DOI: 10.1111/cgf.14598
Eurographics Symposium on Geometry Processing 2022
M. Campen and M. Spagnuolo
(Guest Editors)

Volume 41 (2022), Number 5

Harmonic Shape Interpolation on Multiply-connected Planar Domains

Dongbo Shi   Renjie Chen †
University of Science and Technology of China

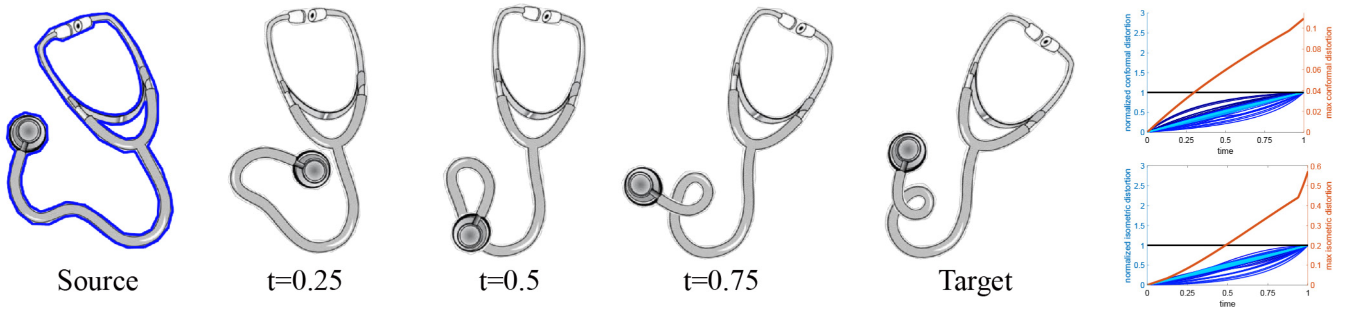


Figure 1: *Interpolation results of Stethoscope. Note that the domain contains 2 holes, and the deformation is quite large. Our intermediate frames look feasible, and the distortions are bounded by the input mappings.*

Abstract
Shape interpolation is a fundamental problem in computer graphics. Recently, there have been some interpolation methods developed which guarantee that the results are of bounded amount of geometric distortion, hence ensure high quality interpolation. However, none of these methods is applicable to shapes within the multiply-connected domains. In this work, we develop an interpolation scheme for harmonic mappings, that specifically addresses this limitation. We opt to interpolate the pullback metric of the input harmonic maps as proposed by Chen et al. [CWKBC13]. However, the interpolated metric does not correspond to any planar mapping, which is the main challenge in the interpolation problem for multiply-connected domains. We propose to solve this by projecting the interpolated metric into the planar harmonic mapping space. Specifically, we develop a Newton iteration to minimize the isometric distortion of the intermediate mapping, with respect to the interpolated metric. For more efficient Newton iteration, we further derived a simple analytic formula for the positive semidefinite (PSD) projection of the Hessian matrix of our distortion energy. Through extensive experiments and comparisons with the state-of-the-art, we demonstrate the efficacy and robustness of our method for various inputs.

CCS Concepts
• ***Computing methodologies*** → *Computer graphics; Animation; Shape analysis;*

1. Introduction

Shape interpolation is essential for many computer graphics and geometry processing applications. In particular, computer animation is a computer-aided process that heavily relies on the shape interpolation technique, which allows the automatic generation of intermediate frames, after the animator sets the tone of the animation using keyframes. A good interpolation method is crucial for producing visually feasible and pleasing intermediate frames.

Harmonic mappings are widely used in computer graphics, due to their smoothness and easy to work with for numerical computation. Chien et al. [CCW16] developed a highly efficient and effective method for producing interpolation results with bounded amount of distortion within the space of planar harmonic maps.

† Corresponding author.

They first interpolate the complex derivatives at each point, and then reconstruct the intermediate map using numerical integration. All their three variants use the complex logarithms, restricted to simply-connected domains. Unfortunately, this is not applicable to multiply-connected domains, as the integrability condition for the interpolated complex derivatives is violated, leading to 'broken' results, as Fig. 2 and the accompanying video show.

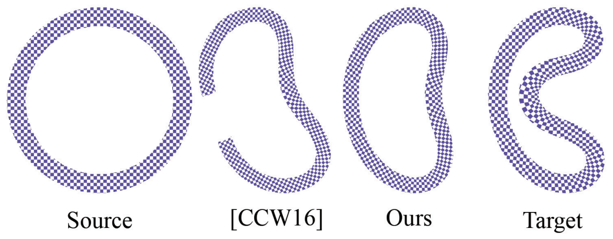


Figure 2: *Interpolation results of the annulus shape for $t = 0.5$. The method of Chien et al. [CCW16] breaks down on this simple multiply-connected domain. In contrast, our method generates a natural looking result.*

We present an interpolation method which is applicable to harmonic mappings on multiply-connected planar domains. The input to our algorithm is two (or more) harmonic mappings, which are in the same homology class, allowing the mappings to be continuously "morphed" between each other without breaking local injectivity. Chen and Weber [CW17] extended the harmonic mappings space onto multiply-connected domains for the deformation task. Their method produces high-quality deformation results which are of bounded amounts of geometric distortion, and this inspires us to perform the shape interpolation task within the same mapping space.

It has been shown in [CWKBC13, CCW16] that the pullback metric tensor is an ideal candidate for the shape interpolation problem as it seamlessly encodes the mapping and naturally leads to results with bounded distortion. As such, we also linearly interpolate the metric tensors of the input harmonic mappings. However, this comes with a major obstacle that, the interpolated metric does not necessarily correspond to a planar mapping. The same problem was encountered in [CCW16] for simply-connected domains, where the problem is addressed by blending the metric along the boundary only. While this approach could be applied to multiply-connected domains, it still does not result in planar harmonic mappings, since the Hilbert transformation proposed in [CCW16] does not apply to multiply-connected domains.

In this work, we propose to project the interpolated metric into the space of planar harmonic mappings as the final interpolation result. To measure the projection distance, we choose the popular symmetric Dirichlet energy of the resulting harmonic mapping, which measures the isometric distortion w.r.t. the reference, i.e. the interpolated metric. We design a Newton iteration routine to optimize the distortion along the domain boundary, and the bounded distortion theorem of planar harmonic maps [CW17, Theorem 4.2] ensures that the resulting map is locally injective and has bounded distortion. Furthermore, noting that the symmetric Dirichlet energy is convex in terms of the pullback metric [ACZW19], we derive a simple analytic formula for the PSD projection of the Hessian matrix during the Newton iteration, and this allows us to design a fully parallel interpolation algorithm on modern GPUs. We performed extensive experiments and comparisons with the state-of-the-art, and show that our method is highly efficient and effectively produces high-quality results with practically bounded distortion on multiply-connected domains.

Our main contributions are:

- We propose to project the blended metric into the harmonic mapping space to obtain feasible results with bounded distortion;
- We derive a simple analytic PSD projection formula for the Symmetric Dirichlet energy;
- Our interpolation algorithm is implemented fully on the GPU, enabling our method to run fast in the low dimensional mapping space.

2. Previous work

Due to the abundance of literature on shape interpolation techniques, we only review some of most related works. We refer the interested reader to [Wol98] and [Ale02] for comprehensive reviews of classical methods. Here, we concentrate on planar shape interpolation methods.

Usually, a shape interpolation method includes two steps: interpolate some geometric quantities which describe the input shapes, and reconstruct the geometry from the interpolated quantities. The main difference between different methods is the quantity being chosen for interpolation.

One popular approach is the As-Rigid-As-Possible (ARAP) method [ACOL00], which chooses the element-wise Jacobian of the source-to-target map as the geometry quantity for interpolation. By interpolating the rotation and shear components of the Jacobians separately, it leads to visually pleasing blending

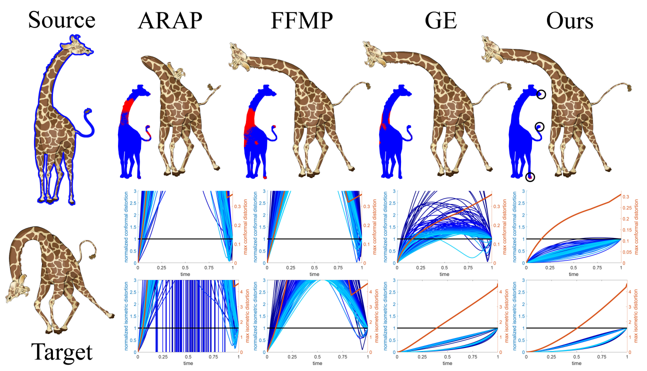


Figure 3: *Interpolation results at $t = 0.5$ for Giraffe. ARAP fails due to large rotation. The results of FFMP and GE are visually similar to ours, however the distortions are unbounded at various locations, which are marked in red in the images. In contrast, our result (last column) contains just a few spots where only the conformal distortion is slightly unbounded.*

with a relatively low amount of distortion in many cases. However, this approach fails to handle large rotations and is sensitive to the tessellation of the mesh. Another popular approach is to pose the interpolation problem as finding geodesics in the shape space [CPSS10, VTSSH15, SGK19, HRWW12], by optimizing a geodesic elastic (GE) energy in order to obtain the interpolation results.

