Computers & Graphics 35 (2011) 639–649

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

SMI 2011: Full Paper

A topology-preserving optimization algorithm for polycube mapping

Shenghua Wan [a], Zhao Yin [b], Kang Zhang [a], Hongchao Zhang [c], Xin Li [a,*]

[a] Department of Electrical and Computer Engineering, and Center for Computation and Technology, Louisiana State University, Baton Rouge, 70803, USA
[b] Department of Automation, Xiamen University, Xiamen 361005, China
[c] Department of Mathematics, Louisiana State University, Baton Rouge, 70803, USA

ARTICLE INFO

Article history:
Received 14 December 2010
Received in revised form
13 March 2011
Accepted 14 March 2011
Available online 23 March 2011

Keywords:
Polycube mapping
Surface parameterization
Domain optimization

ABSTRACT

We present an effective optimization framework to compute polycube mapping. Composed of a set of small cubes, a polycube well approximates the geometry of the free-form model yet possesses great regularity; therefore, it can serve as a nice parametric domain for free-form shape modeling and analysis. Generally, the more cubes are used to construct the polycube, the better the shape can be approximated and parameterized with less distortion. However, corner points of a polycube domain are singularities of this parametric representation, so a polycube domain having too many corners is undesirable. We develop an iterative algorithm to seek for the optimal polycube domain and mapping, with the constraint on using a restricted number of cubes (therefore restricted number of corner points). We also use our polycube mapping framework to compute an optimal common polycube domain for multiple objects simultaneously for lowly distorted consistent parameterization.

1. Introduction

Computing the parameterization of 3D shapes (surfaces/solids) on specific canonical domains is an important problem in shape modeling, and it can facilitate many computer graphics and geometric processing tasks. Polycube mapping was first introduced by [33]. It parameterizes a closed surface onto a polycube domain, which is composed of a set of small cubes. A polycube has the same topology of the given surface, and it is usually constructed to approximate the geometry of the surface. Therefore, the surface parameterization on a polycube domain often has much smaller distortion than that on a planar domain. Meanwhile, the polycube domain still possesses great regularity; each subpatch is a rectangle; transitions between adjacent patches are simple rotation and translation except on corner points. Due to many of these advantages, the polycube mapping has been used in many graphics and shape modeling applications such as texture mapping [33] and synthesis [21], shape morphing [11], spline construction [34,35], and volumetric matching [23,39,40].

Intuitively, the more cubes one uses to construct the polycube, the better the domain can approximate the original model, which brings parameterization very small area and angle distortion. However, corner points are singularity points of the parameterization. They are undesirable in many tasks such as spline construction [34,35], physics-based simulations [15], etc. On the other hand, if one uses fewer cubes to construct a simpler domain with fewer corner points, the parameterization will possess larger distortion due to the dissimilarity of geometric structures between the model and the domain shape. Therefore, when a fundamental question is asked: What is the optimal polycube domain? A reasonable answer can be an optimized balancing between the singularity number and mapping distortion. More specifically, we try to solve the following problem: given a surface S and a budget n of the singularity point number, what is the optimal shape of the polycube domain P so that the parameterization $f: S \rightarrow P$ has the least distortion and P has no more than n corners?

Depending on applications, different metrics (angle distortion, area distortion, isometry distortion, etc.) have been studied and used to measure the mapping quality. *Harmonic functions* are most widely used in constructing lowly distorted mapping. With a fixed boundary condition, a function $\phi(x,y)$ is harmonic if it is a solution of Laplace's equation. When a boundary condition is given, ϕ is a minimizer of the Dirichlet energy [28,10] and it possesses great smoothness. For example, conformal parameterization can be constructed by two conjugate harmonic functions [14,31]. In this paper, we use harmonic functions to construct polycube mapping, minimizing a metric energy composed of shape-preserving and area-preserving terms. The framework is general and can be used for other metrics. A similar idea, proposed by Pietroni et al. [27], considered the trade off between the mapping distortion and the simplicity of the domain, solves the surface parameterization over abstract domains by locally optimizing the mapping on subregions then globally smoothing it.

* Corresponding author. Tel.: +1 225 578 0289; fax: +1 225 578 4831.
E-mail address: xinli@lsu.edu (X. Li).

0097-8493/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2011.03.018

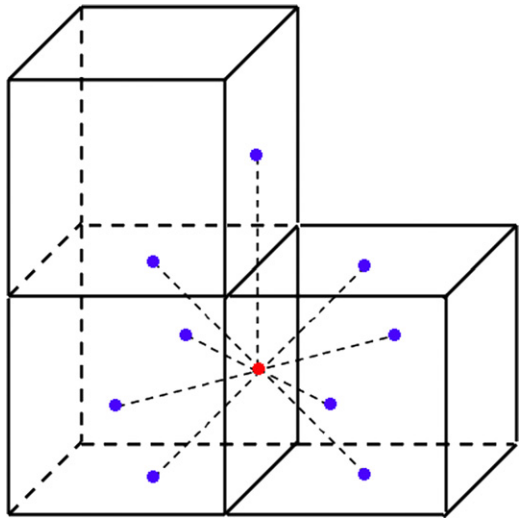

Fig. 1. Part of the dual graph corresponding to one facet (red node) and its neighboring facets (blue node). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Now the optimal polycube maps can be formulated as solving argmin $E(P,f)$ for a given shape S, where energy function E is defined on any mapping $f: S \rightarrow P$ and P is a polycube with n corners. Since the domain P is part of the optimization, it is extremely difficult. We restrict our optimization to a subspace of this problem, which we call a *topology-preserving polycube mapping*. Specifically, given an initial polycube domain $P=\{P_i\}$, the *topology* of the polycube P is defined by its dual graph (see Fig. 1) $DM=\{DV,DE\}$. $DV=\{dv_1,\ldots,dv_n\}$ are nodes corresponding to rectangle subpatches $\{P_i\}$. DE is a set of edges: an edge $[dv_i,dv_j]$ is in DE, if P_i and P_j are adjacent to each other. We say two polycubes $P=\{P_1,\ldots,P_n\}$ and $Q=\{Q_1,\ldots,Q_m\}$ are *topologically equivalent*, if their dual graphs DP and DQ are isomorphic. Therefore, given an initial polycube P, our goal is to find the optimal polycube P' and the mapping f that minimizes distortion $E(P,f)$, in the same topological equivalence class (without changing the structure of its dual graph).

This paper has three main contributions.

- We formulate the above optimal polycube mapping problem, and present a polycube mapping computation framework based on the given restricted complexity of polycube domain.
- We develop efficient optimization solvers to seek the topology-preserving optimal polycube domain and mapping iteratively.
- We extend the polycube optimization algorithm to multiple objects, for the construction of the common optimal domain for multiple models.

2. Related work

Surface parameterization using harmonic functions: Theories and technologies in surface parameterization have been widely studied and they have been playing a critical role in many geometric processing tasks in graphics, CAGD, visualization, vision, medical imaging, physical simulation, etc. Many effective techniques have been developed to solve the parameterization under different distortion metrics with different boundary conditions. A thorough review is beyond the scope of this paper, and we refer to three great surveys/tutorials of surface mapping and their applications in [13,32,19]. One widely used scalar function used for constructing low-distorted surface parameterization is the harmonic function. The discrete harmonic map was first proposed by Pinkall and Polthier [28] and introduced to the computer graphics field by Eck et al. [10]. By discretizing the energy defined in [28], Desbrun et al. [8] constructed free-boundary harmonic maps. Harmonic maps are preferable due to at least two important reasons: (1) it is meaningful from physics' point of view. A harmonic map minimizes the Dirichlet energy and leads to a minimal surface [28]; (2) it can be easily discretized and efficiently calculated from the computational aspect. A discrete harmonic map can be approximated either through FEM analysis of the harmonic energy [10] or via mimicking the mean value property of harmonic functions [12]. The computation of discrete harmonic mapping can be written as the optimization of a quadratic energy and be efficiently solved as a sparse linear system.

Polycube mapping: As a useful parametric domain, polycube maps have been studied in many different shape modeling applications. Tarini et al. [33] invented the concept of polycube map and applied it to the texture mapping and synthesis. Fan et al. [11] extended it to generate cross parameterization and morphing by mapping surfaces to polycubes then composing the map by finding the correspondence between them. In these approaches, polycube maps are computed by extrinsic methods such as projections. Wang et al. [34] introduced an intrinsic method for polycube maps and built splines representation on the polycube parametric domain. Compared with extrinsic methods, the intrinsic approach reduced the mapping distortion significantly. Later, Wang et al. [35] developed user controllable polycube maps for manifold spline construction. Both approaches required much user involvement in polycube design. Lin et al. [24] presented an automatic polycube mapping approach, but the bijectivity was not guaranteed. Recently, He et al. [18] presented a divide-and-conquer approach for automatic polycube map construction. In that paper, the bijectivity was guaranteed and the mapping had shown low angle and area distortion. Han et al. [17] applied volumetric polycube maps to construct hexahedral shell mesh. Xia et al. [36] introduces an editable polycube mapping, based on a divide-and-conquer strategy, which gives much more control over the quality of the induced subdivision surface and makes processing of large models with complex geometry and topology feasible.

3. Algorithms overview

A polycube domain P is composed of a set of rectangular patches P_i. A polycube map is therefore composed of a set of rectangular maps. We use the harmonicity and area distortion to measure the mapping quality and optimize the domain shape as well as the mapping.

