimizing the normalized stretch energy  $E$ . Line-search strategy: `fminbnd`.

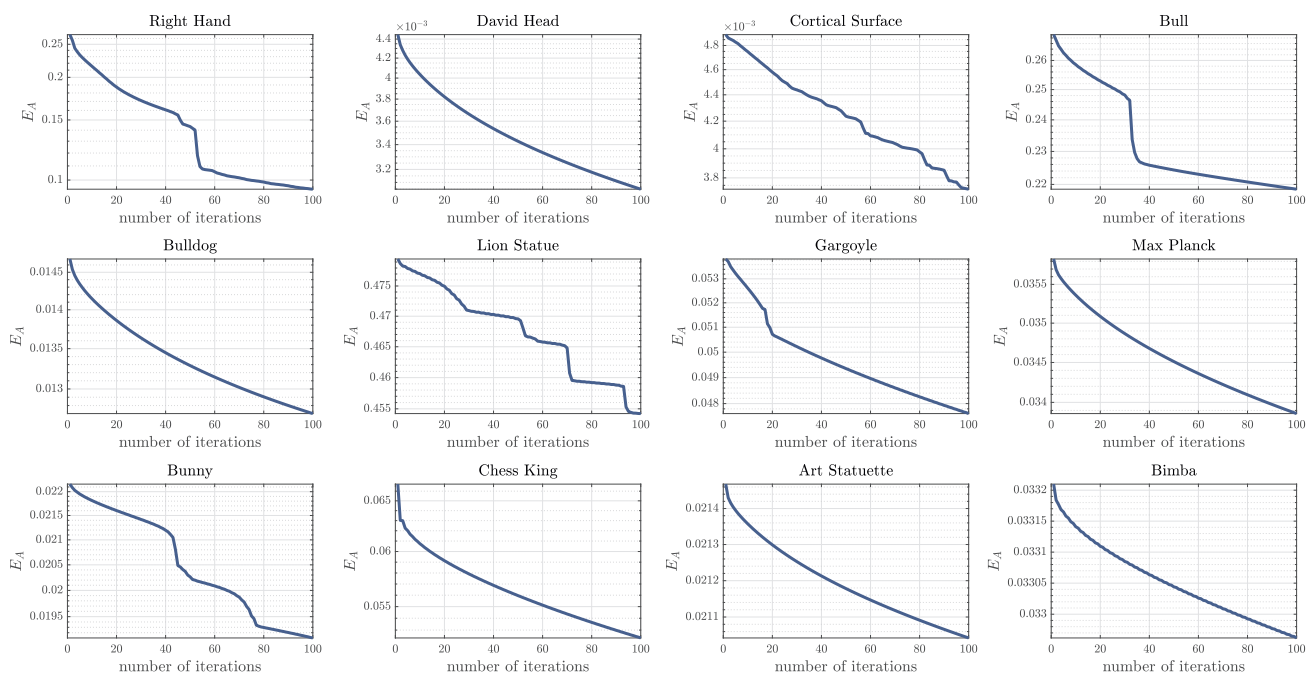
Model Name	10 Iterations				100 Iterations			
	SD/Mean	$E_A(f)$	Time	#Fs	SD/Mean	$E_A(f)$	Time	#Fs
Right Hand	0.1950	$2.17 \times 10^{-1}$	0.85	4	0.1459	$1.28 \times 10^{-1}$	9.37	2
David Head	0.0178	$4.04 \times 10^{-3}$	1.66	0	0.0156	$3.05 \times 10^{-3}$	14.95	0
Cortical Surface	0.0214	$4.68 \times 10^{-3}$	3.62	0	0.0200	$4.15 \times 10^{-3}$	32.40	0
Bull	0.1492	$2.58 \times 10^{-1}$	4.99	4	0.1380	$2.31 \times 10^{-1}$	43.27	1
Bulldog	0.0369	$1.41 \times 10^{-2}$	13.59	0	0.0343	$1.27 \times 10^{-2}$	136.63	0
Lion Statue	0.1935	$4.77 \times 10^{-1}$	16.83	0	0.1922	$4.69 \times 10^{-1}$	160.20	0
Gargoyle	0.0690	$5.28 \times 10^{-2}$	19.55	0	0.0653	$4.84 \times 10^{-2}$	192.62	0
Max Planck	0.0537	$3.54 \times 10^{-2}$	16.52	0	0.0525	$3.38 \times 10^{-2}$	161.01	0
Bunny	0.0417	$2.18 \times 10^{-2}$	20.59	0	0.0404	$2.04 \times 10^{-2}$	226.99	0
Chess King	0.0687	$6.07 \times 10^{-2}$	52.36	21	0.0639	$5.14 \times 10^{-2}$	518.18	17
Art Statuette	0.0408	$2.14 \times 10^{-2}$	140.59	0	0.0405	$2.10 \times 10^{-2}$	1 111.67	0
Bimba Statue	0.0514	$3.31 \times 10^{-2}$	270.63	1	0.0511	$3.29 \times 10^{-2}$	2 320.19	1

**Table 2.** RGD for minimizing the normalized stretch energy  $E$ . Line-search strategy: quadratic/cubic approximation from [11, §6.3.2].

Model Name	10 Iterations				100 Iterations			
	SD/Mean	$E_A(f)$	Time	#Fs	SD/Mean	$E_A(f)$	Time	#Fs
Right Hand	0.1936	$2.16 \times 10^{-1}$	0.36	4	0.1204	$9.40 \times 10^{-2}$	4.07	1
David Head	0.0178	$4.04 \times 10^{-3}$	0.99	0	0.0156	$3.04 \times 10^{-3}$	9.16	0
Cortical Surface	0.0216	$4.75 \times 10^{-3}$	1.40	0	0.0200	$3.72 \times 10^{-3}$	16.01	0
Bull	0.1492	$2.59 \times 10^{-1}$	1.77	4	0.1348	$2.19 \times 10^{-1}$	18.89	1
Bulldog	0.0369	$1.41 \times 10^{-2}$	6.60	0	0.0343	$1.27 \times 10^{-2}$	61.93	0
Lion Statue	0.1935	$4.77 \times 10^{-1}$	7.75	0	0.1894	$4.54 \times 10^{-1}$	76.76	0
Gargoyle	0.0688	$5.26 \times 10^{-2}$	7.81	0	0.0646	$4.76 \times 10^{-2}$	80.52	0
Max Planck	0.0537	$3.54 \times 10^{-2}$	7.18	0	0.0525	$3.39 \times 10^{-2}$	75.60	0
Bunny	0.0417	$2.18 \times 10^{-2}$	8.30	0	0.0390	$1.91 \times 10^{-2}$	89.62	0
Chess King	0.0692	$6.07 \times 10^{-2}$	20.06	21	0.0647	$5.23 \times 10^{-2}$	207.47	17
Art Statuette	0.0408	$2.14 \times 10^{-2}$	57.71	0	0.0405	$2.10 \times 10^{-2}$	654.57	0
Bimba Statue	0.0514	$3.31 \times 10^{-2}$	70.83	1	0.0512	$3.29 \times 10^{-2}$	775.36	1



**Figure 5.** Convergence of the authalic energy for the benchmark mesh models, with the RGD method for minimizing the normalized stretch energy  $E$ . Line-search strategy: `fminbnd`.



**Figure 6.** Convergence of the authalic energy for the benchmark mesh models, with the RGD method for minimizing the normalized stretch energy  $E$ . Line-search strategy: quadratic/cubic approximation from [11, §6.3.2].

A comparison of the computational times between the two line-search strategies shows that the line-search strategy with a quadratic/cubic interpolant (Table 2) is much more efficient than the line-search strategy that uses MATLAB's `fminbnd` (Table 1). In many cases, the former even yields more accurate results than the latter. This is particularly evident for the Art Statuette (1 111.67 s versus 654.57 s) and the Bimba Statue (2 320.19 s versus 775.36 s) mesh models. The numerical results show that, in general, RGD can still decrease energy as the number of iterations increases.

Table 3 presents the results using a different stopping criterion instead of a fixed maximum number of iterations. Specifically, we keep track of the following quantity

$$\Delta E_A^{(k)} := \frac{E_A(f^{(k-1)}) - E_A(f^{(k)})}{E_A(f^{(k-1)})},$$

which represents the relative improvement in the authalic energy  $E_A(f)$  between two successive iterates, and we stop the RGD method if the value of  $\Delta E_A^{(k)}$  is smaller than  $10^{-3}$ .

It is apparent from Table 3 that RGD stops at different numbers of iterations for different mesh models. Comparing with Table 2 above, it suggests that the improvement after doing more iterations is not significant for a few mesh models, especially Lion Statue, Art Statuette, and Bimba Statue. Table 3 also shows the values of the monitored quantities SD/Mean,  $E_A(f)$ , and #Fs before and after the bijectivity correction described in subsection 5.2. In all cases where the bijectivity correction is applied, the number of folding triangles #Fs decreases or reduces to zero. Moreover, the post-processing for bijectivity correction improves the SD/Mean value in all the cases. For the Right Hand mesh model, the SD/Mean value even improves significantly from 0.2063 to 0.1071 before and after the bijectivity correction. In one case (Gargoyle), the value of authalic energy  $E_A(f)$  also improves.

**Table 3.** RGD for minimizing the normalized stretch energy  $E$ . Line-search strategy: quadratic/cubic approximation from [11, §6.3.2]. Stopping criterion:  $\Delta E_A^{(k)} \leq 10^{-3}$ .