Another option for shape interpolation is to use the differential coordinates [SK04, XZWB05, LSLCO05, KG08]. Among existing differential coordinates representations, the generalized barycentric coordinates [HF06, WBCG10, WG10, WBCGH11, WPG12] provide a simple and efficient way to define smooth spatial deformations. In the planar case, the deformation map is restricted to linear combinations of real or complex-valued basis functions (e.g. the Cauchy coordinates). Based on these coordinates, visually plausible interpolation results are produced in [CCW16] for shapes on simply-connected domains. Recently, Chen and Weber [CW17] generalize the Cauchy coordinates based harmonic mapping representation to multiply-connected domains for the deformation problem, which naturally leads to the question of designing shape interpolation methods on multiply-connected domains.

This work is also inspired by the tetrahedral metric interpolation method proposed by Aharon et al. [ACZW19], who proved that within the space of pullback metrics, the popular symmetric Dirichlet energy is convex, which further confirms that the metric is an ideal geometric quantity for the shape interpolation problem.

3. Background

For completeness, we include a brief introduction for some basic concepts for harmonic mapping and its discretization in 2D.

3.1. Harmonic Maps on Multiply-connected Domains

A harmonic map is a map f that satisfies the Laplace equation,

$$\Delta f = 0.$$

As we are focusing on the plane, the map $f : \Omega \to \mathbb{R}^2$, $f(x,y) = [u(x,y), v(x,y)]$ can be conveniently expressed as a complex-valued function $f : \Omega \subset \mathbb{C} \to \mathbb{C}$ where $f(z) = u(z) + \mathrm{i}v(z)$ with $z = x + \mathrm{i}y$. Using the Wirtinger derivatives of complex-valued functions, a map f is said to be *holomorphic* if it satisfies the Cauchy-Riemann equations, which can be expressed as $f_{\bar{z}} = 0$, and a map f is *harmonic* if and only if it satisfies the Laplace equation,

$$f_{\bar{z}z} = 0.$$

Chen and Weber [CW17] proved some key theorems regarding planar harmonic maps on multiply-connected domains. The first theorem [CW17, Theorem 4.1] is about the representation and the structure of harmonic maps:

Theorem 1 (Harmonic Map Decomposition). Let Ω be a multiply-connected planar domain with N holes $K_1, ..., K_N$, and choose N arbitrary points ρ_i inside K_i accordingly. Then, any harmonic map $f : \Omega \to \mathbb{C}$ can be represented as:

$$f(z) = \Phi(z) + \overline{\Psi(z)} + \sum_{i=1}^{N} \omega_i \log|z - \rho_i|, \tag{1}$$

where $\Phi, \Psi : \Omega \to \mathbb{C}$ are holomorphic functions, and $\omega_1, ..., \omega_N$ are complex coefficients.

By setting $\Psi(z_0) = 0$ for an arbitrary point $z_0 \in \Omega$, the harmonic decomposition in multiply-connected domains (1) is unique. The main difference for the harmonic decomposition between simply-connected domains and multiply-connected domains is the additional summation term, which is essential as it makes the representation of harmonic maps complete.

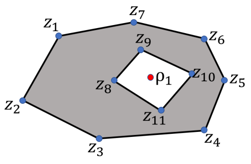
The second theorem [CW17, Theorem 4.2] relates the global distortion bound and the distortion bound over the domain boundary:

Theorem 2 (Bounded Distortion Harmonic Map). A planar harmonic map $f : \Omega \to \mathbb{C}$ on a multiply-connected domain with exterior boundary curve γ_0 oriented counterclockwise and interior boundary curves $\gamma_1, ..., \gamma_N$ oriented clockwise, is locally injective with an upper bound $\boldsymbol{\kappa} \in [0, 1)$ on the conformal distortion, a lower bound $\boldsymbol{\sigma}_2 > 0$ on the small singular value of the Jacobian, and an upper bound $\boldsymbol{\sigma}_1 < \infty$ on the large singular value at every point z in Ω if and only if:

$$\oint_{\gamma_0} \frac{f'_z(\omega)}{f_z(\omega)} d\omega + \sum_{i=1}^{N} \oint_{\gamma_i} \frac{f'_z(\omega)}{f_z(\omega)} d\omega = 0$$
$$0 \le \kappa(\omega) \le \boldsymbol{\kappa} \quad \forall \omega \in \partial\Omega$$
$$\sigma_1(\omega) \le \boldsymbol{\sigma}_1 \quad \forall \omega \in \partial\Omega$$
$$\boldsymbol{\sigma}_2 \le \sigma_2(\omega) \quad \forall \omega \in \partial\Omega.$$

3.2. Discretization of Harmonic Maps

Let $P = \{z_1, \ldots, z_m\} \subset \mathbb{C}$ be the vertices of a multiply-connected planar polygon (i.e. the cage in the inset). Chen and Weber [CW17] discretize the harmonic map (1) on multiply-connected domains based on the complex-valued Cauchy barycentric coordinates [WBCG10] as follows:



$$f(z) = \sum_{j=1}^{n} C_j(z)\varphi_j + \overline{\sum_{j=1}^{n} C_j(z)\psi_j}$$
$$C_j(z) = \begin{cases} \widetilde{C}_j(z) & j = 1, \ldots, m \\ \log|z - \rho_{j-m}| & j = m+1, \ldots, m+N, \end{cases} \tag{2}$$

where $\widetilde{C}_j(z)$ is the j^{th} Cauchy coordinate associated with vertex z_j (see [WBCG10]), N is the number of holes in the domain, ρ_j is any point inside the j^{th} inner boundary loop (i.e. hole), φ_j and ψ_j are complex coefficients with

$$\varphi_j = \overline{\psi_j}, j = m+1, \ldots, m+N. \tag{3}$$

The Wirtinger derivatives are easy to obtain as follows:

$$f_z(z) = \sum_{j=1}^{n} D_j(z)\varphi_j, \qquad \overline{f_{\bar{z}}(z)} = \sum_{j=1}^{n} D_j(z)\psi_j$$
$$D_j(z) = \begin{cases} \widetilde{D}_j(z) & j = 1, \ldots, m \\ \dfrac{1}{z - \rho_{j-m}} & j = m+1, \ldots, m+N, \end{cases} \tag{4}$$

where $\widetilde{D}_j(z)$ is the derivative of the Cauchy coordinates.

4. The Interpolation Problem

In general, a shape interpolation method consists of 2 steps:

1. Interpolate some geometric quantity to obtain the intermediate quantity;
2. Reconstruct the shape from the intermediate quantity as the interpolation result.

Our algorithm follows the same process. Section 4.1 introduces the metric tensor as the geometric quantity for our method. As it is infeasible to reconstruct the shape from the intermediate metric by direct integration, we propose to project it into the harmonic map space in order to obtain the result in section 4.2. Section 4.3 gives the formula of some necessary steps in our method.

We cast the shape interpolation problem as that of the interpolation of mappings. Assuming $f^0 : S \to S_0$ and $f^1 : S \to S_1$ are harmonic maps within the same homology class, which is required a priori for the existence of the smooth transformation from f_0 to f_1, we would like to produce an intermediate harmonic map $f^t : S \to S_t$ with the following properties,

1. (interpolation). $f^t|_{t=0} = f^0$ and $f^t|_{t=1} = f^1$.
2. (local injectivity). f^t is locally injective over the domain.
3. (bounded distortion). The conformal distortion and isometric distortion are bounded by the distortions of input mappings.

The first two properties are strictly enforced in our method, while the third is loosened to be having low overall distortion, which means in practice, the bounded distortion property can get slightly violated for some input mappings.

4.1. Linear Blending of The Pullback Metric

The pullback metric has been proven to be an excellent choice for the shape interpolation problem in both 2D [CWKBC13, CCW16] and 3D [ACZW19]. The pullback metric of a map $f(x) : S \subset \mathbb{R}^2 \to \mathbb{R}^2$ can be expressed in terms of the Jacobian:

$$M_f = J_f^T J_f.$$

This tensor is used to measure differential quantities such as lengths of infinitesimal vectors under map $f(S)$ using the standard Euclidean metric of S.