Ideally, given a metric, we shall simultaneously optimize the polycube domain P as well as the mapping $f: S \rightarrow P$ to minimize the distortion $E(f)$. We can formulate this as minimizing $E(\mathbf{x},\mathbf{y})=E(x_1,x_2,\ldots,x_{3n},y_1,y_2,\ldots,y_{3n})$, with the constraints that $(x_{3i-2},x_{3i-1},x_{3i})$ is a point on S, and $(y_{3i-2},y_{3i-1},y_{3i})$ is the corresponding corner point on the polycube P, for $i=1,\ldots,n$.

Directly solving this nonlinear optimization is highly expensive. As will be discussed shortly (Sections 5 and 6), the derivatives of E over $\mathbf{y}$ can be computed efficiently, but the derivatives of E over $\mathbf{x}$ could not be computed in practice. Without derivatives of the object function, this optimization with complicated constraints is difficult even for moderately large n. To make full use of the partial derivative information of the objective function, we iteratively do the optimization over $\mathbf{x}$ (for optimal polycube corner mapping) and $\mathbf{y}$ (for optimal polycube domain shape) separately. Hence, gradient-based nonlinear optimization methods using the derivatives of $\partial E/\partial \mathbf{y}$ can be developed to efficiently optimize the subproblem $E(\mathbf{x},\mathbf{y})$ for fixed $\mathbf{x}$. Meanwhile, a

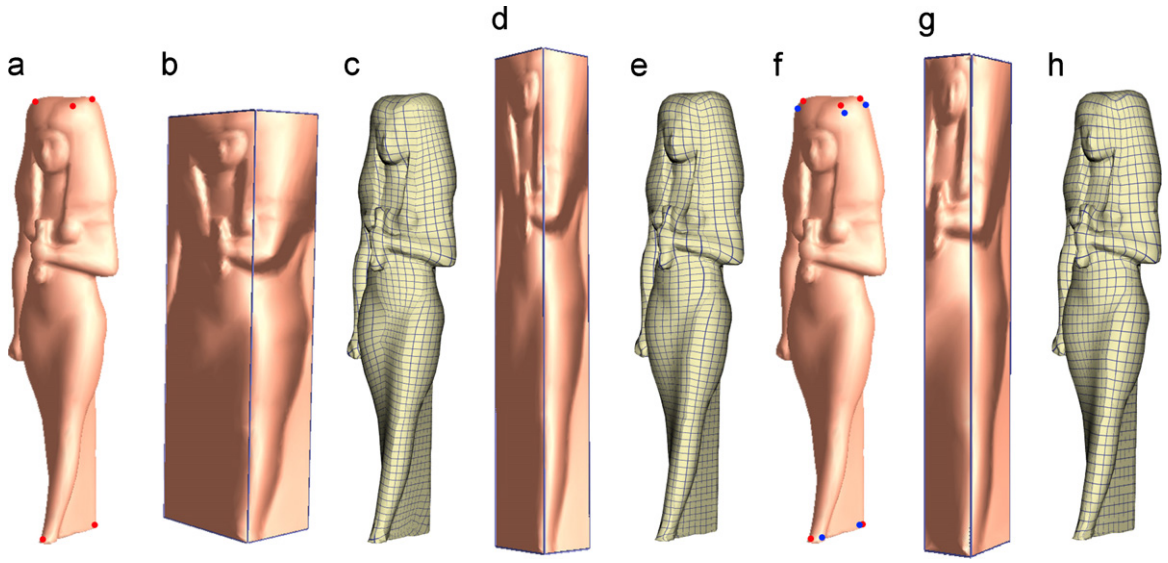

Fig. 2. Algorithm overview. (a) Original surface with eight corner points (red). (b), (c) Initial polycube domain and mapping. (d), (e) Optimized polycube domain and mapping. The harmonic energy with area distortion term is reduced from 5.4414 to 4.7812. (f) The optimized polycube mapping with eight new corner points (blue) with a lower harmonic energy of 4.5961. (g), (h) Final optimized domain and optimized mapping after two iterations. The grid quality is improved. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

derivative-free optimization algorithm is developed to optimize the subproblem $E(\mathbf{x},\mathbf{y})$ for fixed $\mathbf{y}$. During each iteration, when the shape of every rectangle and the mappings of its four corner points are determined, we can compute/update the mapping efficiently (see Sections 4 and 6). The proposed iterative polycube mapping optimization framework therefore has the following three steps (illustrated in Fig. 2).

1. *Initial polycube domain construction* (Section 4): Given a budget number of corner points, an initial polycube domain is constructed either automatically or manually, meeting the corner point budgets; then the corner point mapping and the initial polycube mapping are computed.
2. *Optimizing polycube domain shapes* (Section 5): Preserving the topology of the polycube, the scaling of subpatches is optimized so that mapping energy is minimized.
3. *Optimizing polycube mapping* (Section 6): Without changing shape of the polycube, the surface-polycube mapping is optimized by searching the optimal corner point mapping.

Algorithm 1. Optimal polycube mapping.

input: surface S, corner point number n;
output: polycube mapping $f: S \to P$;
Construct an initial polycube P_0, whose corner point number $\leq n$;
Compute an initial mapping $f_i: S \to P_i; i = 0$;
repeat
  $i \leftarrow i+1$;
  Optimize the polycube domain P_i, s.t. distortion of mapping f_{i-1} is minimized;
  Optimize the polycube map $f_i: S \to P_i$;
until $|P_i - P_{i-1}| < \varepsilon$;
Perform a global smoothing.

The framework is formulated in Algorithm 1. Note that in our iterative process, we keep on optimizing scaling factors of subpatches and the corner points. Then (1) *polycube domain optimization* takes corner points decided by the current mapping f_i as the input and solve scaling of subpatches to reduce mapping distortion; and (2) *polycube mapping optimization* uses the scaled polycube P_{i+1} as the target domain and optimizes the location of corner points. This iterative refinement converges when the polycube domain shape P_i does not change any longer.

4. Constructing initial polycube and mapping

The initial polycube can be constructed manually [33,34], or automatically [24,18]. We also use a simple voxelization algorithm (Section 4.1) to generate the polycube. Since this initial polycube and maps (Section 4.2) will be optimized to minimize the distortion, a simple, efficient, and adaptive (to different corner budgets) scheme such as this voxelization algorithm is sometimes enough. The following optimization framework is general, and can optimize an initial polycube mapping constructed via different methods.

4.1. Polycube construction via voxelization

Given a solid object M, supposing its boundary surface is represented by a triangle mesh $S = (\mathcal{V}_s, \mathcal{E}_s, \mathcal{F}_s)$ where $\mathcal{V}_s, \mathcal{E}_s, \mathcal{F}_s$ are vertex, edge, and face sets, we construct a polycube domain $P = (\mathcal{V}_p, \mathcal{E}_p, \mathcal{F}_P)$ and corresponding corner points mapping using a voxelization algorithm. Fig. 3 illustrates a polycube construction example of a Buddha model through voxelization.

We use an octree to represent the object. The subdivision starts from a rectangular bounding box. Each cell (rectangular cuboid) can be labeled as *inside* or *outside*. We remove all interior faces that are shared by two inside cells, and finally merge all inside cells to one polycube P. The remaining faces form the boundary surface of P. We further merge these remaining faces to a set of big rectangle facets of the polycube. Iteratively, we merge two adjacent faces if the result remains a planar convex polygon. After merging, only rectangle facets are left. The vertices of these rectangles are called corner points, denoted as $\mathcal{V}_{CP}$. And the edges of the rectangles form the connectivity of the corner points $\mathcal{E}_{CP}$. For each corner $v \in \mathcal{V}_{CP}$, we use the simple projection method [33] to find its corresponding points on S. Without ambiguity, we also call these corresponding points *corner points* on S, denoted as $\mathcal{V}_{CS}$; they will be mapped to corners in the initial polycube mapping.

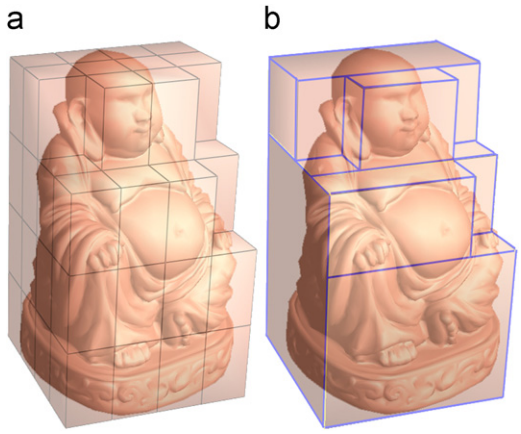

Fig. 3. Voxelization for polycube construction.

The voxelization algorithm is simple, automatic and efficient. Moreover, the octree's depth can be adaptively decided by the number of corner points.

Voxelization approaches sometimes provide unnecessary zig-zagged domain shapes when the geometry of the object is not well aligned with principal axes, which can be undesirable. Then other polycube domain construction algorithms (e.g., [33,34,24,18]) may be used to construct the initial mapping, and our subsequent optimization paradigm can still be applied to refine the domain shape and improve mapping quality.

