A Fast Numerical Solver for Local Barycentric Coordinates

Jiong Tao[a], Bailin Deng[b], Juyong Zhang[a,*]

[a]*School of Mathematical Sciences, University of Science and Technology of China*
[b]*School of Computer Science and Informatics, Cardiff University*

Abstract

The local barycentric coordinates (LBC), proposed in Zhang et al. (2014), demonstrate good locality and can be used for local control on function value interpolation and shape deformation. However, it has no closed-form expression and must be computed by solving an optimization problem, which can be time-consuming especially for high-resolution models. In this paper, we propose a new technique to compute LBC efficiently. The new solver is developed based on two key insights. First, we prove that the non-negativity constraints in the original LBC formulation is not necessary, and can be removed without affecting the solution of the optimization problem. Furthermore, the removal of this constraint allows us to reformulate the computation of LBC as a convex constrained optimization for its gradients, followed by a fast integration to recover the coordinate values. The reformulated gradient optimization problem can be solved using ADMM, where each step is trivially parallelizable and does not involve global linear system solving, making it much more scalable and efficient than the original LBC solver. Numerical experiments verify the effectiveness of our technique on a large variety of models.

Keywords: Barycentric Coordinates, Locality, Local Extrema, Integrability, Parallel

1. Introduction

Generalized barycentric coordinates (GBC) Hormann and Sukumar (2017) provide an intuitive and simple way to interpolate values prescribed at the vertices (often referred to as control points) of a polytope. Given a polytope, there are often multiple ways to define its associated GBC. In the past, various GBC schemes have been developed for different tasks in computer graphics as well as other disciplines, such as mesh parameterization Floater (1997, 2003), shape deformation Ju et al. (2005); Lipman et al. (2007); Ju et al. (2008); Weber et al. (2009), image composition and warping Hormann and Floater (2006); Farbman et al. (2009), interior distance measurement Rustamov et al. (2009), and finite element method Wachspress (1975); Sukumar and Malsch (2006). Most of the GBCs developed so far are *global* in the sense that each point within the domain is influenced by many control points. This can lead to some undesirable consequence in practice, including lack of local control for shape deformation, and excessive storage for the coordinates on densely discretized domains. To address these issues, some recent studies have developed GBC schemes with *local* support for the coordinate functions Landreneau and Schaefer (2010); Zhang et al. (2014); Anisimov et al. (2017). In particular, the Local Barycentric Coordinates (LBC) proposed in Zhang et al. (2014) are computed by minimizing the total variation of the coordinate functions to induce their locality, subject to a set of constraints that ensure desired properties such as partition of unity, reproduction, and non-negativity.

Although the optimization problem for LBC is convex and thus its global minimum can be effectively computed, its numerical solving can be time-consuming especially for models with dense discretization. The main issue is that each iteration of the ADMM solver proposed by Zhang et al. (2014) needs to solve a set of

*Corresponding author
Email addresses: taojiong@mail.ustc.edu.cn (Jiong Tao), DengB3@cardiff.ac.uk (Bailin Deng), juyong@ustc.edu.cn (Juyong Zhang)

global linear systems whose size grows with the discretization density of the domain; as the discretization gets denser, the computational cost is dominated by the linear system solving, which becomes a bottleneck of the solver and makes it inefficient in particular for large models.

In this paper, we address the issue identified above, and propose a more scalable and efficient numerical solver for LBC. This is achieved via the following contributions:

- First, we show that the non-negativity constraint in the original LBC formulation is not necessary, in the sense that a formulation without this constraint will result in the same solution. This already allows us to solve the LBC problem without enforcing non-negativity, which improves the convergence speed on popular convex optimization solvers such as MOSEK MOSEK ApS (2017).
- With the removal of the non-negativity constraints, we further reformulate the computation of LBC as an optimization of the gradients of the coordinate functions, subject to a set of linear equality constraints that enforce integrability conditions as well as standard properties of generalized barycentric coordinates. From the optimized gradients, the actual values of LBC can be recovered easily using a parallel breadth-first integration starting from the boundary. The gradient optimization is a convex problem and can be effectively solved using the alternating direction method of multipliers (ADMM) Boyd et al. (2011), where each step is trivially parallelizable. Unlike the original LBC solver, the gradient-based ADMM solver requires no global linear system solving, making it much more scalable and efficient.

We test our new approach on a variety of 2D and 3D models. Experimental results verify the effectiveness of our approach, and its superior efficiency compared with the original LBC solver.

2. Related Work

2.1. Generalized Barycentric Coordinates

Generalized barycentric coordinates, which have been an active research topic since the pioneering work by Möbius (1827), have wide applications in geometric modeling and processing. As they are in general not uniquely defined on a polytope, numerous generalized barycentric coordinates schemes have been proposed in the past to achieve different properties and for different applications. A complete review of different schemes and their properties is beyond the scope of this paper, and the reader is referred to recent publications from Floater (2015) and Hormann and Sukumar (2017) for such surveys. In the following, we will focus on the most relevant works.