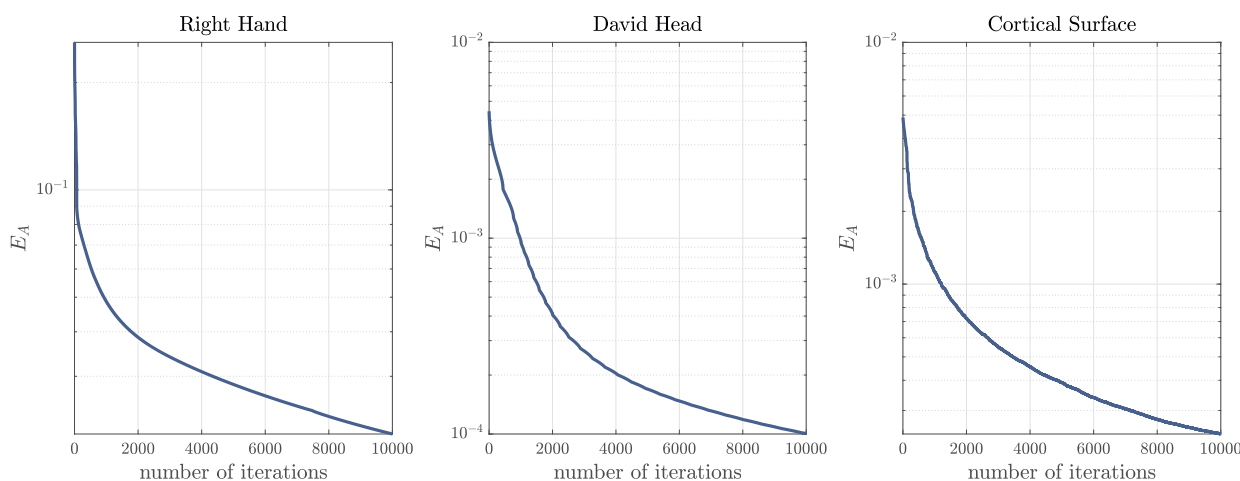
Model Name	Before bijectivity correction			After bijectivity correction			Time	#Its
	SD/Mean	$E_A(f)$	#Fs	SD/Mean	$E_A(f)$	#Fs		
Right Hand	0.2063	$8.01 \times 10^{-2}$	3	0.1071	$8.12 \times 10^{-2}$	0	6.67	127
David Head	0.0123	$1.87 \times 10^{-3}$	0	—————	—————	—————	83.66	432
Cortical Surface	0.0187	$4.23 \times 10^{-3}$	0	—————	—————	—————	9.22	52
Bull	0.1520	$2.37 \times 10^{-1}$	11	0.1403	$2.40 \times 10^{-1}$	3	12.20	50
Bulldog	0.0351	$1.31 \times 10^{-2}$	0	—————	—————	—————	72.64	60
Lion Statue	0.1940	$4.79 \times 10^{-1}$	2	0.1938	$4.79 \times 10^{-1}$	0	2.63	2
Gargoyle	0.0704	$5.17 \times 10^{-2}$	3	0.0682	$5.14 \times 10^{-2}$	0	26.71	17
Max Planck	0.0464	$3.54 \times 10^{-2}$	0	—————	—————	—————	10.58	8
Bunny	0.0417	$2.17 \times 10^{-2}$	0	—————	—————	—————	20.77	13
Chess King	0.0715	$5.17 \times 10^{-2}$	43	0.0641	$5.38 \times 10^{-2}$	17	370.68	107
Art Statuette	0.0409	$2.14 \times 10^{-2}$	0	—————	—————	—————	31.53	3
Bimba Statue	0.0515	$3.32 \times 10^{-2}$	2	0.0514	$3.32 \times 10^{-2}$	0	19.24	2

Figure 7 shows the convergence behavior for the first three smallest mesh models considered,

namely Right Hand, David Head, and Cortical Surface, when the algorithm is run for many more iterations; here, 10 000 iterations. It shows that RGD keeps decreasing the authalic energy  $E_A$ , albeit very slowly. The values of SD/Mean are also improved for all three mesh models. The corresponding numerical results are reported in Table 4; results for 100 iterations are also reported for easier comparison.

**Table 4.** RGD for minimizing the normalized stretch energy  $E$ . Line-search strategy: quadratic/cubic approximation from [11, §6.3.2]. Results for 10 000 iterations for the three smallest mesh models considered.

Model Name	100 Iterations				10 000 Iterations			
	SD/Mean	$E_A(f)$	Time	#Fs	SD/Mean	$E_A(f)$	Time	#Fs
Right Hand	0.1204	$9.40 \times 10^{-2}$	4.07	1	0.0545	$2.07 \times 10^{-2}$	431.28	0
David Head	0.0156	$3.04 \times 10^{-3}$	9.16	0	0.0029	$1.01 \times 10^{-4}$	1 018.87	0
Cortical Surface	0.0200	$3.72 \times 10^{-3}$	16.01	0	0.0045	$2.06 \times 10^{-4}$	1 328.56	0



**Figure 7.** Convergence of the authalic energy for the three smallest benchmark mesh models, with the RGD method for minimizing the normalized stretch energy  $E$ . Line-search strategy: quadratic/cubic approximation from [11, §6.3.2]. 10 000 iterations.

We compute with MATLAB the eigenvalues of the Hessian matrix at the minimizer. Table 5 reports on the smallest eigenvalue of the Hessian of the initial mapping (produced by the FPI method) and eigenvalues of the Hessian of the mapping when the prescribed maximum number of RGD iterations is achieved. The eigenvalues of the Hessian are computed by the MATLAB built-in function `eigs` with the option `smallestabs` and the number of eigenvalues to compute being 2 000. We observe that the Hessian eigenvalues are significantly closer to zero after running the RGD method, as we expected.



**Table 5.** The smallest eigenvalue of the Hessian of the initial mapping given by FPI and after 10 000 iterations of RGD.

Model Name	Smallest eigenvalue of the Hessian	
	After FPI	After RGD
Right Hand	$-2.4429 \times 10^0$	$-1.9013 \times 10^{-1}$
David Head	$-1.8481 \times 10^0$	$-2.8445 \times 10^{-6}$
Cortical Surface	$-4.4940 \times 10^{-1}$	$-3.6768 \times 10^{-3}$

Figure 8 displays the spherical mappings resulting from applying our RGD algorithm to the benchmark mesh models considered, offering a qualitative insight into the goodness of the mappings obtained.



**Figure 8.** The resulting spherical mappings after running RGD.

## 5.2. Correction for bijectivity

The proposed RGD method does not guarantee the produced mapping is bijective. To remedy this drawback, we introduce a post-processing method to ensure bijectivity in the mapping. This is achieved by employing a modified version of the mean value coordinates, as described in [13].

Suppose we have a spherical mapping  $f: \mathcal{M} \rightarrow S^2$  that may not be bijective. First, we map the spherical image to the extended complex plane  $\overline{\mathbb{C}} = \mathbb{C} \cup \{\infty\}$  by the stereographic projection

$$\Pi_{S^2}(x, y, z) = \frac{x}{1-z} + i \frac{y}{1-z}. \quad (5.1)$$

Denote the mapping  $h = \Pi_{S^2} \circ f$  and the complex-valued vector  $\mathbf{h}_i = h(v_i)$ , for  $v_i \in \mathcal{V}(\mathcal{M})$ . The folding triangular faces in the southern hemisphere are now mapped in  $\mathbb{D} \subset \overline{\mathbb{C}}$ , which can be unfolded by solving the linear systems

$$[L_M(h)]_{\mathbf{I}, \mathbf{I}} \widetilde{\mathbf{h}}_{\mathbf{I}} = -[L_M(h)]_{\mathbf{I}, \mathbf{B}} \mathbf{h}_{\mathbf{B}}, \quad (5.2)$$

where  $\mathbf{I} = \{i \mid |h(v_i)| < r\}$  denotes the vertex index set with  $r$  being a value slightly larger than 1, e.g.,  $r = 1.2$ ,  $\mathbf{B} = \{1, \dots, n\} \setminus \mathbf{I}$ , and  $L_M$  is the Laplacian matrix defined as

$$[L_M(h)]_{i,j} = \begin{cases} -\sum_{[v_i, v_j, v_k] \in \mathcal{F}(\mathcal{M})} [\omega_M(h)]_{i,j,k} & \text{if } [v_i, v_j] \in \mathcal{E}(\mathcal{M}), \\ -\sum_{\ell \neq i} [L_M(h)]_{i,\ell} & \text{if } j = i, \\ 0 & \text{otherwise,} \end{cases} \quad (5.3a)$$

with  $\omega_M(h)$  being a variant of the mean value weight [13] defined as

$$[\omega_M(h)]_{i,j,k} = \frac{1}{\|\mathbf{h}_i - \mathbf{h}_j\|} \tan \frac{\varphi_{i,j}^k(h)}{2}, \quad (5.3b)$$

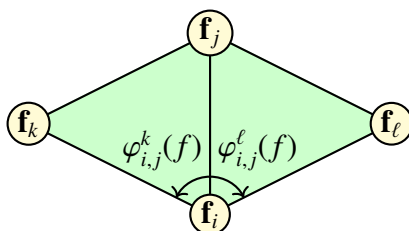
in which  $\varphi_{i,j}^k(h)$  is the angle opposite to the edge  $[h(v_j), h(v_k)]$  at the point  $h(v_i)$  on  $h(\mathcal{M})$ , as illustrated in Figure 9. Then, the mapping is updated by replacing  $\mathbf{h}_{\mathbf{I}}$  with  $\widetilde{\mathbf{h}}_{\mathbf{I}}$ . Next, an inversion

$$\text{Inv}(z) = \frac{1}{\bar{z}} \quad (5.4)$$

is performed to reverse the positions of the southern and northern hemispheres. Then, the linear system (5.2) is solved again to unfold the triangular faces originally located in the northern hemisphere. We denote the updated mapping as  $\widetilde{h}$ . Ultimately, the corrected mapping is given by  $\Pi_{S^2}^{-1} \circ \widetilde{h}$ , where  $\Pi_{S^2}^{-1}$  denotes the inverse stereographic projection

$$\Pi_{S^2}^{-1}(u + iv) = \left( \frac{2u}{u^2 + v^2 + 1}, \frac{2v}{u^2 + v^2 + 1}, \frac{u^2 + v^2 - 1}{u^2 + v^2 + 1} \right). \quad (5.5)$$

In our numerical experiments, we perform the bijectivity correction both after the FPI method and after the RGD method, if needed.