In the case of planar mapping, f can be written in complex numbers, as a function of z and $\bar{z}$, which allows us to derive the expression for $M_f := \begin{pmatrix} m_1 & m_2 \\ m_2 & m_3 \end{pmatrix}$ in terms of f_z and $\overline{f_{\bar{z}}}$, according to [CCW16]:

$$\begin{cases} m_1 = \left(\mathrm{Re} f_z + \mathrm{Re}\overline{f_{\bar{z}}}\right)^2 + \left(\mathrm{Im} f_z - \mathrm{Im}\overline{f_{\bar{z}}}\right)^2 \\ m_2 = -2\left(\mathrm{Re} f_z \cdot \mathrm{Im}\overline{f_{\bar{z}}} + \mathrm{Re}\overline{f_{\bar{z}}} \cdot \mathrm{Im} f_z\right) \\ m_3 = \left(\mathrm{Re} f_z - \mathrm{Re}\overline{f_{\bar{z}}}\right)^2 + \left(\mathrm{Im} f_z + \mathrm{Im}\overline{f_{\bar{z}}}\right)^2. \end{cases} \tag{5}$$

For the interpolation problem, assuming given planar maps f^0 and f^1, we linearly blend their pullback metrics M_f^0 and M_f^1 to obtain $M_f^t = (1-t)M_f^0 + tM_f^1$. Unfortunately, the blended metric M_f^t does not correspond to a planar mapping f^t. Chen et al. [CWKBC13] solved this problem by using a discrete curvature flow to flatten the metric. While Chien et al. [CCW16] blend the metric only for the boundary of the simply-connected domains and use Hilbert transform to obtain the mapping of the whole domain. However, neither approach is available for multiply-connected domains.

4.2. Harmonic Projection

After linearly blending the metrics, we face the problem of realizing M_f^t as a planar mapping within the multiply-connected domain Ω. To solve this, first we also focus on the boundary of Ω, whose bounded distortion implies a global distortion bound by Theorem 2. Then we project the metric to find a map in our harmonic mapping space.

Assume g is some reference mapping that realizes the linearly blended metric $M_g := M_f^t$, we note however that g is most likely to be curved and non-planar, therefore we would like to obtain a planar mapping h as 'close' to g as possible. We suggest that the isometric symmetric Dirichlet energy is a suitable measure for the gap between g and h, as it has been widely adopted for geometric optimization in numerous recent works, ranging from parameterization [SAPH04, SS15, KGL16], deformation [RPPSH17, CW17] to shape interpolation [ACZW19, SGK19]. It has a builtin barrier term that prevents elements from collapsing, ensuring that the local minima sought by the optimizer is locally injective, when given a locally injective initialization. Formally, the isometric energy of a mapping f (at a given point) is defined as:

$$E_{\text{iso}} = \frac{1}{2}\left(|J|^2 + |J^{-1}|^2\right) = \frac{1}{2}\left(\mathrm{Tr}(M) + \mathrm{Tr}(M^{-1})\right), \tag{6}$$

where $M = J^T J$ with J being the Jacobian of the mapping f.

Note that in our context, the mapping in concern is $P : g \to h$, where g is the reference corresponds to the linearly blended metric and h is a planar mapping in the harmonic space. The isometric energy for one sample can be written as follows (Appendix A):

$$E_{\text{iso}} = \frac{1}{2}\left(\mathrm{Tr}(M_g^{-1} M_h) + \mathrm{Tr}(M_g M_h^{-1})\right), \tag{7}$$

and the overall isometric energy is:

$$E_{\text{iso}}^{g \to h} = \oint_{\partial\Omega} E_{\text{iso}}(\omega) ds. \tag{8}$$

Then the interpolation problem becomes an optimization problem of finding a map h within our harmonic mapping space such that it minimizes the isometric energy (8). We note that in this overall energy, the isometric energy is integrated over the domain boundary, as it allows us design an efficient algorithm to perform the minimization, and in the meantime, Theorem 2 ensures that the resulting map is locally injective and has bounded amount of distortions.

4.3. Numerical Optimization

Suppose that:

$$M_g = \begin{pmatrix} g_1 & g_2 \\ g_2 & g_3 \end{pmatrix}, \quad M_h = \begin{pmatrix} h_1 & h_2 \\ h_2 & h_3 \end{pmatrix}, \tag{9}$$

then the isometric energy at one point is:

$$E_{\text{iso}} = \frac{g_3h_1 - 2g_2h_2 + g_1h_3}{2}\left(\frac{1}{g_1g_3 - g_2^2} + \frac{1}{h_1h_3 - h_2^2}\right) \qquad (10)$$
$$:= E(h_1,h_2,h_3).$$

By Eq. (4,5), this energy can be written in terms of φ and ψ, since h_1, h_2 and h_3 are functions of h_z and $\overline{h_{\bar{z}}}$, which in turn are linear combinations of φ and ψ,

$$E_{\text{iso}}(\varphi,\psi) = E\left(h_1(\varphi,\psi), h_2(\varphi,\psi), h_3(\varphi,\psi)\right). \qquad (11)$$

The objective energy of our optimization problem is:

$$E^{g\to h}(\varphi,\psi) = \frac{1}{|\partial\Omega|}\sum_{\omega\in\partial\Omega} E_{\text{iso}}(\omega). \qquad (12)$$

As our optimization problem involves a small number of variables (i.e. the length of vectors φ and ψ), we naturally turn to the Newton's method to optimize the energy given its quadratic convergence rate.

4.3.1. Gradient and Hessian

The first step of the Newton's method is to evaluate the gradient and Hessian of the objective function. In our case, they possess relatively simple closed-form expressions, as derived in Appendix B. The results are summarized in the following. We use bold symbols to denote *real* vectors and matrices. Let $D = (D_1, D_2, ..., D_n) \in \mathbb{C}^{1\times n}$ be a complex-valued row vector, we define the real-valued matrix $\mathbf{D}$ as:

$$\mathbf{D} = \begin{pmatrix} \text{Re}(D) & -\text{Im}(D) \\ \text{Im}(D) & \text{Re}(D) \end{pmatrix} \in \mathbb{R}^{2\times 2n}. \qquad (13)$$

The Wirtinger derivatives can be expressed as real 2-vectors:

$$\mathbf{h}_z = \begin{pmatrix} \text{Re}(h_z) \\ \text{Im}(h_z) \end{pmatrix}, \quad \overline{\mathbf{h}_{\bar{z}}} = \begin{pmatrix} \text{Re}(\overline{h_{\bar{z}}}) \\ \text{Im}(\overline{h_{\bar{z}}}) \end{pmatrix} \in \mathbb{R}^{2\times 1}. \qquad (14)$$

Substitute them into Eq. (5), we get the expressions for $h_i, i = 1,2,3$ as:

$$h_i = \begin{pmatrix} \mathbf{h}_z^T & \overline{\mathbf{h}_{\bar{z}}}^T \end{pmatrix}_{1\times 4} F_i \begin{pmatrix} \mathbf{h}_z \\ \overline{\mathbf{h}_{\bar{z}}} \end{pmatrix}_{4\times 1}, \qquad (15)$$

where $F_i, i = 1,2,3$ are constant 4×4 real matrices given in Appendix B.

The gradient of E_{iso} w.r.t. the $4n$ real variables is:

$$\nabla_{\boldsymbol{\varphi},\boldsymbol{\psi}} E_{\text{iso}} = \nabla_h E_{\text{iso}} \nabla_{\boldsymbol{\varphi},\boldsymbol{\psi}} h, \qquad (16)$$

and the $4n \times 4n$ real Hessian matrix of E_{iso} is:

$$\begin{aligned}\nabla^2_{\boldsymbol{\varphi},\boldsymbol{\psi}} E_{\text{iso}} &= \nabla_{\boldsymbol{\varphi},\boldsymbol{\psi}}\left(\nabla_h E_{\text{iso}} \nabla_{\boldsymbol{\varphi},\boldsymbol{\psi}} h\right) \\ &= \sum_{i=1}^{3} \partial_{h_i} E_{\text{iso}} \nabla^2_{\boldsymbol{\varphi},\boldsymbol{\psi}} h_i + \left(\nabla_{\boldsymbol{\varphi},\boldsymbol{\psi}} h\right)^T \nabla_h^2 E_{\text{iso}} \nabla_{\boldsymbol{\varphi},\boldsymbol{\psi}} h.\end{aligned} \qquad (17)$$

Finally, the gradient and Hessian of the full mapping energy are:

$$\begin{aligned}\nabla_{\boldsymbol{\varphi},\boldsymbol{\psi}} E^{g\to h} &= \frac{1}{|\partial\Omega|}\sum_{\omega\in\partial\Omega} \nabla E_{\text{iso}}(\omega) \\ \nabla^2_{\boldsymbol{\varphi},\boldsymbol{\psi}} E^{g\to h} &= \frac{1}{|\partial\Omega|}\sum_{\omega\in\partial\Omega} \nabla^2 E_{\text{iso}}(\omega).\end{aligned} \qquad (18)$$

4.3.2. Positive Definite Hessian

In Newton's method, the Hessian matrix must be positive definite, in order that the obtained update points in a descending direction. Unfortunately, the Hessian matrix $\nabla^2 E_{\text{iso}}$ is not positive definite in general, and we need to find a way to get a positive semi-definite (PSD) matrix which is the "closest" (in Frobenius norm) to $\nabla^2 E_{\text{iso}}$. Let $\nabla^2 E_{\text{iso}}{}^+$ denotes the PSD Hessian matrix for a single sample, the PSD Hessian matrix for the full map can be constructed as,

$$\nabla^2 E^{g\to h^+} = \frac{1}{|\partial\Omega|}\sum_{\omega\in\partial\Omega} \nabla^2 E_{\text{iso}}{}^+(\omega). \qquad (19)$$

A key observation for obtaining an analytic PSD projection $\nabla^2 E_{\text{iso}}{}^+$ is that the 2nd term in the Hessian expression (17) is always PSD, since $\nabla_h^2 E_{\text{iso}}$ is PSD. More formally, we have the following theorem.