4.2. Initial polycube mapping

Given the initial polycube P, corner point correspondences $\mathcal{V}_{CS}$, $\mathcal{V}_{CP}$, and cube edges $\mathcal{E}_{CP}$, we compute an initial polycube mapping $f: S \to P$ as follows. Denote the position of each vertex v on S as $X=(x^0,x^1,x^2)$ and its image on the polycube as $U=f(X)=(u^0,u^1,u^2)\in P$; also denote three components of the vector function f as f^0,f^1, and f^2.

A discrete harmonic parameterization [10] is a bijective map from S to a 2D (u,v)-domain, $h: S\to D, S\subset\mathcal{R}^3, D\subset\mathcal{R}^2$ such that the discrete harmonic energies of both u and v components are minimized. When the target planar domain D is convex, and a diffeomorphic boundary mapping is given, the harmonic mapping h is bijective. Therefore, we decompose S to multiple patches, each of which will be mapped to a rectangle facet P_i on P. The harmonic energy of a mapping function on k-th $(k=1,2,3)$ component is defined as

$$H^k=\frac{1}{2}\sum_i\sum_{v_j\in N(v_i)} w_{ij}(f^k(X_i)-f^k(X_j))^2, \tag{1}$$

where $N(v_i)$ is the set of all 1-ring neighboring vertices of v_i. $w_{ij}=\frac{1}{2}(\cot\alpha_{ij}+\cot\beta_{ij})$ is the well known cotangent weight [10] defined on the edge $[v_i,v_j]\in\mathcal{E}_S$, where α_{ij} and β_{ij} are two angles opposite to the edge $[v_i,v_j]$.

For each polycube edge in $[v_{pi},v_{pj}]\in\mathcal{E}_{CP}, v_{pi},v_{pj}\in\mathcal{V}_{CP}$, we trace curves to connect their corresponding points $v_{si},v_{sj}\in\mathcal{V}_{CS}$ using shortest paths following algorithms introduced in [30]. After this, the harmonic mapping computation is straightforward. We parameterize these traced paths to polycube edges using the arc-length parameterization. On each facet of the polycube, corner and edge mapping decides the boundary condition and the interior mapping can be computed by solving two sparse linear systems [10].

5. Optimizing polycube domain

Given a polycube mapping $f: S\to P=\{f_i: S_i\to P_i\}$ defined on a set of topological rectangle patches on S. We want to find the optimal re-scaled P_i so that mapping distortion is minimized. We use a distortion energy E composed of the harmonic energies $H^t(f), t=0,1,2$ and an area-stretching term $A(f)$:

$$H^t=\sum_{P_k}H^t_k=\sum_{P_k}\left(\sum_{e_{i,j}\in P_k}\frac{1}{2}w_{ij}(f^t(X_i)-f^t(X_j))^2\right), \tag{2}$$

$$A=\sum_{P_k}\sum_{F_{i,j,h}\in P_k}\frac{(\Delta(U_i,U_j,U_h))^2}{\Delta(X_i,X_j,X_h)}, \tag{3}$$

$$E=H^0+H^1+H^2+\alpha A, \tag{4}$$

where $\Delta(X_i,X_j,X_h)$ and $\Delta(U_i,U_j,U_h)$ denote the original area of triangle (v_i,v_j,v_h) and the area of its image under the mapping; P_k is a facet of polycube and $F_{i,j,h}$ is a triangle on this facet; α is a weighting factor balancing the harmonic and area-stretching terms.

When optimizing the polycube shape, we restrict our re-scaling on P_i such that (1) it preserves the total area of the polycube and (2) it does not increase the number of corner points. Specifically, we divide the polycube P into different rectangular facets in each coordinate plane (see Fig. 4).

First, we sort the coordinates of all corner points in three axes, and denote them as $\{\alpha^i_j\}, i=0,1,2, j=0,\ldots,N_i$. We translate the left-bottom of the polycube to the origin, so that any $\alpha^i_0=0$.

Then supposing a facet P_k is perpendicular to the u^t coordinate axis, we (1) denote the coordinate of P_k in u^t-axis as α^t_j and (2) on each patch perpendicular to u^t, denote its corresponding coordinates as $[\alpha^{t+1}_j,\alpha^{t+1}_{j+1}]$ and $[\alpha^{t+2}_j,\alpha^{t+2}_{j+1}]$. The superscript indicates the corresponding axis (u^0,u^1, or u^2), so $t+1$ actually denotes $(t+1)$ mod 3. In our following derivations, the addition of superscripts denotes their addition modulo 3.

Now we can denote the length of each segment in u^t-axis as $\beta^t_{j+1}=\alpha^t_{j+1}-\alpha^t_j$; and adjacent facets (faces connected by a same polycube edge) should share a same corresponding scaling factor β, to prevent the increase of corner points.

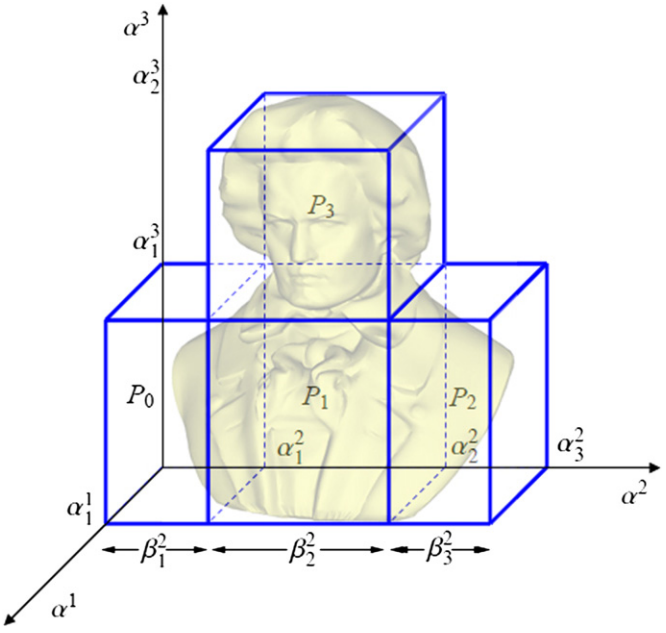

Fig. 4. Definition of polycube coordinates and parameters.

Therefore, supposing a rectangle domain P_k is perpendicular to the axis $u^i,(i=0,1,2)$, we denote the two corresponding segment lengths of the rectangle as $\beta^{i+1}(P_k),\beta^{i+2}(P_k)$, their initial lengths as $\widetilde{\beta}^{i+1}(P_k),\widetilde{\beta}^{i+2}(P_k)$, initial harmonic energies as $\widetilde{H}^{i+1}_{P_k},\widetilde{H}^{i+2}_{P_k}$, and initial area stretching energy as $\widetilde{A}_{P_k}$. These constants $\widetilde{\beta}^{i+1}(P_k)$, $\widetilde{\beta}^{i+2}(P_k),\widetilde{H}^{i+1}_{P_k},\widetilde{H}^{i+2}_{P_k},\widetilde{A}_{P_k}$ are determined by the initial mapping. Then the harmonic energy of all subpatches that are perpendicular to u^i, with respect to the their scalings can be written as

$$E^i_H(\alpha^i_1,\ldots,\alpha^i_{N_i-1},\beta^i_1,\ldots,\beta^i_{N_i-1})=\sum_{P_k}(\beta^{i+1}(P_k))^2\widetilde{C}^{i+1}_k+(\beta^{i+2}(P_k))^2\widetilde{C}^{i+2}_k, \tag{5}$$

where $\widetilde{C}^{i+1}_k$ and $\widetilde{C}^{i+2}_k$ are constants decided by the initial mapping:

$$\widetilde{C}^{i+1}_k=\left(\frac{\widetilde{H}^{i+1}_{P_k}}{(\widetilde{\beta}^{i+1}(P_k))^2}\right),\quad \widetilde{C}^{i+2}_k=\left(\frac{\widetilde{H}^{i+2}_{P_k}}{(\widetilde{\beta}^{i+2}(P_k))^2}\right).$$

Considering all three axes, the global harmonic energy of the polycube mapping is

$$\begin{aligned}E_H(\{\alpha^i_j,\beta^i_j\},\forall i=0,1,2,j=1,\ldots,N_i-1)&=E^1_H(\alpha^0_1,\ldots,\alpha^0_{N_1-1},\beta^0_1,\ldots,\beta^0_{N_1-1})\\&+E^2_H(\alpha^1_1,\ldots,\alpha^1_{N_2-1},\beta^1_1,\ldots,\beta^1_{N_2-1})\\&+E^3_H(\alpha^2_1,\ldots,\alpha^2_{N_3-1},\beta^2_1,\ldots,\beta^2_{N_3-1}).\end{aligned} \tag{6}$$

The area stretching term of the mapping is

$$E_A(\{\alpha^i_j,\beta^i_j\},\forall i=0,1,2,j=1,\ldots,N_i-1)=\sum_{P_k}(\beta^{i+1}(P_k)\beta^{i+2}(P_k))^2\widetilde{C}_k, \tag{7}$$

where $\widetilde{C}_k$ is a constant decided by the initial mapping:

$$\widetilde{C}_k=\left(\frac{\widetilde{A}_{P_k}}{(\widetilde{\beta}^{i+1}(P_k)\widetilde{\beta}^{i+2}(P_k))^2}\right).$$

Finally, we have the entire distortion energy:

$$E(\{\alpha^i_j,\beta^i_j\})=E_H+E_A \tag{8}$$

subject to the constraints:

$$\begin{cases}\alpha^i_1=\beta^i_1,\\ \alpha^i_1+\beta^i_2=\alpha^i_2,\\ \alpha^i_2+\beta^i_3=\alpha^i_3,\\ \vdots\\ \beta^i_j>0,\quad \forall j=1,\ldots,N_i,\ i=0,1,2,\\ \sum_{P_k}\beta^{i+1}(P_k)\beta^{i+2}(P_k)=\widetilde{Area},\end{cases} \tag{9}$$

where $\widetilde{Area}=\sum_{P_k}\widetilde{\beta}^{i_1}(P_k)\widetilde{\beta}^{i_2}(P_k)$; the last equation preserves the total area of the polycube domain. Fig. 5 shows an example of an optimized polycube for the Beethoven model based on the initial polycube mapping. The original polycube (b) is re-scaled to (d); as the grid texture mapping visualized, the distortion of the original mapping (a) reduces when the polycube shape changes (f); as in the zoom-in view (e), the angle distortion is smaller than that in (c).