In terms of their computational approaches, generalized barycentric coordinates schemes proposed so far fall into two categories. For the first one, the coordinates for a given point are directly computed from the positions of the point itself as well as the control points, using a closed-form expression. Examples include the mean value coordinates and variants Floater (2003); Ju et al. (2005); Floater et al. (2005); Hormann and Floater (2006); Li et al. (2013), Green coordinates Lipman et al. (2008), moving least squares coordinates Manson and Schaefer (2010), and Poisson coordinates Li and Hu (2013), to name a few. These coordinates are straightforward and efficient to compute, and with guarantees of local properties such as smoothness, but often lack control of global properties such as non-negativity.

The other type of generalized barycentric coordinates has not closed-form expressions, and needs to be computed using numerical routines. Although this is computational more expensive, it also enables more direct control of desirable global properties of the coordinate functions. For example, the harmonic coordinates by Joshi et al. (2007) are computed by solving the Laplacian equation, such that the coordinates become harmonic functions which are globally smooth and non-negative. The maximum entropy coordinates, proposed by Hormann and Sukumar (2008), are formulated based on the maximum entropy principle and guaranteed to be positive inside any planar polygon; they are computed by solving a convex optimization problem with Newton's method. The bounded biharmonic weights proposed by Jacobson et al. (2011), although not generalized barycentric coordinates, also achieve non-negative and smooth weight functions via convex constrained optimization, and allow for shape-aware control using cages and skeletons. Weber et al. (2012) propose the biharmonic coordinates as the solution to the biharmonic Dirichlet problem, which

generalize the harmonic coordinates and enable interpolation of boundary derivative data. Zhang et al. (2014) derive the local barycentric coordinates by minimizing their total variation subject to constraints of barycentric properties and non-negativity. For many generalized barycentric coordinates without closed-form expressions, their computation often relies on picewise approximation over a discretized domain. Anisimov et al. (2016) show that subdivision schemes can be applied to refine a coarse approximation of such generalized barycentric coordinates, while retaining key properties such as smoothness and non-negativity.

Many of the generalized barycentric coordinates proposed so far have global support, with non-zero values over the whole domain. For shape deformation using such coordinates as control weights, their global support could result in the lack of local control as well as large memory footprint for storing the weights, both of which are undesirable. To this end, generalized barycentric coordinates schemes with local support are sought after in recent years. Landreneau and Schaefer (2010) present a method that reduces the number of control points influencing a vertex to a user-specified number while achieving resemblance to the original deformation, by solving a Poisson minimization problem. Jacobson et al. (2011) also observe localized control using the bounded biharmonic weights. In Zhang et al. (2014), the locality of coordinate functions are induced via the minimization of total variation which penalized the total length of level sets. The subdivision schemes proposed by Anisimov et al. (2016) can be applied to refine a coarse result from Zhang et al. (2014) while maintaining the locality. Recently, Anisimov et al. (2017) also presented a new closed-form construction of generalized barycentric coordinates that are non-negative, smooth, and locally supported.

2.2. Gradient Domain Geometry Processing

One of the main contributions of this paper is to move the computation of local barycentric coordinates from the function domain to the gradient domain, before recovering the function values from the optimized gradients. Indeed, the idea of gradient domain processing coupled with function domain recovery has been applied to different tasks in geometry processing. A notable example is mesh editing and deformation, where local mesh details are first modified in the gradient domain, followed by a global recovery of the updated mesh shape Yu et al. (2004); Xu et al. (2007). Another example is Poission surface reconstruction from oriented points Kazhdan et al. (2006); Kazhdan and Hoppe (2013), which first derive the gradient of the model's indicator function from the points, and then reconstruct the surface from the gradient by solving a Poisson problem. In addition, to compute geodesic distance on discrete surface, Crane et al. (2013) first compute its gradient with heat diffusion, then recover the geodesic distance using a Possion system.

2.3. Improving Scalability of Numerical Solvers

As advancement in sensing technologies and computer hardware enable the capture and creation of larger and larger 3D models, there is a growing demand for new techniques that can overcome the scalability limitations of existing geometry processing algorithms. For example, Wang (2015) proposes a Cheyshev semi-iterative method to accelerate the convergence of projective and position-based dynamics solvers for physics simulation, and to avoid expensive linear system solving in each iteration. Rabinovich et al. (2017) present a scalable approach for the optimization of flip-preventing energies for mesh parameterization, by iteratively minimizing a simple proxy energy instead of the original distortion energy. Tao et al. (2018) propose a scalable heat method for computer geodesic distance, by reformulating the original heat method from Crane et al. (2013) as an optimization problem for the distance gradient. The reformulation of LBC in this paper is similar to the gradient-based approach taken by Tao et al. (2018), but with a non-smooth target function and a different set of constraints to enforce their defining properties.