**Figure 9.** An illustration for the mean value weight [13].

### 5.3. Comparison with the FPI method

To validate the algorithm, we compare it with the numerical results on the same models obtained using the FPI method. This method was proposed by Yueh et al. [21] to calculate area-preserving boundary mapping while parameterizing 3-manifolds.

The FPI method of stretch energy minimization (SEM) is performed as follows. First, a spherical harmonic mapping  $f^{(0)}: \mathcal{M} \rightarrow S^2$  is computed by the Laplacian linearization method [4]. Suppose a spherical mapping  $f^{(k)}$  is computed. Then, we map the spherical image to the extended complex plane by the stereographic projection (5.1) together with an inversion (5.4). Then, we define the mapping  $h^{(k)}: \mathcal{M} \rightarrow \mathbb{C} \cup \{\infty\}$  as

$$h^{(k)}(v) = \text{Inv} \circ \Pi_{S^2} \circ f^{(k)}(v).$$

The mapping  $h^{(k)}$  is represented as the complex-valued vector by  $\mathbf{h}_i^{(k)} = h^{(k)}(v_i)$ . Next, we define the interior vertex index set as

$$\mathbf{I}^{(k)} = \{i \mid |h^{(k)}(v_i)| < r\},$$

which collects the indices of vertices in the circle centered at the origin of radius  $r$ . Other vertices are defined as boundary vertices and the associated index set is defined as

$$\mathbf{B}^{(k)} = \{1, \dots, n\} \setminus \mathbf{I}^{(k)}.$$

Then, the interior mapping is updated by solving the linear system

$$[L_S(f^{(k)})]_{\mathbf{I}^{(k)}, \mathbf{I}^{(k)}} \mathbf{h}_{\mathbf{I}^{(k)}}^{(k+1)} = -[L_S(f^{(k)})]_{\mathbf{I}^{(k)}, \mathbf{B}^{(k)}} \mathbf{h}_{\mathbf{B}^{(k)}}^{(k)},$$

while the boundary mapping remains the same, i.e.,  $\mathbf{h}_{\mathbf{B}^{(k+1)}}^{(k)} = \mathbf{h}_{\mathbf{B}^{(k)}}^{(k)}$ . The updated spherical mapping  $f^{(k+1)}: \mathcal{M} \rightarrow S^2$  is computed by

$$f^{(k+1)}(v_i) = \Pi_{S^2}^{-1}(\mathbf{h}_i^{(k)}),$$

where  $\Pi_{S^2}^{-1}$  is the inverse stereographic projection (5.5). This procedure is summarized by Algorithm 2.

Table 6 reports the numerical results for the FPI method applied to the twelve benchmark mesh models considered. Two different stopping criteria are considered: increase in authalic energy (columns 2 to 6) and maximum number of iterations (columns 7 to 10). From Table 6, it appears that performing more iterations of the FPI method does not necessarily improve the mapping  $f$ . In most cases, except for David Head and Cortical Surface, the authalic energy and the values of SD/Mean increase. This motivates us to use the FPI method only to calculate a good initial mapping and then switch to RGD.

**Algorithm 2:** FPI method of the SEM [21].

- 
- 1 Given a genus-zero closed mesh  $\mathcal{M}$ , a tolerance  $\varepsilon$ , a radius  $r$  (e.g.,  $\varepsilon = 10^{-6}$ ,  $r = 1.2$ );  
**Result:** A spherical area-preserving parameterization  $\mathbf{f}$ .
  - 2 Compute a spherical conformal map  $\mathbf{g}$  using the Laplacian linearization method [4];
  - 3 Perform the stereographic projection  $\mathbf{h}_\ell = \Pi_{S^2}(\mathbf{g}_\ell)$ ,  $\ell = 1, \dots, n$ , as in (5.1);
  - 4 Let  $\delta = \infty$ ;
  - 5 **while**  $\delta > \varepsilon$  **do**
  - 6     Update the matrix  $L \leftarrow L_S(f)$ , where  $L_S(f)$  is defined as in (2.2);
  - 7     Perform the inversion  $\mathbf{h}_\ell \leftarrow \text{Inv}(\mathbf{h}_\ell)$ ,  $\ell = 1, \dots, n$ , as in (5.4);
  - 8     Update the index sets  $\mathbf{I} = \{i \mid \|\mathbf{h}_i\| < r\}$  and  $\mathbf{B} = \{1, \dots, n\} \setminus \mathbf{I}$ ;
  - 9     Update  $\mathbf{h}$  by solving the linear system  $L_{\mathbf{I}, \mathbf{I}} \mathbf{h}_{\mathbf{I}} = -L_{\mathbf{I}, \mathbf{B}} \mathbf{h}_{\mathbf{B}}$ ;
  - 10    Update  $\mathbf{h} \leftarrow \mathbf{h} / \text{median}_\ell \|\mathbf{h}_\ell\|$ ;
  - 11    Let  $\mathbf{f}_\ell \leftarrow \Pi_{S^2}^{-1}(\mathbf{h}_\ell)$  as in (5.5),  $\ell = 1, \dots, n$ ;
  - 12    Update  $\delta \leftarrow E_S(\mathbf{g}) - E_S(\mathbf{f})$ ;
  - 13    Update  $\mathbf{g} \leftarrow \mathbf{f}$ .
  - 14 **end while**
- 

**Table 6.** FPI method for minimizing the authalic energy  $E_A$ , using two different stopping criteria. #Its denotes the number of iterations at which the energy started to increase.

Model Name	Energy $E_A$ Increased					100 Iterations			
	SD/Mean	$E_A(f)$	Time	#Fs	#Its	SD/Mean	$E_A(f)$	Time	#Fs
Right Hand	0.2050	$2.86 \times 10^{-1}$	0.08	12	6	0.4598	$2.92 \times 10^0$	1.35	67
David Head	0.0191	$4.66 \times 10^{-3}$	0.35	0	8	0.0169	$3.58 \times 10^{-3}$	4.30	0
Cortical Surface	0.0220	$4.93 \times 10^{-3}$	0.85	0	15	0.0174	$3.21 \times 10^{-3}$	5.62	0
Bull	0.1504	$2.74 \times 10^{-1}$	1.29	8	18	0.1876	$4.59 \times 10^{-1}$	6.90	40
Bulldog	0.0381	$1.49 \times 10^{-2}$	2.61	0	10	0.1833	$3.99 \times 10^{-1}$	22.22	53
Lion Statue	0.1940	$5.10 \times 10^{-1}$	1.12	1	4	0.2064	$5.28 \times 10^{-1}$	23.67	38
Gargoyle	0.0704	$5.47 \times 10^{-2}$	2.64	0	11	4.1020	$4.85 \times 10^2$	36.10	1955
Max Planck	0.0544	$3.67 \times 10^{-2}$	1.35	0	5	0.1844	$1.67 \times 10^1$	25.99	144
Bunny	0.0423	$2.24 \times 10^{-2}$	6.40	0	20	0.0394	$3.96 \times 10^{-2}$	35.78	2
Chess King	0.0713	$6.91 \times 10^{-2}$	6.35	9	8	1.0903	$1.79 \times 10^1$	88.04	1655
Art Statuette	0.0411	$2.15 \times 10^{-2}$	23.27	0	7	0.0908	$1.07 \times 10^{-1}$	342.95	126
Bimba Statue	0.0515	$3.32 \times 10^{-2}$	29.94	6	9	0.0932	$7.42 \times 10^{-2}$	305.00	144

#### 5.4. Comparison with the adaptive area-preserving parameterization

In this section, we compare the numerical results of our RGD method with the adaptive area-preserving parameterization for genus-zero closed surfaces proposed by Choi et al. [7]. The computational procedure is summarized as follows. First, the mesh is punctured by removing two triangular faces  $\tau_1$  and  $\tau_2$  that share a common edge. Then, the FPI of the SEM [22] is applied to compute an area-preserving initial mapping  $g_0: \mathcal{M} \setminus \{\tau_1, \tau_2\} \rightarrow \mathbb{D} := \{\mathbf{x} \in \mathbb{R}^2 \mid \|\mathbf{x}\|_2 \leq 1\}$ . The Beltrami coefficient of the mapping  $g_0$  is denoted by  $\mu_{g_0}$ . Next, a quasi-conformal map  $g: \mathcal{M} \setminus \{\tau_1, \tau_2\} \rightarrow \mathbb{D}$  with

$\|\mu_g\|_\infty = \|\lambda\mu_{g_0}\|_\infty < 1$  is constructed [9]. The scaling factor  $\lambda \in [0, 1]$  is chosen to be 0.2 in practice. Then, the optimal mass transport mapping  $h_r: \mathcal{M} \setminus \{\tau_1, \tau_2\} \rightarrow r\mathbb{D}$  is computed by the method proposed by Zhao et al. [23], with the optimal radius  $r$  satisfying

$$r = \operatorname{argmin}_r \int |\mu_{(h_r \circ rg)^{-1}(z)}|^2 dz.$$

The final spherical area-preserving parameterization is obtained by the composition mapping  $f = \Pi_{S^2}^{-1} \circ h_r \circ rg$ , where  $\Pi_{S^2}^{-1}$  denotes the inverse stereographic projection (5.5).