5.1. Barzilai–Borwein gradient projection optimization algorithm

In order to solve the energy $E(\{\alpha^i_j,\beta^i_j\})$ in Eq. (8) subject to constraints in Eq. (9), we will strictly enforce all the bound and linear constraints, and put the last nonlinear constraint as a penalty term $\lambda(\sum_{P_k}\beta^{i+1}(P_k)\beta^{i+2}(P_k)-\widetilde{Area})^2$ in the objective function. As a result, this optimization problem could be formulated as minimization of a nonlinear function with bound and linear constraints, i.e.,

$$\begin{aligned}&\min\quad E(\mathbf{x})\\&\text{s.t.}\quad \mathbf{x}\in\Omega:=\{\mathbf{x}:A\mathbf{x}=\mathbf{b},\mathbf{b}_l\le\mathbf{x}\le\mathbf{b}_u\},\end{aligned} \tag{10}$$

where $\mathbf{x}\in\mathcal{R}^n$ is the vector of variables $\{\alpha^i_j,\beta^i_j\}$, $n=2(N_1+N_2+N_3)$, $\mathbf{b_l}$ and $\mathbf{b_u}\in\mathcal{R}^n$ are the bound constraints, and A is an m by n matrix with $\mathbf{b}\in\mathcal{R}^m$ denoting the linear constraints. Although the objective function is continuously differentiable, the dimension n of our reformulated problem generally can be large, and the

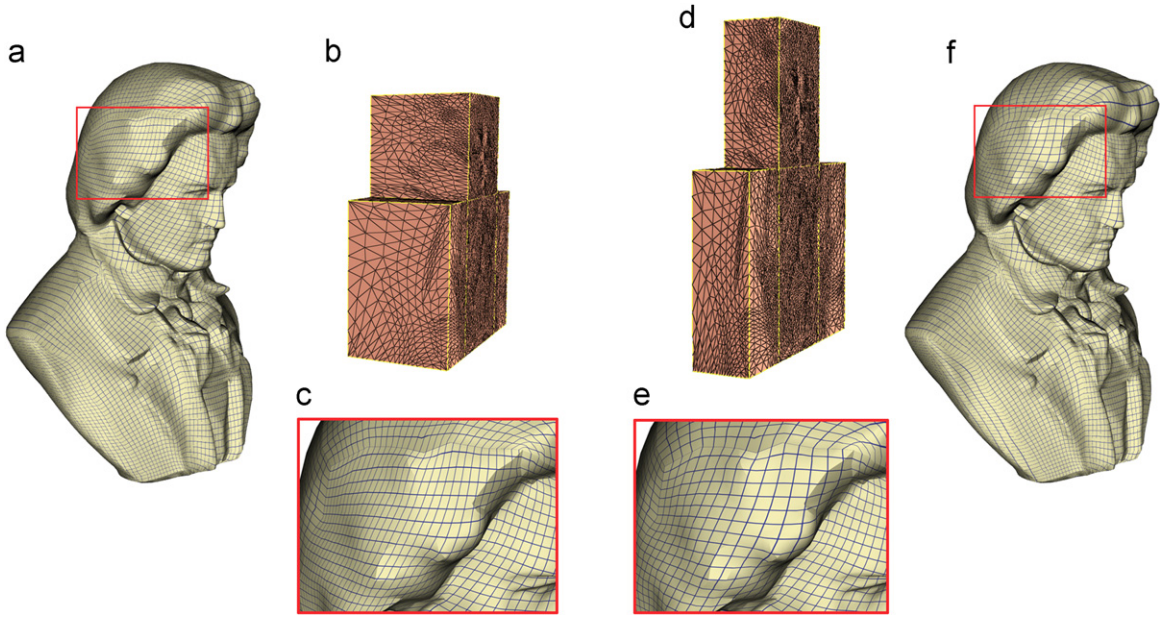

Fig. 5. Polycube domain optimization. (a)–(c) show the initial polycube domain and mapping. (d)–(f) show the optimized polycube domains. Note the improvement of the checkerboard texture mapping between (c) and (e).

explicit computation of the Hessian is difficult. Hence, first order method, which only requires gradient information, is preferred. To solve (10), we develop the following nonmonotone gradient projection algorithm, which is also an iterative algorithm: given the starting $\mathbf{x}_0$, our algorithm takes the following iterations:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k, \tag{11}$$

where k is the iteration number, α_k is a stepsize and $\mathbf{d}_k$ is the searching direction defined as

$$\mathbf{d}_k = P_\Omega\left(\mathbf{x}_k - \frac{1}{\lambda_k^{BB}}\mathbf{g}_k\right) - \mathbf{x}_k.$$

Here, P_Ω is the projection on the feasible set Ω, $\mathbf{g}_k = \nabla f(\mathbf{x}_k)$ and λ_k^{BB}, $k \geq 1$ is the so-called Barzilai–Borwein [2] stepsize parameter generated by satisfying a quasi-Newton property, i.e.,

$$\lambda_k^{BB} = \arg\min_{\lambda \geq \lambda_0} \|\mathbf{\Lambda}(\lambda)\mathbf{s}_{k-1} - \mathbf{y}_{k-1}\|_2, \tag{12}$$

where $\mathbf{s}_{k-1} = \mathbf{x}_k - \mathbf{x}_{k-1}$, $\mathbf{y}_{k-1} = \mathbf{g}_k - \mathbf{g}_{k-1}$, $\mathbf{\Lambda}(\lambda_k) = \lambda_k \mathbf{I}$, and λ_0 is a positive constant. Hence, the proposed λ_k^{BB}, when $k \geq 1$, obtained from (12), is

$$\lambda_k^{BB} = \max\left\{\frac{\mathbf{s}_{k-1}^T \mathbf{y}_{k-1}}{\mathbf{s}_{k-1}^T \mathbf{s}_{k-1}}, \lambda_0\right\}, \tag{13}$$

and λ_0^{BB} can be arbitrarily defined as a positive number and we set $\lambda_0^{BB} = \|\mathbf{g}(\mathbf{x}_0)\|_\infty$ and $\lambda_0 = 10^{-10}$ in practice. This BB initial stepsize (13) has been extensively studied recently and been shown to perform much better than steepest descent type gradient projection methods [3,4]. However, to maintain the efficiency, the stepsize α_k in (11) must be obtained by a nonmonotone line search. In our experiments, we use the nonmonotone line search developed in [5,16].

6. Optimizing polycube mapping

In Section 5, we fix the corner point mapping $f(\mathcal{V}_{CS}) \to \mathcal{V}_{CP}$ to optimize the shape of polycube domain. We further reduce the mapping distortion by moving vertices $\mathcal{V}_{CS}$ (without ambiguity, we also call them corner points) over S. Any 2D manifold S can be parameterized to an atlas $\Omega = \{\Omega_i\}$, and locally any point on S: $(x^1, x^2, x^3) \in S$ can be represented as a 2D coordinate (r^1, r^2) on a local planar chart. We construct local parameterization $g_i : S_i \to \Omega_i$ by mapping the C-ring neighboring regions (in our experiments, we set $C=20$) of each initial corner point $\in \mathcal{V}_{CS}$ to a unit disc Ω_i. Any neighboring points on the domain Ω_i are continuously parameterized.

Let N be the number of the corner points $N = |\mathcal{V}_{CP}|$. The optimization will be conducted on all charts $\{\Omega_1, \ldots, \Omega_N\}$ simultaneously by searching the optimal N corner points, represented as coordinates $(\{r_1, r_2, \ldots, r_{2N-1}, r_{2N}\})$, where (r_{2k-1}, r_{2k}) corresponds to (r^1, r^2) on chart Ω_k.

This problem is formulated as minimizing the distortion energy E of the map f decided by the corner maps:

$$E(r_1, r_2, \ldots, r_{2N}) = H^1(f) + H^2(f) + H^3(f) + A(f), \tag{14}$$

the harmonic energies and area stretching of function f are defined following Eqs. (2) and (3).

For polycube mapping with N corner points, the dimension of this optimization problem is $2N$. f is determined by these $2N$ parameters, and can be efficiently computed (Section 6.1), but since we need to retrace the shortest paths as the subpatch boundaries, we do not have the closed form for f or its derivative. Therefore, we use a derivative-free optimizer (Section 6.2) to solve this problem.