Theorem 3 $\nabla_h^2 E_{\text{iso}}$ is positive semi-definite.

This is due to that the symmetric Dirichlet energy is a convex function of the metric, as proven by Aharon et al. [ACZW19]. Therefore, we can separate the Hessian matrix into 2 parts, $\nabla^2_{\boldsymbol{\varphi},\boldsymbol{\psi}} E_{\text{iso}} = H_1 + H_2$, with $H_1 = \sum_{i=1}^3 \partial_{h_i} E_{\text{iso}} \nabla^2_{\boldsymbol{\varphi},\boldsymbol{\psi}} h_i$ and $H_2 = \left(\nabla_{\boldsymbol{\varphi},\boldsymbol{\psi}} h\right)^T \nabla_h^2 E_{\text{iso}} \left(\nabla_{\boldsymbol{\varphi},\boldsymbol{\psi}} h\right)$. According to Theorem 3, H_2 is PSD, therefore we only need to focus on H_1, whose expression is derived in Appendix B,

$$H_1 = 2\begin{pmatrix} \mathbf{D}^T & 0 \\ 0 & \mathbf{D}^T \end{pmatrix} K_1 \begin{pmatrix} \mathbf{D} & 0 \\ 0 & \mathbf{D} \end{pmatrix}, \qquad (20)$$

with

$$K_1 = \sum_{i=1}^{3} \alpha_i F_i = \begin{pmatrix} \alpha_1+\alpha_3 & 0 & \alpha_1-\alpha_3 & -\alpha_2 \\ 0 & \alpha_1+\alpha_3 & -\alpha_2 & -\alpha_1+\alpha_3 \\ \alpha_1-\alpha_3 & -\alpha_2 & \alpha_1+\alpha_3 & 0 \\ -\alpha_2 & -\alpha_1+\alpha_3 & 0 & \alpha_1+\alpha_3 \end{pmatrix}.$$

The eigenvalues of K_1 are:

$$\begin{aligned}\lambda_{1,2} &= \alpha_1+\alpha_3+\sqrt{(\alpha_1-\alpha_3)^2+\alpha_2^2} \\ \lambda_{3,4} &= \alpha_1+\alpha_3-\sqrt{(\alpha_1-\alpha_3)^2+\alpha_2^2}.\end{aligned} \qquad (21)$$

The "closest" PSD projection of K_1 can be obtained by examining the signs of the eigenvalues and replacing the negative eigenvalues with 0. More specifically, we consider the following 3 cases:

- $\lambda_{1,2} \geq \lambda_{3,4} \geq 0$:

$$K_1^+ = K_1.$$

- $0 > \lambda_{1,2} \geq \lambda_{3,4}$:

$$K_1^+ = 0.$$

- $\lambda_{1,2} \geq 0 > \lambda_{3,4}$:

$$K_1^+ = \frac{\lambda_1}{\lambda_1-\lambda_3}\begin{pmatrix} s & 0 & \alpha_1-\alpha_3 & -\alpha_2 \\ 0 & s & -\alpha_2 & -\alpha_1+\alpha_3 \\ \alpha_1-\alpha_3 & -\alpha_2 & s & 0 \\ -\alpha_2 & -\alpha_1+\alpha_3 & 0 & s \end{pmatrix},$$

where $s = \frac{\lambda_1-\lambda_3}{2}$.

Thus we obtain $\nabla^2_{\boldsymbol{\varphi},\boldsymbol{\psi}} E^{+}_{\text{iso}} = H_1^{+} + H_2$, with $H_1^{+} = 2\begin{pmatrix}\mathbf{D}^T & 0\\ 0 & \mathbf{D}^T\end{pmatrix} K_1^{+} \begin{pmatrix}\mathbf{D} & 0\\ 0 & \mathbf{D}\end{pmatrix}$. This analytic expression enables us to implement our Newton iteration efficiently, and further allows easy parallelization on modern GPUs.

Table 1: *Numbers of iterations and runtime (s) for different types of projection on various shapes at $t = 0.5$. Delete: delete H_1; FullProj: project $H_1 + H_2$ together; PartProj: project H_1 (our method).*

Model	Delete		FullProj		PartProj(CPU)		PartProj(GPU)	
	#iter	runtime	#iter	runtime	#iter	runtime	#iter	runtime
Archery	11	2.944	10	2.942	10	3.069	10	0.281
Bar_holes	15	1.722	10	1.121	10	1.015	10	0.161
Links	13	1.658	11	1.523	11	1.442	11	0.200
Stethoscope	18	3.167	17	2.971	17	2.900	17	0.241
Rect	19	0.880	10	0.484	9	0.391	9	0.095
Rex	21	3.796	12	2.166	13	2.325	13	0.276
Rings	12	3.857	12	3.956	12	3.888	12	0.549
Scissors	15	2.261	13	2.107	14	2.058	14	0.223
Jackbean	12	1.952	11	1.788	12	1.938	12	0.241
Giraffe	19	4.035	24	5.156	23	4.763	23	0.295

Table 1 compares the performance of three types of projection. ‘Delete’ means setting $H = H_2$ which is PSD by construction, ‘FullProj’ means performing numerical eigenvalue decomposition on $H = H_1 + H_2$ to get its PSD projection and ‘PartProj’ means projecting H_1 alone using our analytic expression. The first three tests are performed on the CPU. We also list the results of our GPU implementation for the same input. ‘Delete’ has the least runtime per iteration, while taking the most iterations to converge. It takes nearly the same number of iterations for ‘FullProj’ and ‘PartProj’ to converge, and ‘PartProj’ runs faster since it avoids performing numerical eigenvalue factorizations. Compared with the other three setups, our GPU implementation takes the least runtime till convergence.

4.3.3. Injectivity Certification

To ensure that the map stays locally injective, we verify that condition $|h_z| > |\overline{h_{\bar{z}}}|$ holds, and backtrack in our line search otherwise. For this part, we follow the steps in [CW17, Section 8] and keeps the map locally injective by utilizing the Lipschitz continuity of the Wirtinger derivatives on the boundary segments.

5. Implementation

Algorithm 1 provides the pseudocode of our method. In the remaining of this section, we point out some practical aspects of the implementation.

5.1. Initialization

The initialization for the Newton iteration can have significant influence on its convergence rate. In our experiments, we set Φ_0^t, Ψ_0^t to be the interpolation result of the previous frame. In all the results presented in the paper, we uniformly sample $t \in [0,1]$ and interpolate 500 frames for each input mapping.

5.2. Energy, Gradient and Hessian Evaluation

By the benefits of harmonic maps in Theorem 2, when computing $E^{g\to h}$, $\nabla E^{g\to h}$ and $\nabla^2 E^{g\to h^{+}}$ using Eq. (12), (18) and (19), we use uniformly distributed samples over the boundary. Furthermore,

Algorithm 1 Harmonic Interpolation based on Newton's Method

```
Input: source map $(\varphi^0, \psi^0)$, target map $(\varphi^1, \psi^1)$, time $t$
Output: interpolation result $(\varphi^t, \psi^t)$
 1: compute the reference metric $M_g$
 2: initialize with a locally injective map $(\varphi_0^t, \psi_0^t)$
 3: loop
 4:     compute $E_k^{g\to h}$ by (12)
 5:     set $\boldsymbol{g} \leftarrow \nabla E_k^{g\to h}$ by (18)
 6:     set $\boldsymbol{H} \leftarrow \nabla^2 E_k^{g\to h^{+}}$ by (19)
 7:     $(\boldsymbol{H}, \boldsymbol{g}) \leftarrow$ EliminateVariables $(\boldsymbol{H}, \boldsymbol{g})$
 8:     solve $\boldsymbol{H}\boldsymbol{d} = -\boldsymbol{g}$, where $\boldsymbol{d} = \begin{pmatrix} d\varphi \\ d\psi \end{pmatrix}$
 9:     set $s \leftarrow$ LineSearch$(E_k^{g\to h}, \boldsymbol{d}, \boldsymbol{g}, \varphi_k^t, \psi_k^t)$
10:     if $E_k - E_{k+1} < \varepsilon$ then
11:         return $(\varphi_k^t, \psi_k^t)$
12:     else
13:         $(\varphi_{k+1}^t, \psi_{k+1}^t) \leftarrow (\varphi_k^t, \psi_k^t) + s(d\varphi, d\psi)$
14:     end if
15: end loop
```