3. Background

Before presenting our new approach, we first review in this section the original formulation of local barycentric coordinates (LBC) in Zhang et al. (2014). For a set of control points $\mathbf{c}_1, \ldots, \mathbf{c}_n$ in $\mathbb{R}^2$ or $\mathbb{R}^3$ which are the vertices of a closed control cage, we compute for each $\mathbf{c}_i$ a function $w_i \colon \Omega \mapsto \mathbb{R}$ defined on the domain Ω bounded by the cage, such that $[w_1(\mathbf{x}), \ldots, w_n(\mathbf{x})]$ are generalized barycentric coordinates of $\mathbf{x} \in \Omega$ with respect to the control points and satisfy the following properties:

1. *Reproduction*: $\sum_{i=1}^{n} w_i(\mathbf{x})\mathbf{c}_i = \mathbf{x},\ \forall \mathbf{x} \in \Omega$;
2. *Partition of unity*: $\sum_{i=1}^{n} w_i(\mathbf{x}) = 1$;
3. *Non-negativity*: $w_i(\mathbf{x}) \geq 0\ \ \forall i$;
4. *Lagrange property*: $w_i(\mathbf{c}_j) = \begin{cases} 0, & \text{if } i \neq j \\ 1, & \text{otherwise} \end{cases}$;
5. *Linearity*: Functions $\{w_i\}$ are linear on cage edges and faces;
6. *Smoothness*: Functions $\{w_i\}$ vary smoothly on Ω;
7. *Locality*: A control point only influences its nearby regions, and a point $\mathbf{x} \in \Omega$ is influenced by a small number of control points, i.e., the vector $[w_1(\mathbf{x}), \ldots, w_n(\mathbf{x})]$ is *sparse*.

Zhang et al. (2014) propose to compute LBC by minimizing the sum of their total variation, subject to the constraints of reproduction, partition of unity, non-negative, linearity, and Lagrange property:

$$\begin{aligned} \min_{w_1,\ldots,w_n} \quad & \sum_{i=1}^{n} \int_{\Omega} \phi_i \|\nabla w_i\| \\ \text{s.t.} \quad & \sum_{i=1}^{n} w_i(\mathbf{x})\mathbf{c}_i = \mathbf{x},\ \ \sum_{i=1}^{n} w_i(\mathbf{x}) = 1,\ \ w_i \geq 0,\ \ \forall \mathbf{x} \in \Omega, \\ & w_i(\mathbf{c}_j) = \delta_{ij}\ \ \forall i, j, \\ & w_i \text{ is linear on cage edges and faces } \forall i, \end{aligned} \tag{1}$$

where $\|\nabla w_i\|$ is the gradient norm function of w_i, and $\phi_i : \Omega \mapsto [0, 1]$ is a weight function determined by the distance to $\mathbf{c}_i$. This is a convex constrained optimization problem. It is solved by first discretizing the domain Ω into triangles (in 2D) or tetrahedrons (in 3D), and representing the coordinates $\{w_i\}$ as piecewise linear functions determined by their values at the vertices of the triangles/tetrahedrons. The coordinate values at vertices lying on cage boundary are already determined based on the Lagrange and linearity properties, while the values at the m interior vertices are computed by solving a discretized and reformulated version of the optimization problem (1):

$$\begin{aligned} \min_{\mathbf{W},\mathbf{Z},\mathbf{Y}} \quad & \sum_{s \in \mathcal{C}} \sum_{i=1}^{n} \phi_i^s A_s \|\mathbf{z}_i^s\| + \sigma(\mathbf{W}) \\ \text{s.t.} \quad & \mathbf{Z} = \mathbf{G}(\mathbf{YM} + \mathbf{H}) + \mathbf{E}, \quad \mathbf{W} = \mathbf{YM} + \mathbf{H}, \end{aligned} \tag{2}$$

where $\mathcal{C}$ denotes the set of cells in the discretized domain; A_s is the area (or volume) of the cell s, ϕ_i^s is the value of weight function ϕ_i at s; $\mathbf{Y}$ is an auxiliary variable that parameterizes the feasible set for the linear equality constraints for derive from the reproduction and partition-of-unity properties, with $\mathbf{M}$ being encoding the null space of the linear system matrix and $\mathbf{H}$ being a special solution; $\mathbf{Z} \in \mathbb{R}^{D|\mathcal{C}| \times n}$ is an auxiliary variable containing values $\mathbf{z}_i^s \in \mathbb{R}^D$ that represent the gradients of $\{w_i\}$ at each cell s, and matrices $\mathbf{G}$ and $\mathbf{E}$ encode the affine mappings that derive the gradients on the cells from the function values at the interior vertices; the matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$ is an auxiliary variable that stores the coordinate values at interior vertices, and $\sigma(\mathbf{W})$ is an indicator function for the non-negativity constraint:

$$\sigma(\mathbf{W}) = \begin{cases} 0 & \text{if } \mathbf{W} \geq \mathbf{0}, \\ +\infty & \text{otherwise.} \end{cases}$$

The optimization problem (2) is solved using the alternating direction method of multipliers (ADMM) Boyd et al. (2011), by searching for a stationary point of its augmented Lagrangian function. This is done by alternating between the following steps until convergence:

1. Update $\mathbf{W}$ and $\mathbf{Z}$, by fixing $\mathbf{Y}$ and the dual variables and minimizing the augmented Lagrangian with respect to $\mathbf{W}$ and $\mathbf{Z}$.

2. Update $\mathbf{Y}$, by fixing $\mathbf{W}, \mathbf{Z}$ and the dual variables and minimizing the augmented Lagrangian with respect to $\mathbf{Y}$.