Table 7 reports on the numerical results. The algorithm of Choi et al. [7] was run with the default parameters found in the code package\*. The only satisfactory results are those for the David Head mesh model. However, even in this case, the values of the monitored quantities (SD/Mean,  $E_A(f)$ , and computation time) do not compete with those obtained by running our RGD method for only ten iterations; compare with the fourth column of Table 2. Our method is much more efficient and accurate and provides mappings with better bijectivity properties.

**Table 7.** Adaptive area-preserving parameterization for genus-zero closed surfaces proposed by Choi et al. [7] applied to the twelve mesh models considered in this work. Stopping criterion: default values found in the code package.

Model Name	SD/Mean	$E_A(f)$	Time	#Foldings
Right Hand	18.3283	$4.84 \times 10^3$	218.03	672
David Head	0.0426	$2.27 \times 10^{-2}$	298.76	0
Cortical Surface	0.6320	$1.14 \times 10^0$	420.20	10
Bull	8.5565	$1.82 \times 10^3$	34.42	335
Bulldog	9.2379	$1.22 \times 10^3$	183.94	338
Lion Statue	0.2626	$8.96 \times 10^{-1}$	1498.91	540
Gargoyle	0.3558	$1.30 \times 10^0$	1483.35	571
Max Planck	11.6875	$1.49 \times 10^3$	195.39	575
Bunny	27.6014	$8.94 \times 10^3$	157.87	208
Chess King	11.8300	$1.65 \times 10^3$	608.55	948
Art Statuette	394.4414	$9.93 \times 10^0$	2284.79	2242
Bimba Statue	0.5110	$2.01 \times 10^0$	16 773.34	11 821

### 5.5. Comparison with the spherical optimal transportation mapping

In this section, we compare the numerical results of our RGD method with the spherical optimal transportation mapping proposed by Cui et al. [10]. The computational procedure can be summarized as follows. First, the algorithm calculates a conformal map from a genus-zero closed surface  $\mathcal{M}$  to the unit sphere and then normalizes the area of  $\mathcal{M}$ . Iteratively, a power diagram is computed from the sphere and the power radii. The gradient of the energy is calculated from the vertices, followed by the Hessian matrix, which depends on the source measure. A linear equation is solved at every

\*Available at <https://www.math.cuhk.edu.hk/~ptchoi/files/adaptiveareapreservingmap.zip>.

step to update the power radii, and a line search strategy is used. The step length parameter is chosen so that all cells of the power diagram are non-degenerated. If some cells are degenerated, the step length is reduced by half, and the power diagram is computed again until all cells are non-degenerated. After the loop, the centroid of each power cell in the power diagram is computed, and the mapping from each vertex to each centroid of the power cell is returned. The algorithmic details can be found in [10, Algorithm 1].

Here, we run the algorithm of [10] with the default parameters found in the code package<sup>†</sup>. The target mesh for the mesh model was adapted to be its spherical harmonic mapping as suggested in [10].

From Table 8, it appears that the method of [10] is capable, at least when working, of computing bijective spherical mappings within a relatively moderate number of iterations. However, the area-preserving properties, as measured by SD/Mean and  $E_A(f)$ , are inferior to those obtained with our RGD method for all four mesh models considered.

**Table 8.** Spherical optimal transportation mapping proposed by Cui et al. [10] applied to the twelve mesh models considered in this work. The executable fails to output a mapping for eight mesh models among the twelve, which are not shown in the table.

Model Name	SD/Mean	$E_A(f)$	#Foldings	# Iterations
David Head	0.4189	$2.25 \times 10^0$	0	27
Cortical Surface	0.5113	$3.11 \times 10^0$	0	27
Bulldog	0.8665	$1.00 \times 10^1$	0	33
Max Planck	0.5619	$4.38 \times 10^0$	0	25

### 5.6. Numerical stability

In this section, we investigate the numerical stability of our scheme. To this aim, we introduce Gaussian random noise to the vertex normal of each vertex in every mesh model according to a given value of noise variance  $\sigma_{\text{noise}}$ . We then re-run the entire algorithm, i.e., we first perform a few iterations of the FPI method (Algorithm 2) to obtain a mapping that is used as an initial mapping for the RGD method (Algorithm 1). We then calculate a relative error on the authalic energy, as defined below in (5.6). We repeat this procedure for different values of noise variance  $\sigma_{\text{noise}}$ .

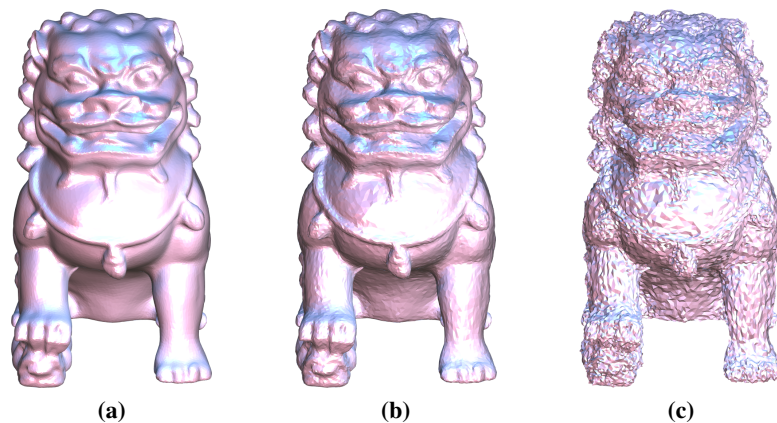
Figure 10 shows the original mesh model of the Lion Statue (panel (a)) and two noisy versions (panels (b) and (c)).

We compute the following relative error on the authalic energy

$$\text{err-}E_A(f, \tilde{f}) := \#\text{Vertices} \times \frac{|E_A(\tilde{f}) - E_A(f)|}{\sum_{v \in \mathcal{V}(\mathcal{M})} \|\tilde{v} - v\|_2}, \quad (5.6)$$

where  $E_A(\tilde{f})$  and  $E_A(f)$  are the authalic energies after 100 iterations of RGD for the mesh model with and without noise, respectively, and  $\tilde{v}$  and  $v$  denote the coordinates of the vertices of the mesh model with and without noise, respectively.

<sup>†</sup>Available at <https://www3.cs.stonybrook.edu/~gu/software/SOT/index.html>



**Figure 10.** The “Lion Statue” mesh model: (a) without noise; (b), (c) with noise variance  $\sigma_{\text{noise}} = 1 \times 10^{-3}$  and  $5 \times 10^{-3}$ , respectively.

Tables 9 and 10 report the numerical results for all the noisy mesh models, with  $\sigma_{\text{noise}} = 1 \times 10^{-3}$  and  $5 \times 10^{-3}$ , respectively. The values of authentic energy for the non-noisy mesh models from Table 2 are reported in the second last column for easier comparison. We observe that the values for SD/Mean and  $E_A(\tilde{f})$  remain bounded and reasonable with respect to the original mesh models, demonstrating that our method is stable to noise.

**Table 9.** Numerical stability study: 100 iterations of RGD with  $\sigma_{\text{noise}} = 1 \times 10^{-3}$ . Line-search strategy: quadratic/cubic approximation from [11, §6.3.2]. The quantity  $\text{err}-E_A$  is defined in (5.6).

Model Name	$\sigma_{\text{noise}} = 1 \times 10^{-3}$					$E_A(f)$	$\text{err}-E_A(f, \tilde{f})$
	SD/Mean	$E_A(\tilde{f})$	Time	#Fs	#Fs after b.c.		
Right Hand	0.1254	$1.03 \times 10^{-1}$	3.13	14	1	$9.40 \times 10^{-2}$	$5.84 \times 10^{-2}$
David Head	0.0108	$1.44 \times 10^{-3}$	9.98	0	0	$3.04 \times 10^{-3}$	$4.47 \times 10^{-3}$
Cortical Surface	0.0187	$3.82 \times 10^{-3}$	12.78	0	0	$3.72 \times 10^{-3}$	$3.45 \times 10^{-3}$
Bull	0.2528	$9.23 \times 10^{-1}$	16.74	19	6	$2.19 \times 10^{-1}$	$1.12 \times 10^0$
Bulldog	0.0273	$6.94 \times 10^{-3}$	59.26	0	0	$1.27 \times 10^{-2}$	$7.19 \times 10^{-6}$
Lion Statue	0.1630	$3.27 \times 10^{-1}$	66.78	1	0	$4.54 \times 10^{-1}$	$1.36 \times 10^{-4}$
Gargoyle	0.1677	$3.66 \times 10^{-1}$	65.72	0	0	$4.76 \times 10^{-2}$	$4.45 \times 10^{-3}$
Max Planck	0.0464	$2.67 \times 10^{-2}$	64.30	0	0	$3.39 \times 10^{-2}$	$5.27 \times 10^{-5}$
Bunny	0.0297	$1.11 \times 10^{-2}$	81.31	0	0	$1.91 \times 10^{-2}$	$5.32 \times 10^{-3}$
Chess King	0.0747	$6.08 \times 10^{-2}$	193.04	119	38	$5.23 \times 10^{-2}$	$1.03 \times 10^{-3}$
Art Statuette	0.0235	$6.88 \times 10^{-3}$	636.30	0	0	$2.10 \times 10^{-2}$	$2.54 \times 10^{-2}$
Bimba Statue	0.0541	$3.33 \times 10^{-2}$	759.36	2	0	$3.29 \times 10^{-2}$	$5.81 \times 10^{-4}$

**Table 10.** Numerical stability study: 100 iterations of RGD with  $\sigma_{\text{noise}} = 5 \times 10^{-3}$ . Line-search strategy: quadratic/cubic approximation from [11, §6.3.2]. The quantity  $\text{err-}E_A$  is defined in (5.6).