due to the high computational cost of the Hessian matrix, we use only a subset of the boundary sample when evaluating the Hessian. Let $\mathbb{G} = \{\omega_1, \omega_2, \ldots, \omega_{|\mathbb{G}|}\} \subset \partial\Omega$ be a set of uniformly distributed samples, $E^{g\to h}$, $\nabla E^{g\to h}$ and $\nabla^2 E^{g\to h^{+}}$ can be approximated as:

$$E^{g\to h} = \frac{1}{|\mathbb{G}|}\sum_{i=1}^{|\mathbb{G}|} E_{\text{iso}}(\omega_i), \quad \nabla E^{g\to h} = \frac{1}{|\mathbb{G}|}\sum_{i=1}^{|\mathbb{G}|} \nabla E_{\text{iso}}(\omega_i) \tag{22}$$
$$\nabla^2 E^{g\to h^{+}} = \frac{1}{|\mathbb{H}|}\sum_{i=1}^{|\mathbb{H}|} \nabla^2 E^{+}_{\text{iso}}(\omega_i),$$

where $\mathbb{H} \subset \mathbb{G}$ with $|\mathbb{H}| = r|\mathbb{G}|$, and r is the Hessian sampling ratio from the energy samples. Table 2 shows the runtime and number of iterations till convergence of our method for one frame under different r. For most inputs, $r = 20\%$ leads to the fastest convergence for each frame. Therefore, we set $r = 20\%$ throughout our experiment.

Table 2: *Average runtime (ms) and number of Newton iterations till convergence of interpolating each frame with different sampling rate r for the Hessian.*

Model	$r = 100\%$		$r = 50\%$		$r = 20\%$		$r = 10\%$		$r = 5\%$	
	time	#iter	time	#iter	time	#iter	time	#iter	time	#iter
Archery	178.4	3.0	121.3	3.0	87.7	3.0	75.5	3.0	1380.8	100.0
Bar_holes	48.3	3.0	39.2	3.0	34.0	3.0	32.6	3.2	35.3	4.2
Links	71.3	3.0	52.9	3.0	41.6	3.0	38.3	3.0	101.6	15.5
Stethoscope	81.4	3.0	61.6	3.0	49.8	3.0	47.1	3.0	51.6	3.9
Rect	24.1	3.4	23.6	3.4	23.0	3.4	24.8	3.5	24.4	3.6
Rex	108.1	3.1	76.9	3.1	58.7	3.1	54.1	3.2	817.5	100.0
Rings	175.5	2.2	118.4	2.2	92.0	2.4	351.1	15.1	1662.1	95.4
Scissors	78.1	3.0	58.5	3.0	47.0	3.0	51.6	4.3	364.9	54.9
Jackbean	79.2	2.9	60.5	2.9	48.5	2.9	46.6	3.0	45.2	3.1
Giraffe	149.1	4.2	105.1	4.2	79.5	4.2	77.0	4.7	961.8	100.0

5.3. Elimination of Variables

The Hessian matrix given in Eq. (18, 19) is singular, due to that the representation of the harmonic map in Eq. (2) has some degrees of redundancy. As discussed in [CW17], it has in fact a nullspace of dimension $4N + N + 1$, among which, $4N$ is due to the 2 complex

DOF per hole for each of the two Cauchy barycentric mapping, N is due to the conjugation constraint on some of the coefficients (3), and the term 1 is due to the decomposition (1). In our optimization problem, there is an additional dimension to be considered. As the objective is defined with $M_h = J_h^T J_h$, it is obvious that, if R is a rotation, then $(RJ_h)^T(RJ_h) = J_h^T(R^TR)J_h = M_h$, which means the objective is invariant to global rotation. To eliminate this nullvector, we set $\mathrm{Re}\varphi_1 = 0$.

6. Results

We implement our method in C++ using the GPU with NVIDIA CUDA Toolkit 11.2 and generate input for our interpolation method using the method of [CW17]. There are some default parameters in the algorithm throughout our experiments. The number of the energy and gradient samples $|\mathbb{G}|$ is set to 10,000, while the number of Hessian samples is $|\mathbb{H}| = 0.2|\mathbb{G}|$. The termination condition for our Newton iteration is set to be $\Delta E_{\mathrm{iso}} \leq 10^{-7}$. When comparing to mesh-based methods, a mesh with 10,000 vertices is used. The conformal distortion $\kappa = \frac{\sigma_1}{\sigma_2}$ and the symmetric Dirichlet isometric distortion are evaluated on each vertex and normalized for the distortion plot in each example. In all the distortion graphs, we show 100 samples with the largest distortion for every method. Our experiments are performed on a machine running Windows 10 with Intel(R) Core(TM) i5-8500 CPU @ 3.00GHz (6 cores), 32GB and has a NVIDIA GeForce RTX 2080 Ti graphics card.

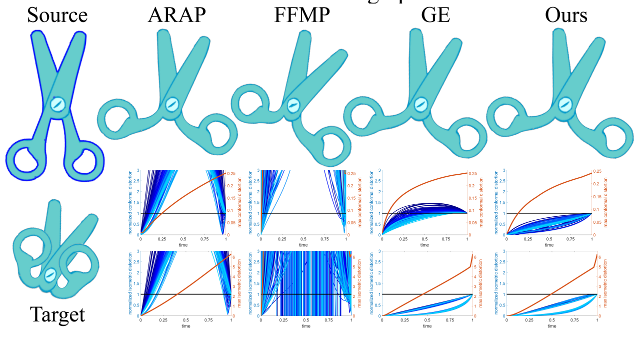


Figure 4: *Comparison of interpolation results of Scissors for $t = 0.5$. The left handle of Scissors in the FFMP result appears smaller than the other. The results of ARAP, GE and our method look similar, however only our result shows bounded amount of distortion, as the distortion plots reveal.*

As the state-of-the-art shape interpolation methods, including [CCW16,CWKBC13], cannot handle multiply-connected domains, we compare our results only to three mesh-based interpolation methods, i.e. ARAP [ACOL00], FFMP [KG08] and GE [SGK19]. The first two methods are direct methods, which generate the intermediate frame by solving one or two large linear systems for the vertex positions. In contrast, GE iteratively solves a nonlinear optimization problem. It's worth noting that GE gave an analytic eigen-system for the symmetric Dirichlet energy, based on the SVD decomposition of the Jacobian. Our method differs from these three methods, mainly in two aspects. First, our input and interpolation results are in the harmonic space. Second, our optimization uses samples only from the boundary, instead of the whole domain.

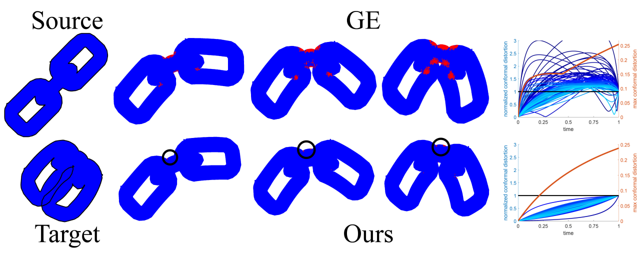


Figure 5: *Interpolation results of Links for $t = 0.25, 0.5, 0.75$. Focus on the change of conformal distortion, plenty of samples are marked as unbounded in red by GE (top), while only few samples of our method (bottom) are marked.*

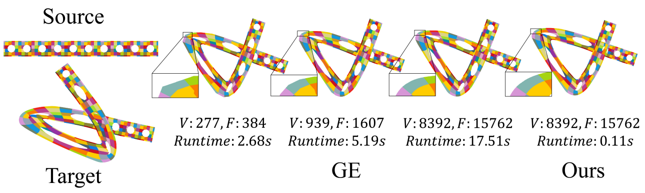


Figure 6: *Interpolation results for $t = 0.75$ of different mesh resolutions. Notice that for mesh-based method (three results in the middle), lower mesh resolution leads to faster runtime but less smooth mappings.*

When the input mappings contain large rotations, ARAP produces unfeasible results, such as the neck and tail of the giraffe in Fig. 3, one arm of Jackbean and a corner of Bar_holes in Fig. 10. For the other inputs, ARAP produces results visually similar to ours, although it does not have bounded distortions. FFMP makes some improvement, but still performs badly in some cases. In Fig. 4 and Fig. 10, the left handle of Scissors and the bottom corners of Rect show unnatural scaling which leads to unbounded distortions. In our experiments, GE produces visually pleasant interpolation results with bounded distortions on most inputs, and the results often appear similar to ours. In Fig. 3 and Fig. 5, we can see that some points in the domain have distortions perform badly within the interpolation sequence for GE, while our results has well behaved distortions throughout the sequence.