As indicated in Algorithm 1, we iteratively perform domain optimization (Section 5) and mapping optimization (this section) until the polycube domain does not change. Despite the optimization of both the domain shape and the corner mapping, the angle distortion near the subregion boundary (e.g., polycube corners, edges) can be large due to the usage of harmonic mapping with fixed boundary. We perform a smoothing process to further reduce the distortion. Smooth transition functions [9] can be easily computed between adjacent polycube faces, then parameterization/smoothing can be computed on a flattened domain covering this boundary region. We adopt the smoothing algorithm of [20] to refine the map near polycube corner/edge regions.

Fig. 6 illustrates an iteration of domain mapping optimization on a Beethoven model. Corners in (a) are adjusted to new positions (f). Meanwhile, the mapping distortion energy reduces, which can also be visualized in the zoom-in regions (d,e vs b,c). If we perform an aforementioned smoothing, the distortion near the boundary region can be further reduced (f, g).

Fig. 7 shows an iteration of our polycube map optimization on the horse model; the initial horse mapping (a) on a polycube with 60 corner points is optimized; the resultant mapping (b) has smaller angular and area distortion.

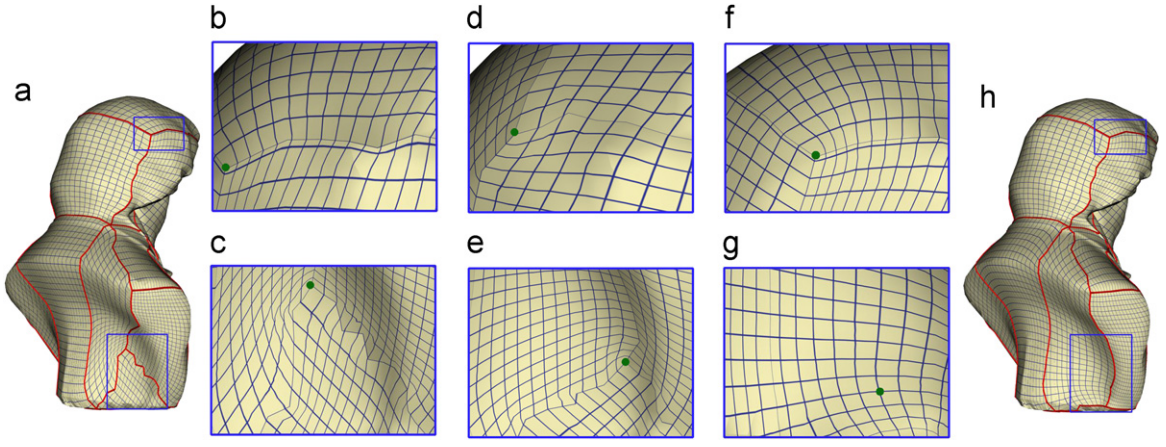

Fig. 6. Polycube mapping optimization. (a) is the model before mapping optimization. (b,c) zoom in to show the distortion before this step. (d,e) illustrate the distortion after mapping optimization. (g,f) show distortion after the smoothing postprocess. (h) is model after smoothing. The corner points are shown in green. With the smoothing, distortion and discontinuity across subregion boundaries significantly reduces. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

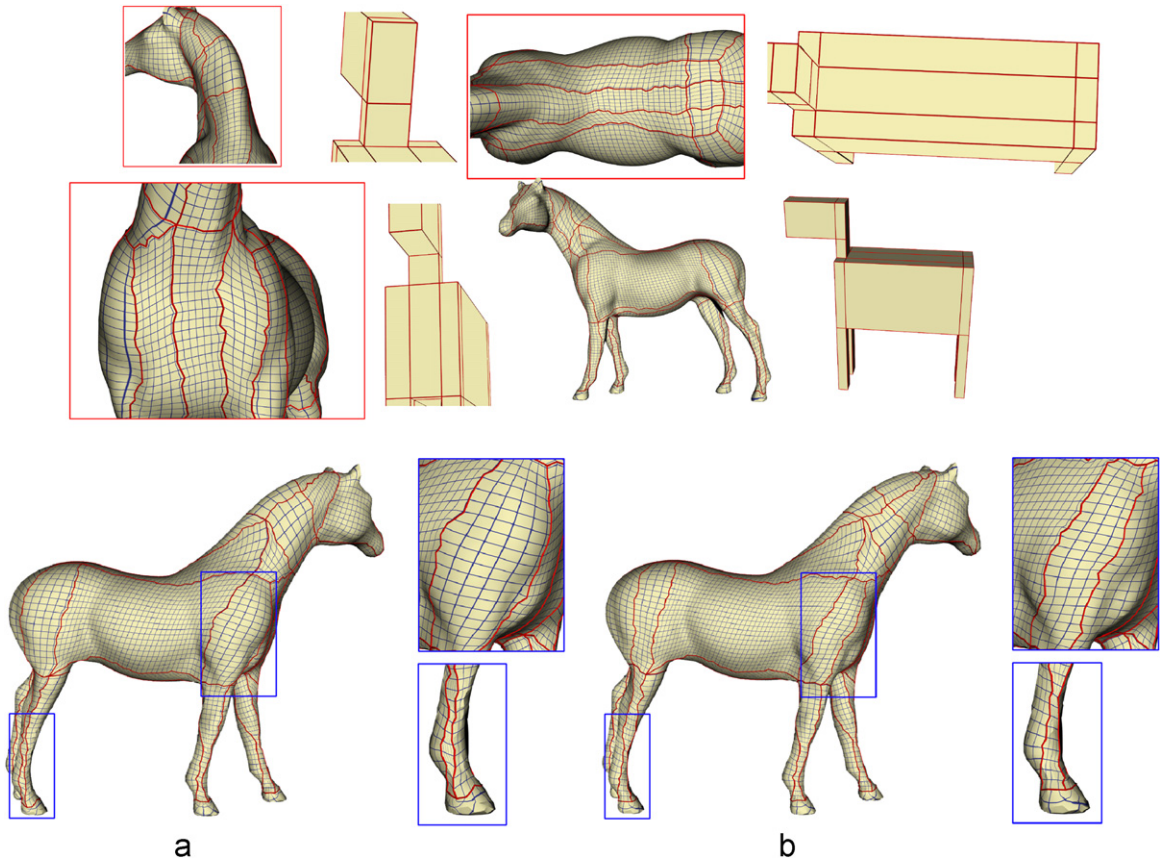

Fig. 7. Mapping optimization of the Horse model on a polycube (upper row) with 60 corner points. The lower row shows the moving of corner points: (a) before optimization and (b) after optimization.

6.1. Efficient mapping recomputation

The typical computation for harmonic surface mapping on each rectangle subpatch involves solving two systems of linear equations. This can be time consuming when we need to recompute it and re-evaluate its distortion in every step during the optimization. Since the boundary condition of the mapping always changes gradually, we can utilize a more efficient linear equation updating algorithm CHOLMOD [6] to accelerate the mapping recomputation.

Mapping on each subpatch is harmonic, so the coefficient matrix is sparse, symmetric and positive definite. This special property makes it feasible to utilize Cholesky decomposition to solve and update the linear systems very quickly. Initially, we precompute the shortest paths between all pairs of vertices using the Floyd–Warshall algorithm and store predecessor matrices on shortest paths. This takes $O(n^3)$ preprocessing time, where n is the number of vertices. During each iteration, when corner points are replaced by some of their neighboring points, between each pair of corners, we retrace corresponding shortest paths in $O(k)$ time where k is the number of vertices on this path. The coefficient matrix only changes slightly (a few rows and columns proportional to the number of mutable boundary conditions due to the change of corner points). This infers an efficient solution-update algorithm. Davis and Hager [6] proposed an approach of dynamic supernodal sparse Cholesky update and downdate, which produces a solution for the newly update linear system without repeatedly computing the coefficient matrix and solving the system. After an initial Cholesky decomposition at a cost of $O(n^3)$, the decomposition can be updated in $O(N)$, where N is the number of changed entries in Cholesky factor, which is typically much smaller than the size of the mesh, leading to efficient harmonic mapping update. The similar approach was introduced to graphics and shape modeling [37] for dynamically updating harmonic fields design.

With this efficient mapping update technique, we can re-evaluate the objective function for a given new planar coordinates for corner points on S. Since the parameterization (and therefore the corner selection) is continuous, we dynamically split each corresponding triangle (where each parametric corner point locates) into three and update the accumulated energy accordingly.