Model Name	$\sigma_{\text{noise}} = 5 \times 10^{-3}$					$E_A(f)$	$\text{err-}E_A(f, \tilde{f})$
	SD/Mean	$E_A(\tilde{f})$	Time	#Fs	#Fs after b.c.		
Right Hand	0.2304	$5.77 \times 10^{-1}$	3.12	10	4	$9.40 \times 10^{-2}$	$1.45 \times 10^0$
David Head	0.0211	$5.51 \times 10^{-3}$	9.43	0	0	$3.04 \times 10^{-3}$	$6.84 \times 10^{-3}$
Cortical Surface	0.0301	$1.02 \times 10^{-2}$	12.72	0	0	$3.72 \times 10^{-3}$	$2.28 \times 10^{-1}$
Bull	0.1723	$2.98 \times 10^{-1}$	17.40	18	4	$2.19 \times 10^{-1}$	$1.19 \times 10^{-1}$
Bulldog	0.0629	$3.86 \times 10^{-2}$	61.75	2	0	$1.27 \times 10^{-2}$	$3.23 \times 10^{-5}$
Lion Statue	0.1044	$1.36 \times 10^{-1}$	69.55	0	0	$4.54 \times 10^{-1}$	$3.36 \times 10^{-4}$
Gargoyle	0.0974	$9.68 \times 10^{-2}$	69.32	1	0	$4.76 \times 10^{-2}$	$6.87 \times 10^{-4}$
Max Planck	0.0782	$7.11 \times 10^{-2}$	67.30	0	0	$3.39 \times 10^{-2}$	$2.75 \times 10^{-4}$
Bunny	0.0455	$2.57 \times 10^{-2}$	80.93	0	0	$1.91 \times 10^{-2}$	$4.40 \times 10^{-3}$
Chess King	0.1521	$2.72 \times 10^{-1}$	187.57	95	35	$5.23 \times 10^{-2}$	$2.64 \times 10^{-2}$
Art Statuette	0.0981	$5.41 \times 10^{-2}$	617.75	0	0	$2.10 \times 10^{-2}$	$5.92 \times 10^{-2}$
Bimba Statue	0.1083	$1.25 \times 10^{-1}$	743.00	76	0	$3.29 \times 10^{-2}$	$1.68 \times 10^{-1}$

### 5.7. Registration problem between two brain surfaces

A registration mapping between surfaces  $\mathcal{M}_0$  and  $\mathcal{M}_1$  refers to a bijective mapping  $g: \mathcal{M}_0 \rightarrow \mathcal{M}_1$ . An ideal registration mapping keeps important landmarks aligned while preserving specified geometry properties. In this section, we demonstrate a framework for the computation of landmark-aligned area-preserving parameterizations of genus-zero closed surfaces.

Suppose a set of landmark pairs  $\{(p_i, q_i) \mid p_i \in \mathcal{M}_0, q_i \in \mathcal{M}_1\}_{i=1}^m$  is given. The goal is to compute an area-preserving simplicial mapping  $g: \mathcal{M}_0 \rightarrow \mathcal{M}_1$  that satisfies  $g(p_i) \approx q_i$ , for  $i = 1, \dots, m$ . First, we compute area-preserving parameterizations  $f_0: \mathcal{M}_0 \rightarrow S^2$  and  $f_1: \mathcal{M}_1 \rightarrow S^2$  of surfaces  $\mathcal{M}_0$  and  $\mathcal{M}_1$ , respectively. The simplicial registration mapping  $h: S^2 \rightarrow S^2$  that satisfies  $h \circ f_0(p_i) = f_1(q_i)$ , for  $i = 1, \dots, m$ , can be carried out by minimizing the registration energy

$$E_R(h) = E_S(h) + \sum_{i=1}^m \lambda_i \|h \circ f_0(p_i) - f_1(q_i)\|^2.$$

Let

$$\mathbf{h} = \begin{bmatrix} (h \circ f_0(v_1))^T \\ \vdots \\ (h \circ f_0(v_n))^T \end{bmatrix} = [\mathbf{h}^1 \quad \mathbf{h}^2 \quad \mathbf{h}^3]$$

be the matrix representation of  $h$ . The gradient of  $E_R$  with respect to  $\mathbf{h}$  can be formulated as

$$\nabla E_R(h) = 2(I_3 \otimes L_S(h)) \text{vec}(\mathbf{h}) + \text{vec}(\mathbf{r}),$$



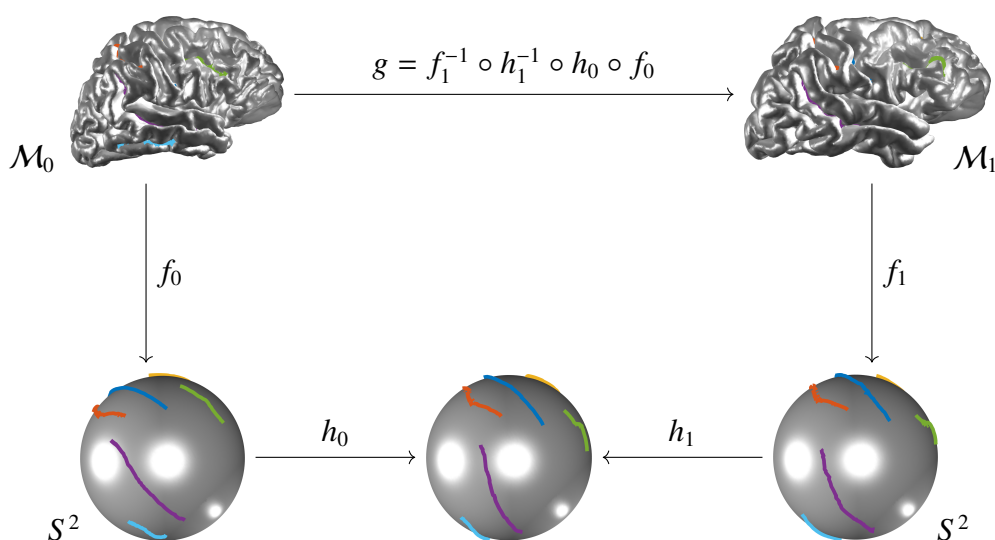
where  $\mathbf{r}$  is the matrix of the same size as  $\mathbf{h}$  given by

$$\mathbf{r}(i, :) = \begin{cases} 2\lambda_i (\mathbf{h}(i, :) - (f_1(q_i))^\top) & \text{if } p_i \text{ is a landmark,} \\ (0, 0, 0) & \text{otherwise.} \end{cases}$$

In practice, we define the midpoints  $c_i$  of each landmark pairs on  $S^2$  as

$$c_i = \frac{1}{2}(f_0(p_i) + f_1(q_i)),$$

for  $i = 1, \dots, m$ , and compute  $h_0$  and  $h_1$  on  $S^2$  that satisfy  $h_0 \circ f_0(p_i) = c_i$  and  $h_1 \circ f_1(q_i) = c_i$ , respectively. Ultimately, the registration mapping  $g: \mathcal{M}_0 \rightarrow \mathcal{M}_1$  is obtained by the composition mapping  $g = f_1^{-1} \circ h_1^{-1} \circ h_0 \circ f_0$ . Figure 11 schematizes this composition of functions for the landmark-aligned morphing process from one brain to another.

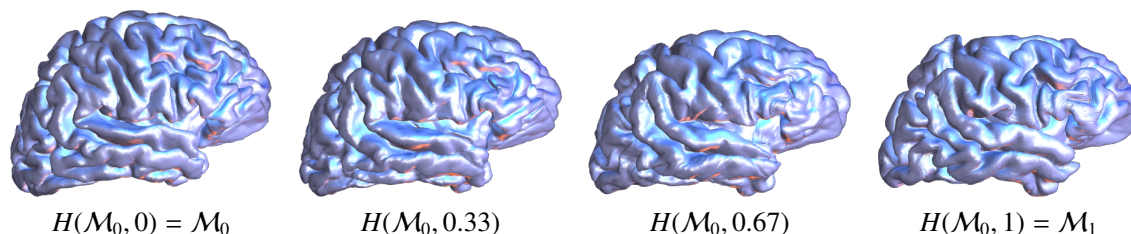


**Figure 11.** Scheme for the landmark-aligned surface registration application described in subsection 5.7.

A landmark-aligned morphing process from  $\mathcal{M}_0$  to  $\mathcal{M}_1$  can be constructed by the linear homotopy  $H: \mathcal{M}_0 \times [0, 1] \rightarrow \mathbb{R}^3$  defined as

$$H(v, t) = (1 - t)v + tg(v). \quad (5.7)$$

In Figure 12, we demonstrate the morphing process from one brain to another brain by four snapshots at four different values of  $t$ . The brain surfaces are obtained from the source code package in [9].