Table 3: *Comparison of average runtime (ms) and number of Newton iterations per frame between GE and our method under two different mesh resolutions.*

Model	$V \approx 8k, F \approx 15k$				$V \approx 80k, F \approx 150k$			
	GE		Ours		GE		Ours	
	runtime	#iter	runtime	#iter	runtime	#iter	runtime	#iter
Archery	1804.9	35.4	87.7	3.0	16532.9	35.2	89.2	3.0
Bar_holes	1889.4	43.2	34.0	3.0	17750.1	41.3	45.6	3.0
Links	1856.8	36.2	41.6	3.0	17359.6	35.6	48.3	3.0
Rex	2802.7	58.4	58.7	3.1	23210.9	55.4	81.2	3.1
Rings	2980.2	58.7	92.0	2.4	25200.4	57.7	107.4	2.4
Scissors	2119.5	38.1	47.0	3.0	21404.0	47.1	63.1	3.0
Jackbean	2421.4	43.8	48.5	2.9	19416.0	41.6	68.5	2.9
Giraffe	7780.3	164.5	79.5	4.2	56775.1	165.6	112.7	4.2

Table 3 shows the average runtime and number of iterations for

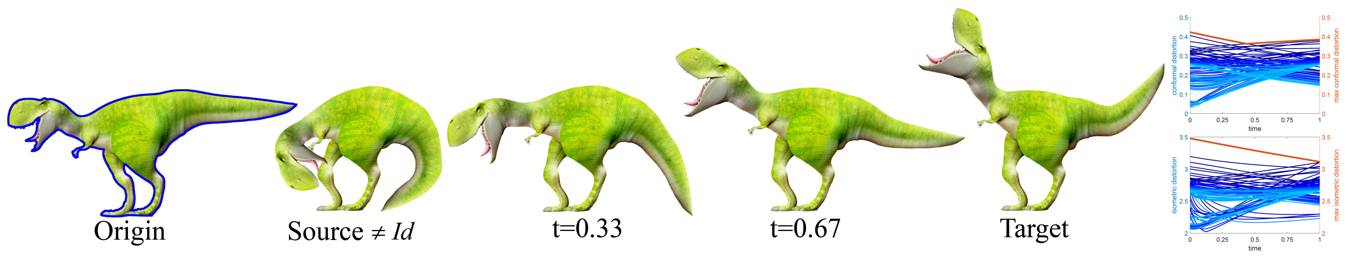


Figure 7: *Interpolation results of Rex. When the input mapping $f_0 \neq Id$, our method also produces feasible intermediate frames. As it is difficult to normalize the distortions in this case, we plot the distortion graph without normalization. As the graph shows, the distortions of the interpolation result are still bounded by the input mappings f_0 and f_1.*

each frame with different mesh resolutions. We can see that the runtime of our method is not affected by the mesh resolution. When the number of vertices and faces increases, the runtime of our method stays nearly the same. In contrast, the runtime of GE increases accordingly. In fact, as Fig. 6 shows, there is a trade-off between the runtime and mapping smoothness for mesh-based methods.

Table 4: *Comparison of average runtime (ms) per frame with different methods and average number of Newton iterations per frame with GE and our method.*

Model	ARAP	FFMP	GE		Ours	
	runtime	runtime	runtime	#iter	runtime	#iter
Archery	48.9	188.3	1804.9	35.4	87.7	3.0
Bar_holes	49.5	186.8	1889.4	43.2	34.0	3.0
Links	51.6	192.2	1856.8	36.2	41.6	3.0
Stethoscope	50.4	183.1	6184.8	149.8	49.8	3.0
Rect	52.4	190.1	3586.5	85.7	23.0	3.4
Rex	54.2	198.8	2802.7	58.4	58.7	3.1
Rings	53.0	187.5	2980.2	58.7	92.0	2.4
Scissors	55.9	198.2	2119.5	38.1	47.0	3.0
Jackbean	59.0	204.0	2421.4	43.8	48.5	2.9
Giraffe	61.9	210.3	7780.3	164.5	79.5	4.2

Table 4 compares the runtime between our method and mesh-based methods. For most shapes, our method takes only 2-4 Newton iterations to produce one intermediate frame, which means that it converges extremely fast. For each frame, ARAP solves one large sparse linear system whose size is proportional to the number of vertices, while FFMP needs to solve two large linear systems of similar sizes. In contrast, our method iteratively solves a dense small linear system whose size is proportional to the dimension of the planar harmonic mapping space. This enables our method to run faster in most cases, as it takes very few iterations to converge.

The Bar_holes and Rings examples in Fig. 10 show that our method performs well for complex domains. Note that our method trivially applies to the simply-connected domains. Fig. 8 shows our iterative algorithm converges to visually the same result as [CCW16] for one such domain, which essentially solves a least square problem based on the Hilbert Transform [Bel15]. Note that the input mapping f_0 for our method is not restricted to identity mapping. Fig. 7 shows the interpolated frames at $t = 0.33$ and $t = 0.67$ for the Rex shape. It can be seen that both frames look feasible and the distortions are bounded by the input mappings f_0 and f_1.

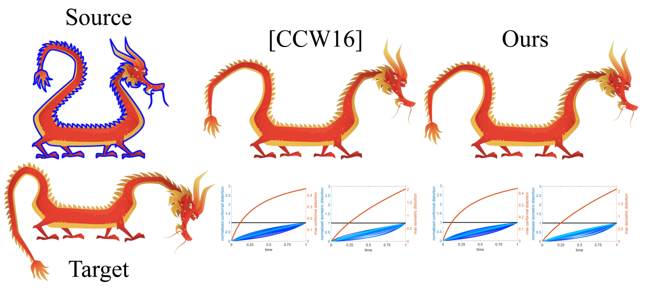


Figure 8: *Interpolation result of Dragon for $t = 0.5$. For this input which is on a simply-connected domain, we get visually the same result as [CCW16].*

In the accompanying video, we show some smooth animations for some of these results, in comparison to the mesh-based methods. Although we have no guarantee that the interpolation will be smooth w.r.t. time since each frame is optimized independently, the results of our method exhibit good temporal coherence in practice, as the video shows.

Fig. 9 shows how the number of intermediate frames (for $t \in [0,1]$) affects the convergence of our method. Apparently, more intermediate frames mean the time difference between every two consecutive frames is smaller, and therefore the interpolation results should be closer. This enables our method to converge faster, as

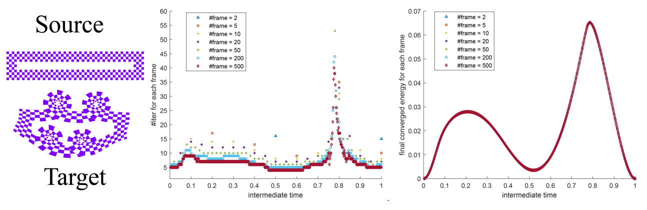


Figure 9: *The convergence behavior of our algorithm. The middle graph plots the number of iterations taken for our method to converge, the right graph plots the energy after convergence.*

can be observed from the scatter plot in the middle. This is that we initialize our method for each frame with the result of the previous frame, therefore with more intermediate frames, the initialization gets closer to the local optima of our energy. The right plot shows the energy when our algorithm converges for each frame. Surprisingly, our method consistently converges to the same minima for each frame, when given different initializations, which proves the robustness of our method.

7. Summary and Discussion

We have presented a method for interpolating planar harmonic mappings on multiply-connected domains. The input to our algorithm is two (or more) harmonic mappings, and our method produces harmonic mapping with practically bounded distortion on the same domain as the output. We propose to linearly blend the metric of the input mappings which ensures that the interpolated metric has pointwise bounded distortion. However, this leads to the problem that the blended metric may not correspond to a planar mapping. We address this by projecting the interpolated metric into the planar harmonic mapping space in order to obtain the interpolation result. In the projecting process, we choose the symmetric Dirichlet energy to measure the distance between the reference and the resulting harmonic map, and we further design a highly efficient Newton iteration for the optimization problem.

Compared with mesh-based methods, our method performs optimization over a sampling of the boundary, which means that the interpolation results and runtime of our method are not influenced by the meshing quality. Furthermore, our method works in the low-dimensional harmonic mapping space, leading to a much smaller system than the mesh-based methods. Using the key observation that the symmetric Dirichlet energy is convex in terms of the metric, the PSD projection for the Hessian matrix possesses a simple analytic form and can be easily implemented on the GPU, enabling our Newton iteration to run extremely efficient.

7.1. Limitation and Future work

While we prove that the interpolated reference metric has distortion strictly bounded pointwise by the input mappings, we cannot guarantee this property holds after the projection. Our method produces visually pleasing results by minimizing the global distortion and guarantees the results are locally injective thanks to the bounded distortion theorem for harmonic maps [CW17]. However, it may occur in practice that the *relative* distortion is not bounded by the input. The Rings in Fig. 10 shows one such example. It remains interesting to find an interpolation method with a strict pointwise distortion bound.