3. Update the dual variables.

Detailed derivation of each step can be found in Zhang et al. (2014). The update of $\mathbf{W}$ and $\mathbf{Z}$ is separable with close-form solutions, thus can be efficiently evaluated in parallel. The update of dual variables amounts to evaluating the violation of linear side constraints in (2), which is also efficient. The update of $\mathbf{Y}$ requires solving an $m \times m$ sparse linear system with n right-hand-side vectors, and is the main contributor to the computational cost of the whole solver. Although the efficiency of $\mathbf{Y}$-update can be improved by pre-factorization of the linear system matrix and parallel solving for different righ-hand-sides, it can still face scalability issues for densely discretized models. This is due to the fast growth of computational time and memory consumption for the prefactorization step with respect to the model size. Another factor that affects the solver performance is the presence of non-negativity constraint, which requires the introduction of an auxiliary variable $\mathbf{W}$ in the ADMM solver. This leads to a large number of variables and slower convergence to the solution.

4. Our Approach

In this section, we present our new approach that addresses the scalability issue of the original LBC solver and improves the efficiency of LBC computation. We first show that the non-negativity constraint in the original LBC optimization problem (1) can be removed without affecting its solution. Based on this observation, we then propose a new gradient-optimization approach that computes LBC without the need for global linear system solving.

4.1. Removing the Non-Negativity Constraint

It is observed in Zhang et al. (2014) that LBC are free of local minimum. Using extensive experiments, we observe that the solution functions to the optimization problem (1) without the non-negativity constraint are free of local minimum as well:

Conjecture 1. *Let functions $\{w_i^* : \Omega \mapsto \mathbb{R}\}$ be the solution to the optimization problem (1) without the non-negativity constraint. Then for each function w_i^* there is no local minimum point in the interior of Ω.*

Although we do not have a rigorous proof for this conjecture, we provide in Appendix A an intuition of why it holds. With this property, we can then show that the non-negativity constraint is not necessary:

Theorem 1. *Let functions $\{w_i^*\}$ be the solution to the optimization problem (1) without the non-negativity constraint. Then they are also the solution to the original optimization problem (1).*

Proof. See Appendix B. □

Model	Time (With Non-Negativity Constraint)	Time (Without)	Squared Distance
Woody (#V:10802)	254.00s	106.67s	1.723×10^{-10}
Cactus (#V:10846)	199.90s	94.58s	4.083×10^{-10}
Gecko (#V:10787)	274.91s	167.35s	1.186×10^{-10}

Table 1: Computation time for solving the LBC problem using the MOSEK library with and without non-negativity constraint, and the average squared distance between these two results. #V denotes the number of vertices in the triangulation of the cage.

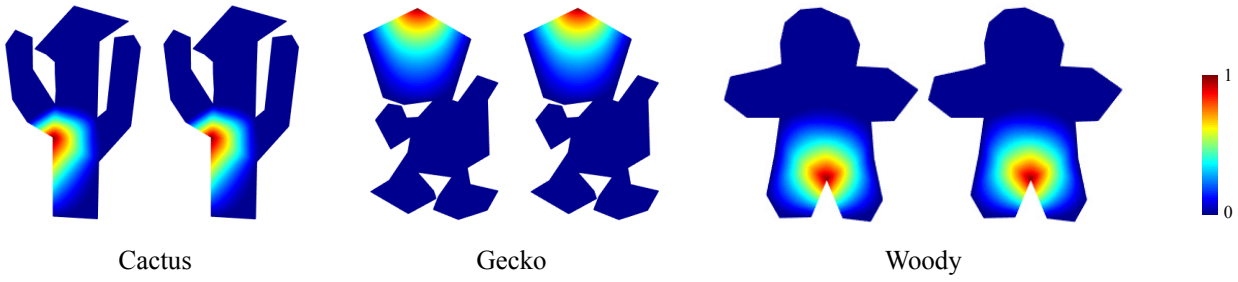


Figure 1: Models for comparing the LBC results with and without the non-negativity constraint. The color coding shows the solution function for one of the control points. Within each pair, the left one is the result with the constraint, and the right one is the result without the constraint.

Theorem 1 shows that we can omit the non-negativity constraint when computing LBC. To numerically verify this, we solve the LBC optimization problem with and without the non-negativity constraints using the MOSEK library MOSEK ApS (2017) — a popular convex optimization solver that produces high-accuracy solutions — and compare their results. Following the notations in Section 3, we discretize the domain bounded by the cage and convert the optimization problem (1) into

$$\min_{\mathbf{w}_1,\ldots,\mathbf{w}_n} \sum_{s\in\mathcal{C}}\sum_{i=1}^{n} \phi_i^s A_s \|\mathbf{G}_s w_i + \mathbf{e}_i^s\|,$$

$$\text{s.t. } \mathbf{P} = \sum_{i=1}^{n} \mathbf{c}_i \mathbf{w}_i^T, \quad \sum_{i=1}^{n} \mathbf{w}_i = \mathbf{1}, \tag{3}$$

$$\mathbf{w}_i \geq \mathbf{0} \quad \forall i, \tag{4}$$

where $\mathbf{P} \in \mathbb{R}^{D\times m}$ are the interior vertex positions, $\mathbf{w}_i \in \mathbb{R}^m$ collects the values of w_i at the interior vertices, $\mathbf{1} \in \mathbb{R}^m$ is a vector with all components equal to 1, and $\mathbf{G}_s w_i + \mathbf{e}_i^s$ is the affine expression of the gradient of w_i in cell s. We solve this problem on three models with and without the constraint (4) (see Fig. 1), and compare the two solutions $\{\mathbf{w}_i^*\}$ and $\{\mathbf{w}_i^\star\}$ using their average squared distance

$$\left(\sum_{i=1}^{n} \|\mathbf{w}_i^* - \mathbf{w}_i^\star\|^2\right)/(m\cdot n). \tag{5}$$

Tab. 1 shows the computational time for the two approaches and the average squared distance between their results. We can see that the removal of non-negativity constraints reduces almost half of the computational time, while the differences between the results is negligible.