**Figure 12.** The images of the linear homotopy (5.7) from one brain to another brain.

## 6. Conclusions and outlook

In this paper, we introduced an RGD method for computing spherical area-preserving mappings of genus-zero closed surfaces. Our approach combines the tools of Riemannian optimization and computational geometry to develop a method based on the minimization of the normalized stretch energy. The proposed algorithm has theoretically guaranteed convergence and is accurate, efficient, and robust. We tested two different line-search strategies and conducted extensive numerical experiments on various mesh models to demonstrate the algorithm's stability and effectiveness. By comparing with three existing methods for computing area-preserving mappings, we demonstrated that our algorithm is more efficient than these state-of-the-art methods. Moreover, we show that our approach is stable and robust even when the mesh model undergoes small perturbations. Finally, we applied our algorithm to the practical problem of landmark-aligned surface registration between two human brain models.

There are some directions in which we could conduct further research. Specifically, we would like to enhance the speed of convergence of the algorithm we have proposed while keeping the computational cost low. One potential way to improve this would be to use appropriate Riemannian generalizations of the conjugate gradient method or the limited memory BFGS (L-BFGS) method, as suggested in [17]. Another research direction may target genus-one or higher genus closed surfaces.

## Appendix A Calculation of the gradient of the image area

The image area of the simplicial mapping  $f$  can be calculated as follows. Let  $\tau = [v_i, v_j, v_k]$  and denote  $f_{ij}^\ell = f_i^\ell - f_j^\ell$ ,  $f_{ik}^\ell = f_i^\ell - f_k^\ell$ , and  $f_{jk}^\ell = f_j^\ell - f_k^\ell$ , for  $\ell = 1, 2, 3$ . The image area of a simplicial mapping  $f$  can be formulated as

$$\mathcal{A}(f) = \sum_{\tau \in \mathcal{F}(\mathcal{M})} |f(\tau)| = \sum_{\tau \in \mathcal{F}(\mathcal{M})} \frac{1}{2} \sqrt{\mathcal{A}_{12}(f|_\tau)^2 + \mathcal{A}_{13}(f|_\tau)^2 + \mathcal{A}_{23}(f|_\tau)^2},$$

where

$$\mathcal{A}_{12}(f|_\tau) = f_{ij}^1 f_{ik}^2 - f_{ij}^2 f_{ik}^1, \quad \mathcal{A}_{13}(f|_\tau) = f_{ij}^1 f_{jk}^3 - f_{ij}^3 f_{jk}^1, \quad \mathcal{A}_{23}(f|_\tau) = f_{ij}^2 f_{jk}^3 - f_{ij}^3 f_{jk}^2.$$

The functionals  $\mathcal{A}_{12}$ ,  $\mathcal{A}_{13}$  and  $\mathcal{A}_{23}$  measure the image area of mappings  $\Pi_{P_{xy}} \circ f$ ,  $\Pi_{P_{xz}} \circ f$  and  $\Pi_{P_{yz}} \circ f$ , respectively. The partial derivatives of  $\mathcal{A}(f|_\tau)$  can be formulated as

$$\begin{aligned} \frac{\partial}{\partial f_i^1} \mathcal{A}(f|_\tau) &= \frac{1}{4\mathcal{A}(f|_\tau)} \left( \mathcal{A}_{12}(f|_\tau) f_{jk}^2 + \mathcal{A}_{13}(f|_\tau) f_{jk}^3 \right), \\ \frac{\partial}{\partial f_i^2} \mathcal{A}(f|_\tau) &= \frac{-1}{4\mathcal{A}(f|_\tau)} \left( \mathcal{A}_{12}(f|_\tau) f_{jk}^1 - \mathcal{A}_{23}(f|_\tau) f_{jk}^3 \right), \\ \frac{\partial}{\partial f_i^3} \mathcal{A}(f|_\tau) &= \frac{-1}{4\mathcal{A}(f|_\tau)} \left( \mathcal{A}_{13}(f|_\tau) f_{jk}^1 - \mathcal{A}_{23}(f|_\tau) f_{jk}^2 \right), \\ \frac{\partial}{\partial f_j^1} \mathcal{A}(f|_\tau) &= \frac{-1}{4\mathcal{A}(f|_\tau)} \left( \mathcal{A}_{12}(f|_\tau) f_{ik}^2 + \mathcal{A}_{13}(f|_\tau) f_{ik}^3 \right), \\ \frac{\partial}{\partial f_j^2} \mathcal{A}(f|_\tau) &= \frac{1}{4\mathcal{A}(f|_\tau)} \left( \mathcal{A}_{12}(f|_\tau) f_{ik}^1 - \mathcal{A}_{23}(f|_\tau) f_{ik}^3 \right), \end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial f_j^3} \mathcal{A}(f|_\tau) &= \frac{1}{4\mathcal{A}(f|_\tau)} \left( \mathcal{A}_{13}(f|_\tau) f_{ik}^1 + \mathcal{A}_{23}(f|_\tau) f_{ik}^2 \right), \\ \frac{\partial}{\partial f_k^1} \mathcal{A}(f|_\tau) &= \frac{1}{4\mathcal{A}(f|_\tau)} \left( \mathcal{A}_{12}(f|_\tau) f_{ij}^2 + \mathcal{A}_{13}(f|_\tau) f_{ij}^3 \right), \\ \frac{\partial}{\partial f_k^2} \mathcal{A}(f|_\tau) &= \frac{-1}{4\mathcal{A}(f|_\tau)} \left( \mathcal{A}_{12}(f|_\tau) f_{ij}^1 - \mathcal{A}_{23}(f|_\tau) f_{ij}^3 \right), \\ \frac{\partial}{\partial f_k^3} \mathcal{A}(f|_\tau) &= \frac{-1}{4\mathcal{A}(f|_\tau)} \left( \mathcal{A}_{13}(f|_\tau) f_{ij}^1 + \mathcal{A}_{23}(f|_\tau) f_{ij}^2 \right).\end{aligned}$$

## Appendix B Calculation of the Hessian of the stretch energy functional

In this appendix, we describe the calculation of the Hessian of the stretch energy functional. Let  $\tau = [v_i, v_j, v_k]$ . From [20, Theorem 3.5 and (3.1)], we know that

$$\nabla_{\mathbf{f}^\ell} E_S(f|_\tau) = 2L_S(f|_\tau) \mathbf{f}|_\tau^\ell, \quad \ell = 1, 2, 3,$$

where

$$L_S(f|_\tau) = \frac{1}{4|\tau|} \begin{bmatrix} (\mathbf{f}_i - \mathbf{f}_k)^\top (\mathbf{f}_j - \mathbf{f}_k) & -(\mathbf{f}_i - \mathbf{f}_k)^\top (\mathbf{f}_j - \mathbf{f}_k) & -(\mathbf{f}_i - \mathbf{f}_j)^\top (\mathbf{f}_k - \mathbf{f}_j) \\ +(\mathbf{f}_i - \mathbf{f}_j)^\top (\mathbf{f}_k - \mathbf{f}_j) & (\mathbf{f}_i - \mathbf{f}_k)^\top (\mathbf{f}_j - \mathbf{f}_k) & -(\mathbf{f}_j - \mathbf{f}_i)^\top (\mathbf{f}_k - \mathbf{f}_i) \\ -(\mathbf{f}_i - \mathbf{f}_k)^\top (\mathbf{f}_j - \mathbf{f}_k) & +(\mathbf{f}_j - \mathbf{f}_i)^\top (\mathbf{f}_k - \mathbf{f}_i) & (\mathbf{f}_i - \mathbf{f}_j)^\top (\mathbf{f}_k - \mathbf{f}_j) \\ -(\mathbf{f}_i - \mathbf{f}_j)^\top (\mathbf{f}_k - \mathbf{f}_j) & -(\mathbf{f}_j - \mathbf{f}_i)^\top (\mathbf{f}_k - \mathbf{f}_i) & +(\mathbf{f}_j - \mathbf{f}_i)^\top (\mathbf{f}_k - \mathbf{f}_i) \end{bmatrix},$$

and  $\mathbf{f}|_\tau^\ell = (\mathbf{f}_i^\ell, \mathbf{f}_j^\ell, \mathbf{f}_k^\ell)^\top$ ,  $\ell = 1, 2, 3$ . A direct calculation yields that the Hessian matrix

$$\text{Hess}(E_S(f|_\tau)) = \begin{bmatrix} \frac{\partial E_S(f|_\tau)}{\partial f_i^3 \partial f_i^1} & \frac{\partial E_S(f|_\tau)}{\partial f_i^3 \partial f_j^1} & \frac{\partial E_S(f|_\tau)}{\partial f_i^3 \partial f_k^1} \\ \frac{\partial E_S(f|_\tau)}{\partial f_j^3 \partial f_i^1} & \frac{\partial E_S(f|_\tau)}{\partial f_j^3 \partial f_j^1} & \frac{\partial E_S(f|_\tau)}{\partial f_j^3 \partial f_k^1} \\ \frac{\partial E_S(f|_\tau)}{\partial f_k^3 \partial f_i^1} & \frac{\partial E_S(f|_\tau)}{\partial f_k^3 \partial f_j^1} & \frac{\partial E_S(f|_\tau)}{\partial f_k^3 \partial f_k^1} \end{bmatrix}_{s,t=1}^3$$