Acknowledgements

We thank the anonymous reviewers for their constructive comments that help improve this paper. This research is supported by the National Key R&D Program of China (2020YFC1523102), the National Natural Science Foundation of China (62072422), and the Anhui Provincial Natural Science Foundation (2008085MF195).

References

[ACOL00] ALEXA M., COHEN-OR D., LEVIN D.: As-rigid-as-possible shape interpolation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), pp. 157–164. 2, 7

[ACZW19] AHARON I., CHEN R., ZORIN D., WEBER O.: Bounded distortion tetrahedral metric interpolation. *ACM Transactions on Graphics 38*, 6 (2019), 182.1–182.17. 2, 3, 4, 5

[Ale02] ALEXA M.: Recent advances in mesh morphing. In *Computer graphics forum* (2002), vol. 21, Wiley Online Library, pp. 173–198. 2

[Bel15] BELL S. R.: *The Cauchy transform, potential theory and conformal mapping*. CRC press, 2015. 8

[CCW16] CHIEN E., CHEN R., WEBER O.: Bounded distortion harmonic shape interpolation. *ACM Transactions on Graphics 35*, 4 (2016), 1–15. 1, 2, 3, 4, 7, 8

[CPSS10] CHAO I., PINKALL U., SANAN P., SCHRÖDER P.: A simple geometric model for elastic deformations. *ACM Transactions on graphics (TOG) 29*, 4 (2010), 1–6. 3

[CW17] CHEN R., WEBER O.: GPU-accelerated locally injective shape deformation. *ACM Transactions on Graphics (TOG) 36*, 6 (2017), 1–13. 2, 3, 4, 6, 7, 9

[CWKBC13] CHEN R., WEBER O., KEREN D., BEN-CHEN M.: Planar shape interpolation with bounded distortion. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 1–12. 1, 2, 4, 7

[HF06] HORMANN K., FLOATER M.: Mean value coordinates for arbitrary planar polygons. *ACM Transactions on Graphics 25*, 4 (2006), 1424–1441. 3

[HRWW12] HEEREN B., RUMPF M., WARDETZKY M., WIRTH B.: Time-discrete geodesics in the space of shells. In *Computer Graphics Forum* (2012), vol. 31, Wiley Online Library, pp. 1755–1764. 3

[KG08] KIRCHER S., GARLAND M.: Free-form motion processing. *ACM Transactions on Graphics 27*, 2 (2008). 3, 7

[KGL16] KOVALSKY S. Z., GALUN M., LIPMAN Y.: Accelerated quadratic proxy for geometric optimization. *ACM Transactions on Graphics 35*, 4 (2016), 1–11. 4

[LSLCO05] LIPMAN Y., SORKINE O., LEVIN D., COHEN-OR D.: Linear rotation-invariant coordinates for meshes. *ACM Transactions on Graphics 24*, 3 (2005), 479–487. 3

[RPPSH17] RABINOVICH M., PORANNE R., PANOZZO D., SORKINE-HORNUNG O.: Scalable locally injective mappings. *ACM Transactions on Graphics 36*, 2 (2017), 1–16. 4

[SAPH04] SCHREINER J., ASIRVATHAM A., PRAUN E., HOPPE H.: Inter-surface mapping. *ACM Transactions on Graphics 23*, 3 (2004), 870–877. 4

[SGK19] SMITH B., GOES F. D., KIM T.: Analytic eigensystems for isotropic distortion energies. *ACM Transactions on Graphics (TOG) 38*, 1 (2019), 1–15. 3, 4, 7

[SK04] SHEFFER A., KRAEVOY V.: Pyramid coordinates for morphing and deformation. In *Proceedings. 2nd International Symposium on 3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004.* (2004), IEEE, pp. 68–75. 3

[SS15] SMITH J., SCHAEFER S.: Bijective parameterization with free boundaries. *ACM Transactions on Graphics 34*, 4 (2015), 70:1–70:9. 4

[VTSSH15] VON-TYCOWICZ C., SCHULZ C., SEIDEL H.-P., HILDEBRANDT K.: Real-time nonlinear shape interpolation. *ACM Transactions on Graphics (TOG) 34*, 3 (2015), 1–10. 3

[WBCG10] WEBER O., BEN-CHEN M., GOTSMAN C.: Complex barycentric coordinates with applications to planar shape deformation. *Computer Graphics Forum 28*, 2 (2010), 587–597. 3

[WBCGH11] WEBER O., BEN-CHEN M., GOTSMAN C., HORMANN K.: A complex view of barycentric mappings. *Computer Graphics Forum 30*, 5 (2011). 3

[WG10] WEBER O., GOTSMAN C.: Controllable conformal maps for shape deformation and interpolation. *ACM Transactions on Graphics 29*, 4 (2010). 3

[Wol98] WOLBERG G.: Image morphing: a survey. *Visual Computer 14*, 8/9 (1998), 360–372. 2

[WPG12] WEBER O., PORANNE R., GOTSMAN C.: Biharmonic coordinates. *Computer Graphics Forum 31*, aop (2012). 3

[XZWB05] XU D., ZHANG H., WANG Q., BAO H.: Poisson shape interpolation. *Graphical Models 68*, 3 (2005), 268–281. 3

Appendix A: Isometric energy

We consider the mapping $P : g \to h$ which can be expressed as $P = h \circ g^{-1}$, implying that its Jacobian matrix is $J = J_h J_{g^{-1}}$ and its metric is $M = J_P^T J_P = J_{g^{-1}}^T J_h^T J_h J_{g^{-1}}$.

For the $\mathrm{Tr}(M)$ part, we have:

$$\mathrm{Tr}(M) = \mathrm{Tr}(J_{g^{-1}}^T J_h^T J_h J_{g^{-1}}) = \mathrm{Tr}(J_{g^{-1}} J_{g^{-1}}^T J_h^T J_h) = \mathrm{Tr}(M_g^{-1} M_h),$$

which is due to that $\left(J_{g^{-1}} J_{g^{-1}}^T\right) M_g = \left(J_{g^{-1}} J_{g^{-1}}^T\right)\left(J_g^T J_g\right) = I_d$.

Similarly, we obtain $\mathrm{Tr}(M^{-1})$:

$$\begin{aligned}\mathrm{Tr}(M_P^{-1}) &= \mathrm{Tr}\left(\left(J_{g^{-1}}^T J_h^T J_h J_{g^{-1}}\right)^{-1}\right)\\ &= \mathrm{Tr}\left(\left(J_{g^{-1}}\right)^{-1} (M_h)^{-1} \left(J_{g^{-1}}^T\right)^{-1}\right)\\ &= \mathrm{Tr}\left(J_g M_h^{-1} J_g^T\right) = \mathrm{Tr}\left(J_g J_g^T M_h^{-1}\right) = \mathrm{Tr}\left(M_g M_h^{-1}\right).\end{aligned}$$

Appendix B: gradient and Hessian of the isometric energy

Let $\varphi = (\varphi_1, ..., \varphi_n) \in \mathbb{C}^{n\times 1}$ and $\psi = (\psi_1, ..., \psi_n) \in \mathbb{C}^{n\times 1}$ be complex column vectors containing our variables. Their real expressions are:

$$\boldsymbol{\varphi} = \begin{pmatrix}\mathrm{Re}(\varphi)\\ \mathrm{Im}(\varphi)\end{pmatrix}, \quad \boldsymbol{\psi} = \begin{pmatrix}\mathrm{Re}(\psi)\\ \mathrm{Im}(\psi)\end{pmatrix}$$

Apply the multivariable chain rule, we get the gradient of E with respect to the $4n$ variables $\boldsymbol{\varphi}$, $\boldsymbol{\psi}$:

$$\nabla_{\boldsymbol{\varphi},\boldsymbol{\psi}} E = \nabla_h E \nabla_{\boldsymbol{\varphi},\boldsymbol{\psi}} h = \sum_{i=1}^{3} \alpha_i \nabla_{\boldsymbol{\varphi},\boldsymbol{\psi}} h_i,$$

where $\alpha_i = \partial_{h_i} E$ can be derived from Eq. (10). Substitute $\mathbf{h}_z = \mathbf{D}\boldsymbol{\varphi}$, $\overline{\mathbf{h}_{\bar z}} = \mathbf{D}\boldsymbol{\psi}$ into (5), we get:

$$h_i = \begin{pmatrix}\boldsymbol{\varphi}^T & \boldsymbol{\psi}^T\end{pmatrix}\begin{pmatrix}\mathbf{D}^T & 0\\ 0 & \mathbf{D}^T\end{pmatrix} F_i \begin{pmatrix}\mathbf{D} & 0\\ 0 & \mathbf{D}\end{pmatrix}\begin{pmatrix}\boldsymbol{\varphi}\\ \boldsymbol{\psi}\end{pmatrix}$$

$$F_1 = \begin{pmatrix}1 & 0 & 1 & 0\\ 0 & 1 & 0 & -1\\ 1 & 0 & 1 & 0\\ 0 & -1 & 0 & 1\end{pmatrix}, \quad F_2 = \begin{pmatrix}0 & 0 & 0 & -1\\ 0 & 0 & -1 & 0\\ 0 & -1 & 0 & 0\\ -1 & 0 & 0 & 0\end{pmatrix}$$

$$F_3 = \begin{pmatrix}1 & 0 & -1 & 0\\ 0 & 1 & 0 & 1\\ -1 & 0 & 1 & 0\\ 0 & 1 & 0 & 1\end{pmatrix},$$

then we obtain the gradient:

$$\begin{aligned}\nabla E &= 2\begin{pmatrix}\mathbf{D}^T & 0\\ 0 & \mathbf{D}^T\end{pmatrix}(\alpha_1 F_1 + \alpha_2 F_2 + \alpha_3 F_3)\begin{pmatrix}\mathbf{h}_z\\ \overline{\mathbf{h}_{\bar z}}\end{pmatrix}\\ &= 2\begin{pmatrix}\mathbf{D}^T & 0\\ 0 & \mathbf{D}^T\end{pmatrix} K_1 \begin{pmatrix}\mathbf{h}_z\\ \overline{\mathbf{h}_{\bar z}}\end{pmatrix}.\end{aligned}$$