For practical computation of LBC, the ADMM solver proposed by Zhang et al. (2014) is often more desirable as it converges faster to a solution with practical accuracy. By removing the non-negativity constraint, we introduce in the next subsection a new gradient-based ADMM solver for computing LBC, which outperforms the original ADMM solver from Zhang et al. (2014) and overcomes its scalability limitation.

4.2. Gradient-Based Formulation

Although the removal of non-negativity constraint simplifies the LBC optimization problem, it does not address directly the scalability issue faced by the ADMM solver from Zhang et al. (2014), which is due to global linear system solving in each iteration. In the following, we introduce a new gradient-based formulation for computing LBC, which avoids solving global linear systems and is trivially parallelizable. Our key insight is that although the original LBC formulation optimizes the coordinate function values, it is equivalent to a constrained optimization of their gradients if we remove the non-negativity constraint.

Specifically, within the original formulation (1), the target function is already formulated using the gradients. For the constraints of the reconstruction and partition-of-unity properties

$$\sum_{i=1}^{n} w_i(\mathbf{x})\mathbf{c}_i = \mathbf{x}, \qquad \sum_{i=1}^{n} w_i(\mathbf{x}) = 1, \qquad \forall \mathbf{x} \in \Omega,$$

we can take the gradient on both sides of each equation, and derive their equivalent gradient conditions

$$\sum_{i=1}^{n} \mathbf{c}_i \left(\nabla w_i(\mathbf{x})\right)^T = \mathbf{I}, \qquad \sum_{i=1}^{n} \nabla w_i(\mathbf{x}) = \mathbf{0}, \qquad \forall \mathbf{x} \in \Omega. \tag{6}$$

where $\mathbf{I}$ is an identity matrix. Moreover, the constraints of the Lagrange and linearity properties imply pre-determined and piecewise-constant projected gradients onto the boundary edges (in 2D) and boundary faces (in 3D) of the control cage. Thus they can be formulated as linear equality constraints for the gradient functions along the cage boundary. Finally, an arbitrary vector function defined on a given domain is in general not the gradient of a scalar function; to ensure that the optimization results are valid gradient functions, we must enforce an integrability condition on each variable function. Combining all the gradient conditions above, we derive a convex constrained optimization problem for the gradient functions of LBC.

To solve this gradient optimization problem, we follow a similar procedure as Zhang et al. (2014): we represent the LBC as piecewise linear functions on a disretized domain, and introduce auxiliary variables to rewrite the problem in a form that can be solved using ADMM. In the following, we first present the approach for 2D problems, and then discuss its extension to 3D.

4.2.1. 2D Formulation

For a piecewise linear coordinate function, its gradient is piecewise constant. Therefore, we introduce for each discretization triangle s a gradient variable $\mathbf{g}_s^i$. The target function of our gradient optimization problem, as well as the reconstruction and partition-of-unity conditions (6), can be directly translated using these variables. For the boundary conditions, we first use the Lagrange and linearity properties to compute the directional derivative h_b^i of each coordinate function w_i at each boundary edge b. Then the projected gradient condition along edge b can be written as

$$\mathbf{b} \cdot \mathbf{g}_b^i = h_b^i, \quad \forall i,$$

where $\mathbf{b}$ is the unit edge vector of b, and $\mathbf{g}_b^i$ is the gradient variable at the triangle incident with b. Finally, for the triangle-wise gradients to be integrable, the gradients $\mathbf{g}_{e_1}^i, \mathbf{g}_{e_2}^i$ on two adjacent triangles must have a common projection onto the edge e shared by the triangles, i.e.,

$$\mathbf{e} \cdot \mathbf{g}_{e_1}^i = \mathbf{e} \cdot \mathbf{g}_{e_2}^i,$$

where $\mathbf{e}$ is the unit vector for edge e. In this way, we formulate a convex optimization problem for the gradient variables:

$$\min_{\{\mathbf{g}_s^i\}} \quad \sum_{s \in \mathcal{C}} \sum_{i=1}^{n} \phi_i^s A_s \|\mathbf{g}_s^i\|, \tag{7}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \mathbf{g}_s^i = \mathbf{0}, \quad \sum_{i=1}^{n} \mathbf{c}_i (\mathbf{g}_s^i)^T = \mathbf{I}, \quad \forall\ s \in \mathcal{C}, \tag{8}$$

$$\mathbf{b} \cdot \mathbf{g}_b^i = h_b^i, \quad i = 1, \ldots, n, \quad \forall\ b \in \mathcal{B}, \tag{9}$$

$$\mathbf{e} \cdot (\mathbf{g}_{e_1}^i - \mathbf{g}_{e_2}^i) = \mathbf{0}, \quad i = 1, \ldots, n, \quad \forall\ e \in \mathcal{E}_{\text{int}} \tag{10}$$

where $\mathcal{B}$ and $\mathcal{E}_{\text{int}}$ denote the set of boundary edges and the set of interior edges, respectively.