can be formulated as

$$\text{Hess}(E_S(f|_\tau)) = \frac{1}{2|\tau|} \begin{bmatrix} \mathbf{h}_{ijk}^2 \mathbf{h}_{ijk}^{2\top} + \mathbf{h}_{ijk}^3 \mathbf{h}_{ijk}^{3\top} & \mathbf{h}_{ijk}^1 \mathbf{h}_{ijk}^{2\top} - 2\mathbf{h}_{ijk}^2 \mathbf{h}_{ijk}^{1\top} & \mathbf{h}_{ijk}^1 \mathbf{h}_{ijk}^{3\top} - 2\mathbf{h}_{ijk}^3 \mathbf{h}_{ijk}^{1\top} \\ \mathbf{h}_{ijk}^2 \mathbf{h}_{ijk}^{1\top} - 2\mathbf{h}_{ijk}^1 \mathbf{h}_{ijk}^{2\top} & \mathbf{h}_{ijk}^1 \mathbf{h}_{ijk}^{1\top} + \mathbf{h}_{ijk}^3 \mathbf{h}_{ijk}^{3\top} & \mathbf{h}_{ijk}^2 \mathbf{h}_{ijk}^{3\top} - 2\mathbf{h}_{ijk}^3 \mathbf{h}_{ijk}^{2\top} \\ \mathbf{h}_{ijk}^3 \mathbf{h}_{ijk}^{1\top} - 2\mathbf{h}_{ijk}^1 \mathbf{h}_{ijk}^{3\top} & \mathbf{h}_{ijk}^3 \mathbf{h}_{ijk}^{2\top} - 2\mathbf{h}_{ijk}^2 \mathbf{h}_{ijk}^{3\top} & \mathbf{h}_{ijk}^1 \mathbf{h}_{ijk}^{1\top} + \mathbf{h}_{ijk}^2 \mathbf{h}_{ijk}^{2\top} \end{bmatrix},$$

where  $\mathbf{h}_{ijk}^\ell = (f_j^\ell - f_k^\ell, f_k^\ell - f_i^\ell, f_i^\ell - f_j^\ell)^\top$ ,  $\ell = 1, 2, 3$ .

## Appendix C Line-search procedure of the RGD method

In this appendix, we describe the line-search procedure used in our RGD method. We start by detailing how to compute the derivative appearing in the sufficient decrease condition, and then we describe the interpolant line-search strategy from [11, §6.3.2].

In Euclidean space  $\mathbb{R}^n$ , the steepest descent method updates a current iterate by moving in the direction of the anti-gradient, by a step size chosen according to an appropriate line-search rule. The step size is related to the sufficient decrease condition [16]

$$\phi(\alpha_k) \leq \phi(0) + c_1 \alpha_k \phi'(0). \quad (\text{C.1})$$

In the Euclidean setting, the univariate function  $\phi(\alpha)$  in the line-search procedure is

$$\phi(\alpha) = E(\mathbf{f}^{(k)} + \alpha \mathbf{d}^{(k)}),$$

where  $E$  is the objective function considered, and  $\mathbf{f}^{(k)}$  is the current iterate. However, this function changes in the Riemannian optimization framework because we cannot directly perform the vector addition  $\mathbf{f}^{(k)} + \alpha \mathbf{d}^{(k)}$  without leaving the manifold.

Let  $\mathbf{f}^{(k)}$  be a point of  $(S^2)^n$  at the  $k$ th iteration of our RGD algorithm, and  $\mathbf{R}_{\mathbf{f}^{(k)}}$  the retraction at  $\mathbf{f}^{(k)}$ , as defined in subsection 3.2.3. Let the objective function be  $E: (S^2)^n \rightarrow \mathbb{R}$ . For a fixed point  $\mathbf{f}^{(k)}$  and a fixed tangent vector  $\mathbf{d}^{(k)}$ , we introduce the vector-valued function  $\psi: \mathbb{R} \rightarrow (S^2)^n$ , defined by  $\alpha \mapsto \mathbf{R}_{\mathbf{f}^{(k)}}(\alpha \mathbf{d}^{(k)})$ . Hence, by composition of functions, we have  $\phi: \mathbb{R} \rightarrow \mathbb{R}$ , defined by

$$\phi(\alpha) := E(\psi(\alpha)).$$

Note that  $\phi(0) = E(\psi(0)) = E(\mathbf{f}^{(k)}) =: E^{(k)}$ , due to the definition of retraction.

To evaluate the sufficient decrease condition (C.1), we need to calculate  $\phi'(0)$ , and to compute  $\phi'(0)$  we need to calculate the derivative of the retraction  $\mathbf{R}_{\mathbf{f}^{(k)}}(\alpha \mathbf{d}^{(k)})$  with respect to  $\alpha$ . The derivative of  $\phi(\alpha)$  is given by the chain rule

$$\phi'(\alpha) = \nabla E(\psi(\alpha))^\top \psi'(\alpha),$$

and then we evaluate it at  $\alpha = 0$ , i.e.,

$$\phi'(0) = \nabla E(\mathbf{f}^{(k)})^\top \psi'(0).$$

In general, the retraction and the derivative  $\psi'(\alpha)$  depend on the choice of the manifold. The differentials of a retraction provide the so-called *vector transports*, and the derivative of the retraction on the unit sphere appears in [2, §8.1.2].

In the following formulas, we omit the superscript  $(k)$  referring to the current iteration of RGD and assume that the line-search direction  $\mathbf{d}$  is also partitioned as the matrix  $\mathbf{f}$ ; see (2.1). Recalling the retraction on the power manifold  $(S^2)^n$  from (3.3), we can write  $\psi(\alpha)$  as

$$\psi(\alpha) = \mathbf{R}_{\mathbf{f}}(\alpha \mathbf{d}) = \begin{bmatrix} \frac{1}{\|\mathbf{f}_1 + \alpha \mathbf{d}_1\|_2} & & & & \\ & \frac{1}{\|\mathbf{f}_2 + \alpha \mathbf{d}_2\|_2} & & & \\ & & \ddots & & \\ & & & \frac{1}{\|\mathbf{f}_n + \alpha \mathbf{d}_n\|_2} & \\ & & & & \begin{bmatrix} \mathbf{f}_1^\top + \alpha \mathbf{d}_1^\top \\ \mathbf{f}_2^\top + \alpha \mathbf{d}_2^\top \\ \vdots \\ \mathbf{f}_n^\top + \alpha \mathbf{d}_n^\top \end{bmatrix} \end{bmatrix}.$$

The derivative  $\psi'(\alpha)$  can be computed via the formula (cf. [2, Example 8.1.4])

$$\psi'(\alpha) = \begin{bmatrix} -\frac{(\mathbf{f}_1 + \alpha \mathbf{d}_1)^\top \mathbf{d}_1}{\|\mathbf{f}_1 + \alpha \mathbf{d}_1\|_2^3} (\mathbf{f}_1 + \alpha \mathbf{d}_1)^\top \\ -\frac{(\mathbf{f}_2 + \alpha \mathbf{d}_2)^\top \mathbf{d}_2}{\|\mathbf{f}_2 + \alpha \mathbf{d}_2\|_2^3} (\mathbf{f}_2 + \alpha \mathbf{d}_2)^\top \\ \vdots \\ -\frac{(\mathbf{f}_n + \alpha \mathbf{d}_n)^\top \mathbf{d}_n}{\|\mathbf{f}_n + \alpha \mathbf{d}_n\|_2^3} (\mathbf{f}_n + \alpha \mathbf{d}_n)^\top \end{bmatrix} + \begin{bmatrix} \frac{\mathbf{d}_1^\top}{\|\mathbf{f}_1 + \alpha \mathbf{d}_1\|_2} \\ \frac{\mathbf{d}_2^\top}{\|\mathbf{f}_2 + \alpha \mathbf{d}_2\|_2} \\ \vdots \\ \frac{\mathbf{d}_n^\top}{\|\mathbf{f}_n + \alpha \mathbf{d}_n\|_2} \end{bmatrix},$$

At  $\alpha = 0$ , this simplifies into

$$\psi'(0) = \begin{bmatrix} \frac{\mathbf{d}_1^\top}{\|\mathbf{f}_1\|_2} \\ \frac{\mathbf{d}_2^\top}{\|\mathbf{f}_2\|_2} \\ \vdots \\ \frac{\mathbf{d}_n^\top}{\|\mathbf{f}_n\|_2} \end{bmatrix} - \begin{bmatrix} \frac{(\mathbf{f}_1)^\top \mathbf{d}_1}{\|\mathbf{f}_1\|_2^3} \mathbf{f}_1^\top \\ \frac{(\mathbf{f}_2)^\top \mathbf{d}_2}{\|\mathbf{f}_2\|_2^3} \mathbf{f}_2^\top \\ \vdots \\ \frac{(\mathbf{f}_n)^\top \mathbf{d}_n}{\|\mathbf{f}_n\|_2^3} \mathbf{f}_n^\top \end{bmatrix}.$$

This value is needed to evaluate the sufficient decrease condition (C.1).

### Quadratic/cubic approximation

At every step of our RGD algorithm, we want to satisfy the sufficient decrease condition (C.1). Here, we adopt the safeguarded quadratic/cubic approximation strategy described in [11, §6.3.2].