6.2. Derivative-free optimization algorithm

The objective function (14) can be reformulated in the following format:

$$\min \Phi(\mathbf{x}) = \sum_{i}^{m} \mathrm{sgn}(i) f_i^2(\mathbf{x}) \quad \text{s.t. } \mathbf{b_l} \le \mathbf{x} \le \mathbf{b_u}, \tag{15}$$

where $\mathbf{x} \in \mathcal{R}^n$, $\mathbf{b}_l$ and $\mathbf{b}_u \in \mathcal{R}^n$ are the bound constraints, and $\mathrm{sgn}(i) = \pm 1$ is the sign in front of the squares of f_i, $i = 1, \ldots, m$. The main difficulty of solving this problem is that the explicit derivatives are not available. We develop a trust region-based derivative-free algorithm in spirit similar to the approach proposed in [38]. Our algorithm does not require the derivative information of the objective function, nor does it explicitly approximate the derivative. Instead, at each iteration it builds a local quadratic model of the objective function by multivariate interpolation in combination with trust region techniques. More specifically, at each iteration, the algorithm adaptively chooses a set of interpolation points $\mathbf{Y}_k$, with $(n+1) \le |\mathbf{Y}_k| \le (n+1)(n+2)/2$,

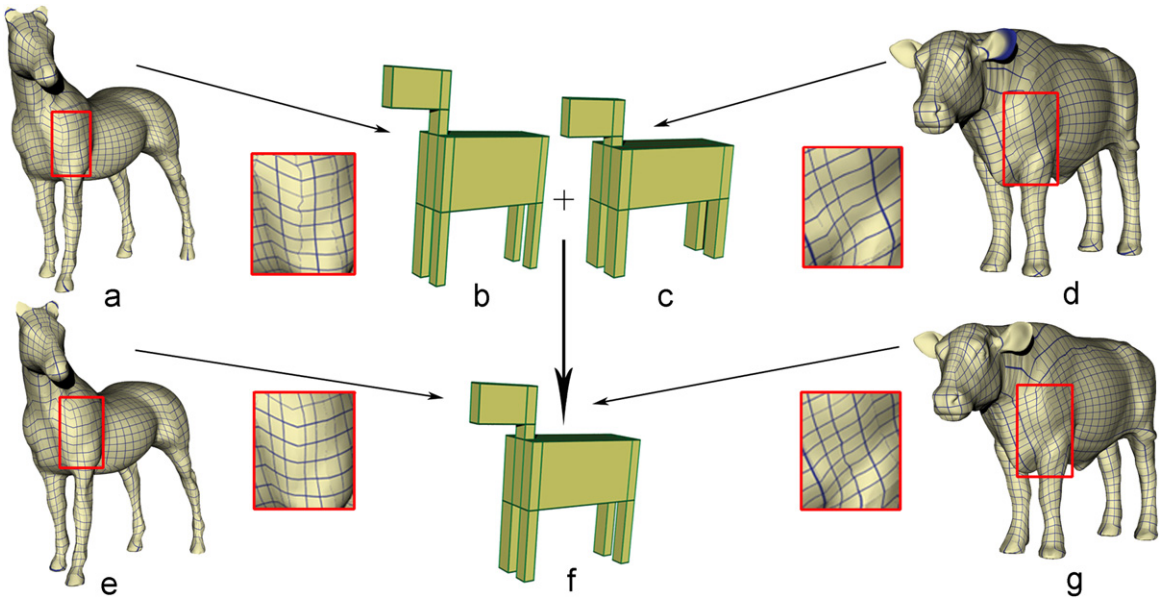

Fig. 8. Common polycube mapping for multiple models. Initial polycube maps of the horse and cow are as (a) and (d); individually optimal polycube domains are shown in (b) and (c); the common optimal polycube domain is shown in (f); and the final common optimal polycube mapping of both models are as (e) and (g). Note: the common polycube balances both individually optimal polycubes, see the neck region.

where k is the iteration number and $|\mathbf{Y}_k|$ is the cardinality of $\mathbf{Y}_k$. Our algorithm takes the following major steps:

Step 0 (*Initialization*). Set up initial starting guess $\mathbf{x}_0$, trust region radius Δ_0 and sampling points $\mathbf{Y}_0$. Build initial trust region model on $\mathbf{Y}_0$ and set $k=0$.

Step 1 (*Criticality step*). Choose a base point $\mathbf{y}_k \in \mathbf{Y}_k$ and calculate the gradient of our model. If the gradient is sufficiently small, stop. Otherwise, make sure the model is well-posed [38] in a trust region with radius proportional to the norm of model gradient.

Step 2 (*Step calculation*). Solve the following trust region subproblem:

$$\min \phi_k(\mathbf{d}) \quad \text{s.t. } \|\mathbf{d}\| \le \Delta_k,\ \mathbf{l} \le \mathbf{x}_k + \mathbf{d} \le \mathbf{u}, \tag{16}$$

where $\phi_k(\mathbf{d})$ is a local quadratic model of $\Phi(\mathbf{x})$ in a trust region with radius Δ_k. Here, $\|\cdot\|$ is the 2-norm.

Step 3 (*Acceptance of the trial step*). Compute the ratio of actual and predicted function reduction

$$r_k = \frac{\Phi(\mathbf{x}_k) - \Phi(\mathbf{x}_k + \mathbf{d}_k)}{\phi_k(\mathbf{0}) - \phi_k(\mathbf{d}_k)},$$

where $\mathbf{d}_k$ is the minimizer of (16). If $r_k > 0$, then $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}_k$; otherwise, $\mathbf{x}_{k+1} = \mathbf{x}_k$.

Step 4 (*Trust region radius update*). Update trust region radius by

$$\Delta_{k+1} = \begin{cases} \frac{1}{2}\|\mathbf{d}_k\| & \text{if } r_k < 0.1, \\ \max\{\frac{1}{2}\Delta_k, \|\mathbf{d}_k\|\} & \text{if } 0.1 \le r_k < 0.7, \\ \max\{\Delta_k, 2\|\mathbf{d}_k\|\} & \text{if } r_k \ge 0.7. \end{cases}$$

If $r_k \ge 0.1$, form $\mathbf{Y}_{k+1}$ from $\mathbf{Y}_k$ by merging new point $\mathbf{x}_{k+1}$. Set $k = k+1$, go to Step 1.

Step 5 (*Model improvement*). This step applies only when $r_k < 0.1$. In this case, before shrinking the trust region radius, make sure the model is well-posed [38] in the current trust region. Set $k = k+1$, go to Step 1.

One critical advantage of this algorithm is using the least Frobenius norm updating strategy [29] to update the quadratic model (16). Hence, to build our quadratic model, we only need $\mathcal{O}(n)$ (in our experiments, $2n+1$) function evaluations, while normally $(n+1)(n+2)/2 = \mathcal{O}(n^2)$ number of valuations are required for building a fully quadratic model (Note, $(n+1)(n+2)/2$ could be much bigger than $2n+1$ for relatively large n.) In addition, at each iteration, only one new function evaluation is required to update the local quadratic model.

Therefore, our approach is usually more efficient [26,38] than other widely used strategies in derivative-free optimization, such as using finite-difference to approximate derivatives [25] or some direct search methods [1]. Global convergence of the algorithm as well as the good local geometry of the set of interpolation points are guaranteed by trust region techniques [38,29].

7. Polycube mapping for multiple objects

We also demonstrate an application of our polycube mapping framework in multiple objects mapping. Polycube can be used as a canonical base domain for multiple objects (preferably, these objects have the same topology and similar geometry). Our framework can be used to generate such a common regular domain, and multiple objects are parameterized onto this single polycube with low distortion. Multiple shapes can be analyzed, processed, and integrated over this single domain. Supposing we have a set of models $\{S_1, \ldots, S_n\}$ to be integrated. We construct a common polycube P using S_1. We also compute initial mapping f_i between P and each $S_i, i = 1, \ldots, n$. Then simultaneously, we optimize P and the mapping $f_i : S_i \to P$ using the above proposed framework. The final polycube domain P is the one that minimizes the total distortion of multiple polycube parameterization $E = \sum_i E(f_i)$. The final polycube is an optimal domain for all these models. Inter-surface mapping between two models S_i and S_j can be composed and optimized over this domain as $f_{i,j} : S_i \to S_j = f_j^{-1} \circ f_i$. We visualize our optimal polycube and the mapping results using inter-object morphing by linearly interpolating them over the common polycube domain.

Specifically, we construct initial polycube P for S_1 and use projection to determine corner points mapping. However, this simple projection approach does not work well when we map P to other models $S_2, S_3, \ldots, S_n$, especially when S_i is not geometrically

similar to P. Especially for this situation (when we want to map a surface to a dissimilar polycube), we compute the initial polycube mapping in the following more robust way (Note that any other suitable polycube mapping approach can also be used to generate initial f_i). We partition P and each S_i consistently (i.e., the segmentation of P and S_i has the isomorphic dual graph); then compute the mapping $f_i : S_i \to P$ by merging all individual subregion mappings. Such an approach based on canonical pants decomposition is introduced in [22]. We briefly recap the basic idea, and refer readers to [22] for details. The pants patch is a genus-0 surface with three boundaries. Any surface (except for a few trivial cases) can be decomposed into a set of pants patches, including g handle patches and a base patch, where g is the genus. The base patch is then further iteratively partitioned into a set of pants patches. Finally, every pants patch is decomposed into two subpatches, each of which can be parameterized on a regular planar hexagon. Therefore, the global surface mapping between two objects is composed of parameterizations of subpatches on these hexagonal domains. This approach can easily and robustly handle the surface mapping between two objects with arbitrary topology [41] and feature points, therefore it is suitable here for generating initial mapping $f_i : S_i \to P$. Fig. 8 shows an example of using the above approach to construct optimal common polycube for the horse and the cow. Individually optimal polycubes for the horse and cow are shown in (b) and (c), and initial polycube maps are visualized in (a) and (d); the optimal common polycube is shown in (f). Specifically, a compromise can be seen in the neck region. The final common polycube mappings are visualized in (e) and (g).

Table 1
Comparisons of different polycube mapping methods. *PC Constr.*, *Opt. PC*, *Sing. Control*, *Common PC* indicate whether polycube construction can be automatic, whether polycube shape is optimal, whether polycube complexity can be controlled by the given restriction on singularity number, and whether it can be used to construct a canonical domain for multiple objects, respectively.