To solve this problem with ADMM, we introduce auxiliary variables to rewrite it with a separable target function and linear equality constraints. To facilitate presentation, we denote all gradient variables with a

matrix $\mathbf{G} \in \mathbb{R}^{2m_f \times n}$, where m_f is the number of faces. Additionally, we introduce the following auxiliary variables: matrix $\mathbf{Z} \in \mathbb{R}^{2m_f \times n}$ for computing the total variation; matrix $\mathbf{X} \in \mathbb{R}^{2m_b \times n}$ for enforcing the boundary conditions (9), where m_b is the number boundary edges; matrix $\mathbf{Y} \in \mathbb{R}^{4m_e \times n}$, which contains two gradient vectors for each internal edge for enforcing the integrability conditions (10), with m_e being the number of internal edges. Then we rewrite the problem as

$$\begin{aligned} \min_{\mathbf{G},\mathbf{Z},\mathbf{X},\mathbf{Y}} \quad & \sum_{s\in\mathcal{C}} \sum_{i=1}^{n} \phi_i^s A_s \|\mathbf{z}_s^i\| + \sigma_1(\mathbf{G}) + \sigma_2(\mathbf{X}) + \sigma_3(\mathbf{Y}), \\ \text{s.t.} \quad & \mathbf{Z} = \mathbf{G}, \quad \mathbf{X} = \mathbf{G}\mathbf{S}_X, \quad \mathbf{Y} = \mathbf{G}\mathbf{S}_Y. \end{aligned}$$

where $\sigma_1, \sigma_2, \sigma_3$ are indicator functions for the reconstruction and partition-of-unity conditions (8), the boundary conditions (9), and the integrability conditions (10), respectively. $\mathbf{S}_X$ and $\mathbf{S}_Y$ are sparse selection matrices that choose the gradient variables that are subject to the boundary conditions and the integrability conditions, respectively. ADMM solves this problem by searching for a stationary point of its augmented Lagrangian function, which is defined as

$$\begin{aligned} L(\mathbf{G},\mathbf{Z},\mathbf{X},\mathbf{Y},\boldsymbol{\lambda}_Z,\boldsymbol{\lambda}_X,\boldsymbol{\lambda}_Y) \; = & \sum_{s\in\mathcal{C}} \sum_{i=1}^{n} \phi_i^s A_s \|\mathbf{z}_s^i\| + \sigma_1(\mathbf{G}) + \sigma_2(\mathbf{X}) + \sigma_3(\mathbf{Y}) + \boldsymbol{\lambda}_Z \odot (\mathbf{Z} - \mathbf{G}) + \boldsymbol{\lambda}_X \odot (\mathbf{X} - \mathbf{G}\mathbf{S}_X) \\ & + \boldsymbol{\lambda}_Y \odot (\mathbf{Y} - \mathbf{G}\mathbf{S}_Y) + \frac{\mu}{2}(\|\mathbf{Z} - \mathbf{G}\|^2 + \|\mathbf{X} - \mathbf{G}\mathbf{S}_X\|^2 + \|\mathbf{Y} - \mathbf{G}\mathbf{S}_Y\|^2), \end{aligned}$$

where $\boldsymbol{\lambda}_Z \in \mathbb{R}^{2m_f \times n}$, $\boldsymbol{\lambda}_X \in \mathbb{R}^{2m_b \times n}$, $\boldsymbol{\lambda}_Y \in \mathbb{R}^{4m_e \times n}$ are dual variables, $\odot$ denotes coefficient-wise multiplication, and $\mu > 0$ is a penalty parameter. To search for a stationary point, we iteratively run the following three steps.

1. Update $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ while fixing $\mathbf{G}$ and the dual variables The $\mathbf{X}$-, $\mathbf{Y}$-, and $\mathbf{Z}$-updates are separable and can be done in parallel:

- The $\mathbf{X}$-subproblem takes the following form:

$$\min_{\mathbf{X}} \quad \sigma_2(\mathbf{X}) + \frac{\mu}{2} \left\| \mathbf{X} - \mathbf{G}\mathbf{S}_X + \frac{\boldsymbol{\lambda}_X}{\mu} \right\|^2,$$

  which requires projecting $\mathbf{G}\mathbf{S}_X - \boldsymbol{\lambda}_X/\mu$ onto the closet gradient vectors that satisfy the boundary conditions. It has a closed-form solution that is further separable between different boundary edges:

$$\mathbf{x}_b^i = \mathbf{p}_b^i - \mathbf{b}(\mathbf{b} \cdot \mathbf{p}_b^i - h_b^i),$$

  where $\{\mathbf{x}_b^i\}$ are the components of $\mathbf{X}$ corresponding to the boundary edge b, and $\{\mathbf{p}_b^i\}$ are the corresponding components from matrix $\mathbf{G}\mathbf{S}_X - \boldsymbol{\lambda}_X/\mu$.