In the following, we let  $\alpha_k$  and  $\alpha_{k-1}$  denote the step lengths used at iterations  $k$  and  $k - 1$  of the optimization algorithm, respectively. We denote the initial guess using  $\alpha_0$ . We suppose that the initial guess is given; alternatively, one can use [16, (3.60)] as initial guess, i.e.,

$$\alpha_0 = \frac{2(E^{(k)} - E^{(k-1)})}{\phi'(0)}.$$

If  $\alpha_0$  satisfies the sufficient decrease condition (C.1), i.e.,

$$\phi(\alpha_0) \leq \phi(0) + c_1 \alpha_0 \phi'(0),$$

then  $\alpha_0$  is accepted as step length, and we terminate the search. Otherwise, we build a quadratic approximation  $\phi_q(\alpha)$  of  $\phi(\alpha)$  using the information we have, that is,  $\phi(0)$ ,  $\phi'(0)$ , and  $\phi(\alpha_0)$ . The quadratic model is

$$\phi_q(\alpha) = [\phi(\alpha_0) - \phi(0) - \phi'(0) \alpha_0] \alpha^2 + \phi'(0) \alpha + \phi(0).$$

The new trial value  $\alpha_1$  is defined as the minimizer of this quadratic, i.e.,

$$\alpha_1 = -\frac{\phi'(0) \alpha_0^2}{2 [\phi(\alpha_0) - \phi(0) - \phi'(0) \alpha_0]}.$$

We terminate the search if the sufficient decrease condition is satisfied at  $\alpha_1$ , i.e.,

$$\phi(\alpha_1) \leq \phi(0) + c_1 \alpha_1 \phi'(0).$$

Otherwise, we need to backtrack again. We now have four pieces of information about  $\phi(\alpha)$ , so it is desirable to use all of them. Hence, at this and any subsequent backtrack steps during the current iteration of RGD, we use a cubic model  $\phi_c(\alpha)$  that interpolates the four pieces of information  $\phi(0)$ ,  $\phi'(0)$ , and the last two values of  $\phi(\alpha)$ , and set  $\alpha_k$  to the value of  $\alpha$  at which  $\phi_c(\alpha)$  has its local minimizer.

Let  $\alpha_{\text{prev}}$  and  $\alpha_{2\text{prev}}$  be the last two previous values of  $\alpha_k$  tried in the backtrack procedure. The cubic that fits  $\phi(0)$ ,  $\phi'(0)$ ,  $\phi(\alpha_{\text{prev}})$ , and  $\phi(\alpha_{2\text{prev}})$  is

$$\phi_c(\alpha) = a\alpha^3 + b\alpha^2 + \phi'(0)\alpha + \phi(0),$$

where

$$\begin{bmatrix} a \\ b \end{bmatrix} = \frac{1}{\alpha_{\text{prev}} - \alpha_{2\text{prev}}} \begin{bmatrix} \frac{1}{\alpha_{\text{prev}}^2} & \frac{-1}{\alpha_{2\text{prev}}^2} \\ -\frac{\alpha_{2\text{prev}}}{\alpha_{\text{prev}}^2} & \frac{\alpha_{\text{prev}}}{\alpha_{2\text{prev}}^2} \end{bmatrix} \begin{bmatrix} \phi(\alpha_{\text{prev}}) - \phi(0) - \phi'(0)\alpha_{\text{prev}} \\ \phi(\alpha_{2\text{prev}}) - \phi(0) - \phi'(0)\alpha_{2\text{prev}} \end{bmatrix}.$$

The local minimizer of this cubic is given by [11, (6.3.18)]

$$\frac{-b + \sqrt{b^2 - 3a\phi'(0)}}{3a},$$

and we set  $\alpha_k$  equal to this value. If necessary, this process is repeated, using a cubic interpolant of  $\phi(0)$ ,  $\phi'(0)$ , and the two most recent values of  $\phi$ , namely,  $\phi(\alpha_{\text{prev}})$  and  $\phi(\alpha_{2\text{prev}})$ , until a step size that satisfies the sufficient decrease condition is located. Numerical experiments in Section 5 demonstrate the usage and effectiveness of this line-search technique.

### Use of AI tools declaration

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

### Author contributions

Marco Sutti and Mei-Heng Yueh: conceptualization, data curation, formal analysis, funding acquisition, investigation, methodology development, project administration, resource provision, software development, supervision, validation, visualization, writing of the original draft, review, and editing. All authors of this article have been contributed equally. All authors have read and approved the final version of the manuscript for publication.

### Acknowledgments

The work of the first author was supported by the National Center for Theoretical Sciences in Taiwan (R.O.C.) under the NSTC grant 112-2124-M-002-009-. The work of the second author was supported by the National Science and Technology Council and the National Center for Theoretical Sciences in Taiwan.

## Conflict of interest

The authors declare that they have no conflict of interest.

## References

1. P. A. Absil, C. G. Baker, K. A. Gallivan, Trust-region methods on Riemannian manifolds, *Found. Comput. Math.*, **7** (2007), 303–330. <https://doi.org/10.1007/s10208-005-0179-9>
2. P. A. Absil, R. Mahony, R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*, Princeton University Press, Princeton, NJ, 2008. <https://doi.org/10.1515/9781400830244>
3. P. A. Absil, J. Malick, Projection-like retractions on matrix manifolds, *SIAM J. Optim.*, **22** (2012), 135–158. <https://doi.org/10.1137/100802529>
4. S. Angenent, S. Haker, A. Tannenbaum, R. Kikinis, On the Laplace-Beltrami operator and brain surface flattening, *IEEE Trans. Med. Imaging*, **18** (1999), 700–711. <https://doi.org/10.1109/42.796283>
5. N. Boumal, *An Introduction to Optimization on Smooth Manifolds*, Cambridge University Press, 2023. <https://doi.org/10.1017/9781009166164>
6. G. P. T. Choi, B. Chiu, C. H. Rycroft, Area-Preserving Mapping of 3D Carotid Ultrasound Images Using Density-Equalizing Reference Map, *IEEE. Trans. Biomed. Eng.*, **67** (2020), 2507–2517. <https://doi.org/10.1109/TBME.2019.2963783>
7. G. P. T. Choi, A. Giri, L. Kumar, Adaptive area-preserving parameterization of open and closed anatomical surfaces, *Comput. Biol. Med.*, **148** (2022), 105715. <https://doi.org/10.1016/j.compbiomed.2022.105715>
8. G. P. T. Choi, C. H. Rycroft, Density-equalizing maps for simply connected open surfaces, *SIAM J. Imaging Sci.*, **11** (2018), 1134–1178. <https://doi.org/10.1137/17M1124796>
9. P. T. Choi, K. C. Lam, L. M. Lui, FLASH: Fast landmark aligned spherical harmonic parameterization for genus-0 closed brain surfaces, *SIAM J. Imaging Sci.*, **8** (2015), 67–94. <https://doi.org/10.1137/130950008>
10. L. Cui, X. Qi, C. Wen, N. Lei, X. Li, M. Zhang, et al., Spherical optimal transportation, *Comput. Aided Des.*, **115** (2019), 181–193. <https://doi.org/10.1016/j.cad.2019.05.024>
11. J. E. Dennis Jr, R. B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations*, vol. 16 of Classics in Applied Mathematics, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1996. <https://doi.org/10.1137/1.9781611971200>
12. A. Edelman, T. A. Arias, S. T. Smith, The geometry of algorithms with orthogonality constraints, *SIAM J. Matrix Anal. Appl.*, **20** (1998), 303–353. <https://doi.org/10.1137/S0895479895290954>
13. M. S. Floater, Mean value coordinates, *Comput. Aided Geom. Des.*, **20** (2003), 19–27. [https://doi.org/10.1016/S0167-8396\(03\)00002-5](https://doi.org/10.1016/S0167-8396(03)00002-5)
14. D. Gabay, Minimizing a differentiable function over a differential manifold, *J. Optim. Theory Appl.*, **37** (1982), 177–219. <https://doi.org/10.1007/BF00934767>

15. D. G. Luenberger, *Introduction to linear and nonlinear programming*, vol. 28, Addison-Wesley Reading, MA, 1973.
16. J. Nocedal, S. J. Wright, *Numerical Optimization*, 2nd edition, Springer New York, NY, 2006. <https://doi.org/10.1007/978-0-387-40065-5>
17. W. Ring, B. Wirth, Optimization methods on Riemannian manifolds and their application to shape space, *SIAM J. Optim.*, **22** (2012), 596–627. <https://doi.org/10.1137/11082885X>
18. K. Su, L. Cui, K. Qian, N. Lei, J. Zhang, M. Zhang, et al., Area-preserving mesh parameterization for poly-annulus surfaces based on optimal mass transportation, *Comput. Aided Geom. Design*, **46** (2016), 76–91. <https://doi.org/10.1016/j.cagd.2016.05.005>
19. C. Udriște, *Convex functions and optimization methods on Riemannian manifolds*, vol. 297 of Mathematics and its applications, Kluwer Academic Publishers, Dordrecht, 1994. <https://doi.org/10.1007/978-94-015-8390-9>
20. M. H. Yueh, Theoretical foundation of the stretch energy minimization for area-preserving simplicial mappings, *SIAM J. Imaging Sci.*, **16** (2023), 1142–1176. <https://doi.org/10.1137/22M1505062>
21. M. H. Yueh, T. Li, W. W. Lin, S. T. Yau, A novel algorithm for volume-preserving parameterizations of 3-manifolds, *SIAM J. Imaging Sci.*, **12** (2019), 1071–1098. <https://doi.org/10.1137/18M1201184>
22. M. H. Yueh, W. W. Lin, C. T. Wu, S. T. Yau, A novel stretch energy minimization algorithm for equiareal parameterizations, *J. Sci. Comput.*, **78** (2019), 1353–1386. <https://doi.org/10.1007/s10915-018-0822-7>
23. X. Zhao, Z. Su, X. D. Gu, A. Kaufman, J. Sun, J. Gao, et al., Area-preservation mapping using optimal mass transport, *IEEE T. Vis. Comput. Gr.*, **19** (2013), 2838–2847. <https://doi.org/10.1109/TVCG.2013.135>



AIMS Press

©2024 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (<https://creativecommons.org/licenses/by/4.0>)