Methods	PC Constr.	Opt. PC	Sing. Control	Common PC
Tarini [33]	Manual	No	Manual	No
Wang [34]	Manual	No	Manual	No
Wang [35]	Manual	No	Manual	No
Lin [24]	Auto.	No	No	No
He [18]	Auto.	No	Yes	No
Ours	Auto.	Yes	Yes	Yes

8. Experimental results

We compare the property of our polycube mapping framework with existing methods and list them in Table 1. Our method generates the optimal polycube within the same topological class, and the complexity of the polycube is flexibly bounded by the given number of singularities. We test our optimization framework on a few 3D shapes. Fig. 9 shows the optimization on Bimba and Max-planck. The texture-mapped rectangular grids become closer to squares, indicating the reducing of angle distortion.

Fig. 10 shows a common polycube parameterization for multiple objects. We parameterize the horse, cow, and goat onto an optimized common polycube domain. (a–c) visualize the geometry

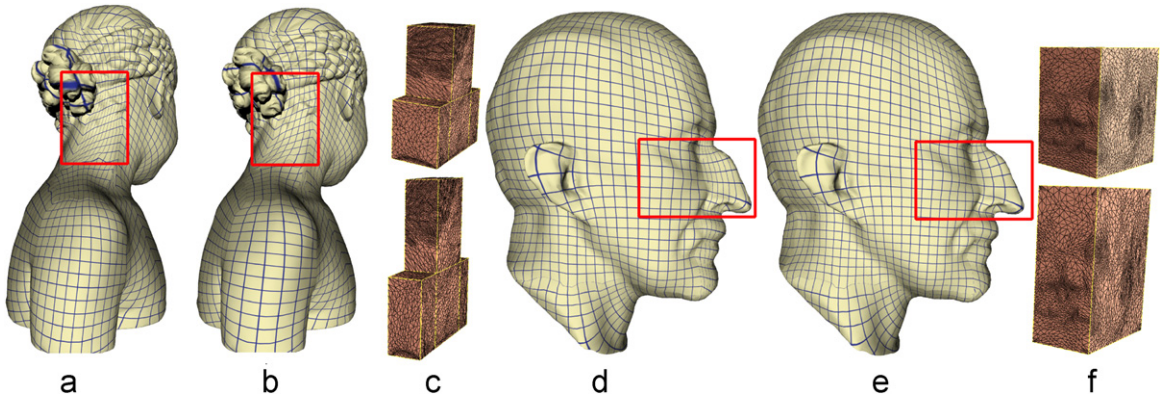

Fig. 9. Polycube mapping of Bimba and Max-Planck. (a,d) initial mapping, (b,e) optimized mapping. The texture mappings of grids show the reduction of angle distortions after the optimization. (c,f) initial polycube (in upper row) and optimized polycube (in lower row) domains.

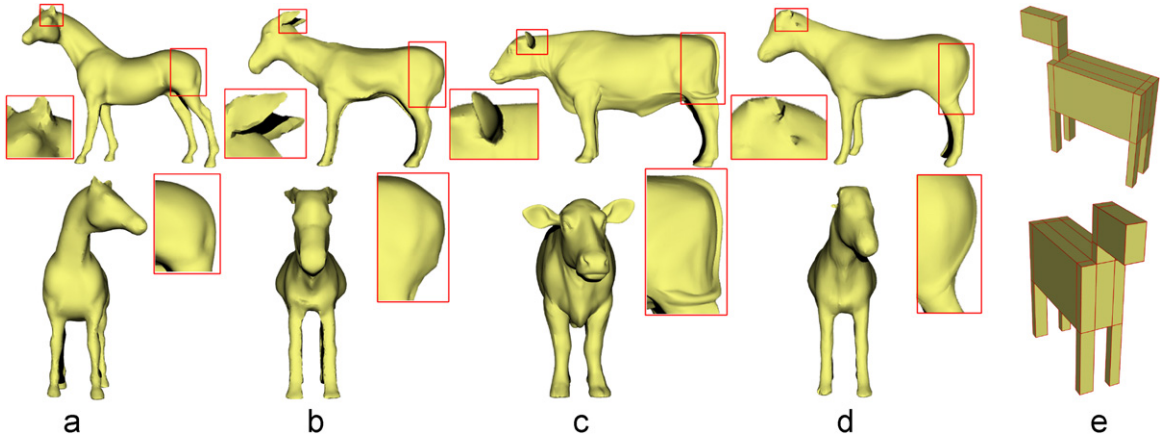

Fig. 10. Integration of multiple objects over a common polycube domain. The horse (a), goat (b), and cow (c) are blended in this polycube domain. Features from the original models can still be seen in the interpolated shape (e.g., the mouth and neck of the horse, ears of the goat, and the tail of the cow).

Table 2
Runtime table: #Δ (number of triangles); #C number of corner points, ε^0_{angle} and ε^0_{area} are angle and area distortions before optimization; ε_{angle} and ε_{area} are distortions after optimization; T_1 and T_2 is the execution time for domain optimization and mapping optimization (in seconds).

Models	#Δ	#C	ε^0_{angle}	ε^0_{area}	ε_{angle}	ε_{area}	T1	T2
Isis	5K	8	1.261	1.429	1.134	1.385	0.52	112
Beethoven	21K	20	1.387	1.563	1.215	1.236	7.74	504
Max-Planck	10K	8	1.104	1.477	1.060	1.395	1.36	33
Bimba	30K	20	1.292	1.243	1.283	1.209	10.62	744
Horse	16K	60	1.352	1.302	1.258	1.229	11.72	1842
Cow	39K	60	1.198	1.210	1.191	1.161	21.21	2898
Goat	21K	60	1.359	1.304	1.241	1.190	10.83	2032

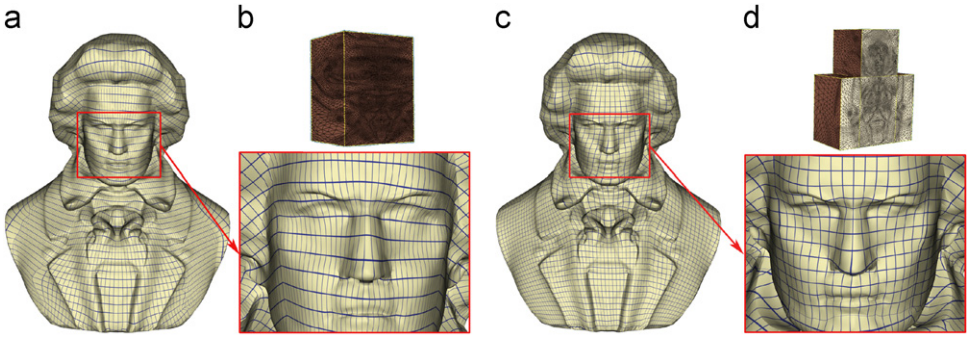

Fig. 11. Different initial corner budgets. With increase of the initial budget (from 8 to 20), the mapping quality is improved (from a,b to c,d).

Table 3
Testing different weighting on the area-stretching term (α in Eq. (4)), on the polycube-Beethoven mapping. ε_{angle} and ε_{area} are the corresponding angle and area distortion.

α	0.1	0.5	1.0	1.5
ε_{angle}	1.219	1.235	1.253	1.264
ε_{area}	1.380	1.316	1.292	1.281

represented on the polycube parameterization (using the connectivity of the polycube), then we can easily interpolate them and generate a "mixed creature". (d) shows an interpolated shape with 20%, horse, 50%, goat, and 30%, cow. Features of horse, goat, and cow can be seen on the final interpolated shape.

The quality of polycube mapping can be measured by area distortion ε_{area} and angle distortion ε_{angle} [7]:

$$\varepsilon_{area}(T) = \frac{area(\Delta_{M'}(T))}{area(\Delta_M(T))} + \frac{area(\Delta_M(T))}{area(\Delta_{M'}(T))},$$

$$\varepsilon_{angle}(T) = \frac{\cot\alpha|a^2| + \cot\beta|b^2| + \cot\gamma|c^2|}{2area(\Delta_M(T))}.$$

The closer the values of ε_{area} and ε_{angle} is to 1, the better the quality of polycube mapping we get. The statistics and performance of our test cases are reported in Table 2.

Intuitively, the more complicated the polycube domain is used, the more freedom we have to optimize its shape. And generally when the polycube is closer to the original model, we can get a less distorted/stretched polycube mapping. Fig. 11 illustrates an example on the Beethoven model. When only one cube is used as the parameterization domain, the distortion is larger (a, b), compared with the mapping constructed on a more complicated polycube domain (c, d). On the other hand, a more complicated polycube domain indicates more corner points (singularities) [34] and potentially more-distorted parameterization across subregion boundaries.

We also adjust the weighting factor α in Eq. (4) to see different mapping results. Table 3 shows the different angle and area distortions under different settings. $\alpha = 1.0$ was used when we perform our other experiments. Fig. 12 illustrates this mapping result. When the area term is emphasized, a more uniform but less conformal mapping is obtained (a, b); when α is small, the angle distortion is reduced (c, d).

9. Conclusion

We present an interactive optimization framework to solve the optimal polycube mapping problem. Because directly solving optimal polycube domain and mapping together is too expensive, we iteratively optimize polycube domain shape and polycube mapping separatively, to make full use of the available partial derivative information of the objective function. We develop an efficient nonlinear optimization algorithm with linear bound constraints for the first subproblem. For the second subproblem, we develop an efficient derivative-free solver, making use of the summation-of-square structure of the objective function; and we develop a fast mapping re-computation algorithm to accelerate the evaluation in the optimization process. The polycube mapping framework has been demonstrated effective in several experiments, and can be used to construct common polycube domains for multiple objects.