- The $\mathbf{Y}$-subproblem takes the following form:

$$\min_{\mathbf{Y}} \quad \sigma_3(\mathbf{Y}) + \frac{\mu}{2} \left\| \mathbf{Y} - \mathbf{G}\mathbf{S}_Y + \frac{\boldsymbol{\lambda}_Y}{\mu} \right\|^2,$$

  which requires projecting $\mathbf{G}\mathbf{S}_Y - \boldsymbol{\lambda}_Y/\mu$ onto the closest vectors that satisfy the integrability condition. It also has a closed-form solution that is separable between different internal edges:

$$\mathbf{y}_{e_1}^i = \mathbf{q}_{e_1}^i + \mathbf{d}_e^i, \qquad \mathbf{y}_{e_2}^i = \mathbf{q}_{e_2}^i - \mathbf{d}_e^i,$$

  where $\{\mathbf{y}_{e_1}^i, \mathbf{y}_{e_2}^i\}$ are the components of $\mathbf{Y}$ corresponding to the internal edge e, $\{\mathbf{q}_{e_1}^i, \mathbf{q}_{e_2}^i\}$ are the corresponding components from $\mathbf{G}\mathbf{S}_Y - \boldsymbol{\lambda}_Y/\mu$, and $\mathbf{d}_e^i = \frac{1}{2}\mathbf{e}\mathbf{e}^T(\mathbf{q}_{e_2}^i - \mathbf{q}_{e_1}^i)$.

- The $\mathbf{Z}$-subproblem takes the following form:

$$\min_{\mathbf{Z}} \quad \sum_{s\in\mathcal{C}}\sum_{i=1}^{n} \phi_i^s A_s \|\mathbf{z}_s^i\| + \frac{\mu}{2}\left\|\mathbf{Z} - \mathbf{G} + \frac{\boldsymbol{\lambda}_Z}{\mu}\right\|^2,$$

with a separable closed-form solution

$$\mathbf{z}_s^i = \begin{cases} \mathbf{0}, & \text{if } \|\mathbf{a}_s^i\| \le \frac{\phi_i^s A_s}{\mu}, \\ (1 - \frac{\phi_i^s A_s}{\mu\|\mathbf{a}_s^i\|})\mathbf{a}_s^i, & \text{otherwise,} \end{cases}$$

where $\{\mathbf{z}_s^i\}$ are the components of $\mathbf{Z}$ corresponding to the triangle s, and $\{\mathbf{a}_s^i\}$ are the corresponding component from $\mathbf{G} - \boldsymbol{\lambda}_Z/\mu$.

2. Update G while fixing X, Y, Z and the dual variables This reduces to independent subproblems for each triangle:

$$\min_{\mathbf{g}_s} \quad \sigma_1^s(\mathbf{g}_s) + \|\mathbf{g}_s - \mathbf{r}_s\|^2, \tag{11}$$

where $\mathbf{g}_s \in \mathbb{R}^{2\times n}$ stacks the n gradient variables from $\mathbf{G}$ that correspond to triangle s, $\sigma_1^s(\mathbf{g}_s)$ is the indicator function for the conditions (8), and

$$\mathbf{r}_s = \frac{1}{M}\left(\mathbf{z}_s + \frac{\boldsymbol{\lambda}_Z^s}{\mu} + \sum_{t\in\mathcal{X}(s)}(\mathbf{x}_t + \frac{\boldsymbol{\lambda}_X^t}{\mu}) + \sum_{u\in\mathcal{Y}(s)}(\mathbf{y}_u + \frac{\boldsymbol{\lambda}_Y^u}{\mu})\right),$$

where $\mathbf{z}_s$, $\{\mathbf{x}_t \mid t \in \mathcal{X}(s)\}$, $\{\mathbf{y}_u \mid u \in \mathcal{Y}(s)\}$, $\{\boldsymbol{\lambda}_X^t \mid t \in \mathcal{X}(s)\}$, and $\{\boldsymbol{\lambda}_Y^u \mid u \in \mathcal{Y}(s)\}$ denote the $2 \times n$ blocks from matrices $\mathbf{Z}$, $\mathbf{X}$, $\mathbf{Y}$, $\boldsymbol{\lambda}_X$, and $\boldsymbol{\lambda}_Y$ that correspond to $\mathbf{r}_s$, and $M = 1 + |\mathcal{X}(s)| + |\mathcal{Y}(s)|$. In other words, $\mathbf{r}_s$ is the average of all candidate values for $\mathbf{g}_s$ derived from the auxiliary and dual variables. Solving problem (11) means projecting $\mathbf{r}_s$ onto the subspace that satisfies the linear constraints (8), with a closed-form solution

$$\mathbf{g}_s = \widetilde{\mathbf{g}} + (\mathbf{r}_s - \widetilde{\mathbf{g}})\,\mathbf{K}^T$$

where $\widetilde{\mathbf{g}}$ is the least-norm solution to equations (8), and matrix $\mathbf{K}$ represents the projection operator onto the null space of the system matrix for (8). Note that $\widetilde{\mathbf{g}}$ and $\mathbf{K}$ only need to be precomputed once and then reused for all triangles.

3. Update dual variables This is done by adding $\mu(\mathbf{X} - \mathbf{G}\mathbf{S}_X)$, $\mu(\mathbf{Y} - \mathbf{G}\mathbf{S}_Y)$, and $\mu(\mathbf{Z} - \mathbf{G})$ to matrices $\boldsymbol{\lambda}_X$, $\boldsymbol{\lambda}_Y$, and $\boldsymbol{\lambda}_Z$, respectively.