Apply the chain rule again, we get the $4n \times 4n$ Hessian of E as:

$$\begin{aligned}\nabla^2_{\boldsymbol{\varphi},\boldsymbol{\psi}} E &= \sum_{i=1}^{3} \partial_{h_i} E \nabla^2_{\boldsymbol{\varphi},\boldsymbol{\psi}} h_i + (\nabla_{\boldsymbol{\varphi},\boldsymbol{\psi}} h)^T \nabla_h^2 E\, (\nabla_{\boldsymbol{\varphi},\boldsymbol{\psi}} h)\\ &= \sum_{i=1}^{3} \alpha_i \nabla^2_{\boldsymbol{\varphi},\boldsymbol{\psi}} h_i + \sum_{i=1,j=1}^{3,3} \beta_{i,j} \nabla_{\boldsymbol{\varphi},\boldsymbol{\psi}} h_i (\nabla_{\boldsymbol{\varphi},\boldsymbol{\psi}} h_j)^T,\end{aligned}$$

where $\beta_{i,j} = \frac{\partial^2 E}{\partial h_i \partial h_j}$ can also be derived from Eq. (10). For $\nabla^2_{\boldsymbol{\varphi},\boldsymbol{\psi}} h_i$ and $\nabla_{\boldsymbol{\varphi},\boldsymbol{\psi}} h_i (\nabla_{\boldsymbol{\varphi},\boldsymbol{\psi}} h_j)^T$, we have:

$$\begin{aligned}&\nabla^2_{\boldsymbol{\varphi},\boldsymbol{\psi}} h_i = 2\begin{pmatrix}\mathbf{D}^T & 0\\ 0 & \mathbf{D}^T\end{pmatrix} F_i \begin{pmatrix}\mathbf{D} & 0\\ 0 & \mathbf{D}\end{pmatrix}\\ &\nabla_{\boldsymbol{\varphi},\boldsymbol{\psi}} h_i (\nabla_{\boldsymbol{\varphi},\boldsymbol{\psi}} h_j)^T =\\ &\quad 4\begin{pmatrix}\mathbf{D}^T & 0\\ 0 & \mathbf{D}^T\end{pmatrix} F_i \begin{pmatrix}\mathbf{h}_z\\ \overline{\mathbf{h}_{\bar z}}\end{pmatrix}\begin{pmatrix}\mathbf{h}_z^T & \overline{\mathbf{h}_{\bar z}}^T\end{pmatrix} F_j \begin{pmatrix}\mathbf{D} & 0\\ 0 & \mathbf{D}\end{pmatrix}.\end{aligned}$$

Finally, the $4n \times 4n$ Hessian with respect to $\boldsymbol{\varphi}, \boldsymbol{\psi}$ is:

$$\begin{aligned}&\nabla^2 E_{\mathrm{iso}} = \begin{pmatrix}\mathbf{D}^T & 0\\ 0 & \mathbf{D}^T\end{pmatrix} K \begin{pmatrix}\mathbf{D} & 0\\ 0 & \mathbf{D}\end{pmatrix}\\ &K = 2\sum_{i=1}^{3} \alpha_3 F_i + 4\sum_{i=1,j=1}^{3,3} \beta_{i,j} F_i \begin{pmatrix}\mathbf{h}_z \mathbf{h}_z^T & \mathbf{h}_z \overline{\mathbf{h}_{\bar z}}^T\\ \overline{\mathbf{h}_{\bar z}} \mathbf{h}_z^T & \overline{\mathbf{h}_{\bar z}}\,\overline{\mathbf{h}_{\bar z}}^T\end{pmatrix} F_j\\ &\quad = 2K_1 + 4K_2.\end{aligned}$$

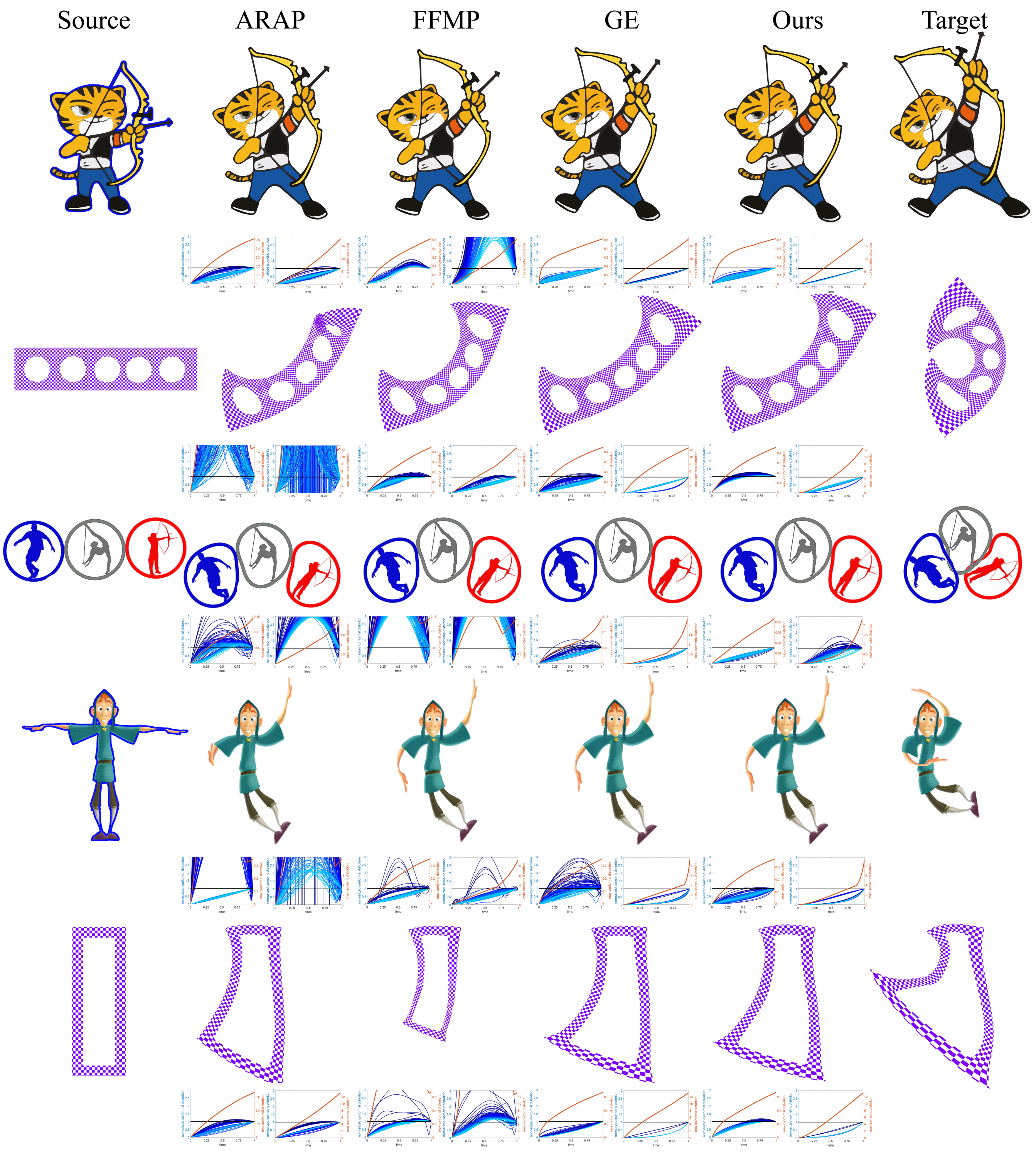


Figure 10: *More interpolation results at t = 0.5 of Archery, Bar_holes, Rings, Jackbean and Rect, in comparison with three other methods.*