Allowing the alignment of feature points in the polycube mapping can benefit many graphics applications such as morphing and registration. However, this is challenging and has not been well discussed/solved in existing polycube mapping literature. Within our current framework, on a subpatch, directly enforcing the harmonic mapping to map an interior feature point to a specific position on the polycube domain may cause local flip-over around the feature point. One possible approach is to simply add feature alignment as a soft constraint in the mapping optimization step, such that feature matching errors are penalized like the angle-distortion and area-distortion terms. To enforce a

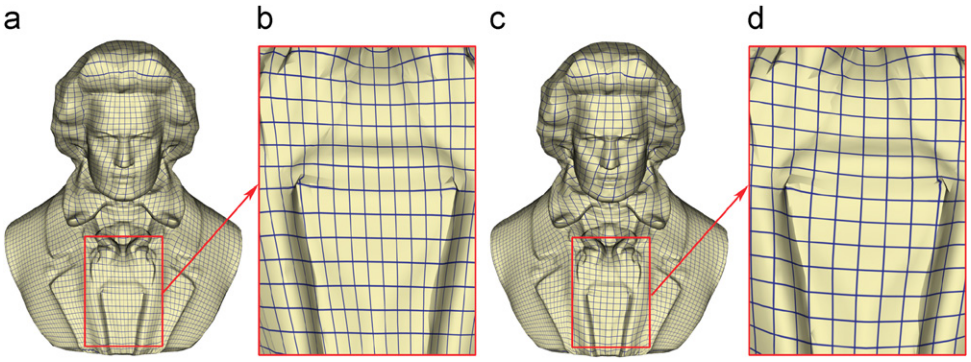

Fig. 12. Different weighting factors. (a,b) Area-stretching term $\alpha = 1000$, (c,d) $\alpha = 0.01$.

hard constraint on feature matching, additional domain partitioning to make the features on the subpatch boundary can be another solution. We will study this problem in the near future.

Acknowledgments

This work is supported by Louisiana Board of Regents Research Competitiveness Subprogram (RCS) LEQSF(2009-12)-RD-A-06, PFund: NSF(2009)-PFund-133, LSU Faculty Research Grant 2010, and NSF DMS-1016204. We also thank AIM@Shape and Stanford Shape Repository for their datasets.

References

[1] Audet C, Dennis Jr JE. A pattern search filter method for nonlinear programming without derivatives. SIAM Journal on Optimization 2004;14:980–1010.
[2] Barzilai J, Borwein JM. Two-point step size gradient methods. IMA Journal of Numerical Analysis 1988;8(1):141–8.
[3] Birgin EG, Jos MM, Raydan M. Nonmonotone spectral projected gradient methods on convex sets. SIAM Journal on Optimization 2000:1196–211.
[4] Dai Y-H, Hager WW, Schittkowski K, Zhang H. The cyclic Barzilai–Borwein method for unconstrained optimization. IMA Journal of Numerical Analysis 2006;26:604–27.
[5] Dai Y-H, Zhang H. Adaptive two-point stepsize gradient algorithm. Numerical Algorithms 2001;27:377–85.
[6] Davis TA, Hager WW. Dynamic supernodes in sparse Cholesky update/downdate and triangular solves. ACM Transactions on Mathematical Software 2009;35:27:1–23.
[7] Degener P, Meseth J, Klein R. An adaptable surface parameterization method. In: IMR; 2003. p. 201–13.
[8] Desbrun M, Meyer M, Alliez P. Intrinsic parameterizations of surface meshes. Computer Graphics Forum 2002;21(3):209–18.
[9] Dong S, Bremer P-T, Garland M, Pascucci V, Hart JC. Spectral surface quadrangulation. ACM Transactions on Graphics 2006;25:1057–66.
[10] Eck M, DeRose T, Duchamp T, Hoppe H, Lounsbery M, Stuetzle W. Multiresolution analysis of arbitrary meshes. In: SIGGRAPH; 1995. p. 173–82.
[11] Fan Z, Jin X, Feng J, Sun H. Mesh morphing using polycube-based cross-parameterization: animating geometrical models. Computer Animation and Virtual Worlds 2005;16(3–4):499–508.
[12] Floater MS. Mean value coordinates. Computer Aided Geometric Design 2003;20(1):19–27.
[13] Floater MS, Hormann K. Surface parameterization: a tutorial and survey. Advances in Multiresolution for Geometric Modelling 2005:157–86.
[14] Gu X, Yau S-T. Global conformal surface parameterization. In: Proceedings of the symposium on geometry processing; 2003. p. 127–37.
[15] Guo X, Li X, Bao Y, Gu X, Qin H. Meshless thin-shell simulation based on global conformal parameterization. IEEE Transactions on Visualization and Computer Graphics 2006;12(3):375–85.
[16] Hager WW, Zhang H. A new active set algorithm for box constrained optimization. SIAM Journal on Optimization 2006;17:526–57.
[17] Han S, Xia J, He Y. Hexahedral shell mesh construction via volumetric polycube map. In: SPM. SPM '10. New York, NY, USA: ACM; 2010. p. 127–36.
[18] He Y, Wang H, Fu C-W, Qin H. A divide-and-conquer approach for automatic polycube map construction. Computers & Graphics 2009;33(3):369–80.
[19] Hormann K, Lévy B, Sheffer A. Mesh parameterization: theory and practice. In: ACM Siggraph 2007 course, vol. 11; 2007. p. 1–87.
[20] Kraevoy V, Sheffer A. Cross-parameterization and compatible remeshing of 3D models. ACM Transactions on Graphics 2004;23(3):861–9.
[21] Li H, Lo K-Y, Leung M-K, Fu C-W. Dual poisson-disk tiling: an efficient method for distributing features on arbitrary surfaces. IEEE Transactions on Visualization and Computer Graphics 2008;14(5):982–98.
[22] Li X, Gu X, Qin H. Surface mapping using consistent pants decomposition. IEEE Transactions on Visualization and Computer Graphics 2009;15(4):558–71.
[23] Li X, Guo X, Wang H, He Y, Gu X, Qin H. Harmonic volumetric mapping for solid modeling applications. In: SPM. New York, NY, USA; ACM: 2007. p. 109–20.
[24] Lin J, Jin X, Fan Z, Wang CCL. Automatic polycube-maps. In: GMP; 2008. p. 3–16.
[25] Moré J. The Levenberg–Marquardt algorithm: implementation and theory. In: Watson G, editor. Numerical analysis. Lecture notes in mathematics, vol. 630. Berlin, Heidelberg: Springer; 1978. p. 105–16.
[26] Moré JJ, Wild SW. Benchmarking derivative-free optimization algorithms. SIAM Journal on Optimization 2009;20(1):172–91.
[27] Pietroni N, Tarini M, Cignoni P. Almost isometric mesh parameterization through abstract domains. IEEE Transactions on Visualization and Computer Graphics 2010;16(4):621–35.
[28] Pinkall U, Polthier K. Computing discrete minimal surfaces and their conjugates. Experimental Mathematics 1993;2(1):15–36.
[29] Powell MJD. Least Frobenius norm updating of quadratic models that satisfy interpolation conditions. Mathematical Programming 2004;100:183–215.
[30] Praun E, Sweldens W, Schröder P. Consistent mesh parameterizations. In: ACM SIGGRAPH; 2001. p. 179–84.
[31] Ray N, Li W, Lévy B, Sheffer A, Alliez P. Periodic global parameterization. ACM Transactions on Graphics 2006;25(4):1460–85.
[32] Sheffer A, Praun E, Rose K. Mesh parameterization methods and their applications. Foundations and Trends in Computer Graphics and Vision 2006;2(2):105–71.
[33] Tarini M, Hormann K, Cignoni P, Montani C. Polycube-maps. In: ACM SIGGRAPH; 2004. p. 853–60.
[34] Wang H, He Y, Li X, Gu X, Qin H. Polycube splines. In: SPM. New York, NY, USA: ACM; 2007. p. 241–51.
[35] Wang H, Jin M, He Y, Gu X, Qin H. User-controllable polycube map for manifold spline construction. In: SPM. New York, NY, USA: ACM; 2008. p. 397–404.
[36] Xia J, Garcia I, He Y, Xin S-Q, Patow G. Editable polycube map for GPU-based subdivision surfaces. In: Symposium on interactive 3D graphics and games; 2011. p. 151–8.
[37] Xu K, Zhang H, Cohen-Or D, Xiong Y. Technical section: dynamic harmonic fields for surface processing. Computers & Graphics 2009;33:391–8.
[38] Zhang H, Conn AR, Scheinberg K. A derivative-free algorithm for the least-squares minimization. SIAM Journal on Optimization 2010;20:3555–76.
[39] Li X, Guo X, Wang H, He Y, Gu X, Qin H. Meshless harmonic volumetric mapping using fundamental solution methods. IEEE Transactions on Automation Science and Engineering (TASE) 2009;6(3):409–22.
[40] Li X, Xu X, Wan S, Yin Z, Yu W. Feature-aligned harmonic volumetric mapping using MFS. Computers & Graphics 2010;34(3):242–51.
[41] Li X, Bao Y, Guo X, Jin M, Gu X, Qin H. Globally optimal surface mapping for shapes of arbitrary topology. IEEE Transactions on Visualization and Computer Graphics (TVCG) 2008;14(4):805–19.