As the optimization problem is convex, the above iterative algorithm is guaranteed to converge to a global optimum Boyd et al. (2011). We check the convergence of the algorithm using the primal residual $\mathbf{r}_\mathrm{p}$ and dual residual $\mathbf{r}_\mathrm{d}$:

$$\mathbf{r}_\mathrm{p} = \begin{bmatrix} \mathbf{X} - \mathbf{G}\mathbf{S}_X \\ \mathbf{Y} - \mathbf{G}\mathbf{S}_Y \\ \mathbf{Z} - \mathbf{G} \end{bmatrix}, \qquad \mathbf{r}_\mathrm{d} = \mu \begin{bmatrix} \delta\mathbf{G}\mathbf{S}_X \\ \delta\mathbf{G}\mathbf{S}_Y \\ \delta\mathbf{G} \end{bmatrix},$$

where $\delta\mathbf{G}$ is the difference of $\mathbf{G}$ between two consecutive iterations. We terminate the algorithm if the norms of $\mathbf{r}_\mathrm{p}$ and $\mathbf{r}_\mathrm{d}$ are smaller than thresholds ε_p and ε_d respectively, or if the number of iterations exceeds a user-specified threshold I_max.

Remark. For both the original LBC solver described in Section 3 and our new solver, there is a step that combines the auxiliary variables to update the main primal variable: the $\mathbf{Y}$-update of the original LBC solver, and the $\mathbf{G}$-update of our new solver. This is where the difference between their scalability and efficiency comes from. In this step, the original LBC solver needs to solve multiple $m \times m$ global linear systems where m is the number of internal vertices; the size of the linear system grows with the discretization density, causing bottleneck for large models. In comparison, our solver only needs to perform projection onto the null space of the small local linear system (8), whose size does not depend on the discretization density. Therefore, our method remains efficient for large models and is more scalable than the original LBC solver. The original LBC solver has to solve global linear systems, because its optimization variables contain both the coordinate values and their gradients; as a result, when updating the coordinate values using the auxiliary gradient variables, it must solve global linear systems to implicitly integrate the gradient variables. In comparison, our formulation only involves gradient variables, which avoids frequent switching from the gradient domain to the coordinate function domain and eliminates the need for global linear system solving.

4.2.2. Recovery of Coordinate Values

After the optimization of gradients, they still need to be integrated to derive the actual values of LBC. As the LBC values at boundary dicretization vertices are already pre-determined from the Lagrange and linearity properties, we can integrate the gradients starting from the boundary, to recover the LBC values at all interior vertices. Specifically, we perform breadth-first search from the boundary to separate the all vertices into multiple levels, with the first level containing all vertices on the boundary. Then we propagate the LBC values from the boundary vertices level by level. Note that for each interior vertex v_k, there exists an adjacent vertex v'_k from the previous level. Thus the LBC values are propagated from v'_k to v_k via

$$w_i(v_k) = w_i(v'_k) + \mathbf{g}^i_{v_k} \cdot (\mathbf{v}_k - \mathbf{v}'_k), \quad i = 1, \ldots, n,$$

where $\mathbf{v}_k, \mathbf{v}'_k \in \mathbb{R}^2$ are the positions of v_k and v'_k respectively, and $\mathbf{g}^i_{v_k}$ is the optimized gradient from a face incident with both v_k and v'_k. Within each level, the value updates are independent between different vertices and can be done in parallel. Overall, the computational time for recovering coordinate values is comparable with one iteration of the ADMM solver.

4.2.3. Extension to 3D

To extend our solver to 3D models, we first discretize the domain into tetrahedrons and introduce a gradient variable $\mathbf{g}^i_s \in \mathbb{R}^3$ for each tetrahedron s. The target function (7) and the linear constraints (8) can be directly extended to 3D. Regarding the boundary condition, for each boundary face b the projection of gradient $\mathbf{g}^i_b$ from its incident tetrahedron must coincide with the pre-determined gradient $\mathbf{h}^i_b$ derived from Lagrange and linearity properties. Therefore, condition (9) is replaced by

$$\mathbf{n}_b \times (\mathbf{g}^i_b - \mathbf{h}^i_b) = \mathbf{0}, \quad i = 1, \ldots, n, \quad \forall\ b \in \mathcal{B},$$

where $\mathcal{B}$ denotes the set of boundary faces of the cage, and $\mathbf{n}_b$ is the unit normal of face b. For the integrability condition, it is necessary that the gradients $\mathbf{g}^i_{f_1}, \mathbf{g}^i_{f_2}$ on two tetrahedrons sharing a face f must project to the same vector on f. Thus condition (10) is replaced by

$$\mathbf{n}_f \times (\mathbf{g}^i_{f_1} - \mathbf{g}^i_{f_2}) = \mathbf{0}, \quad i = 1, \ldots, n, \quad \forall\ f \in \mathcal{F}_{\text{int}},$$

where $\mathcal{F}_{\text{int}}$ is the set of interior faces, and $\mathbf{n}_f$ is the unit normal of face f. Similar to the 2D case, this problem can be solved efficiently using ADMM, with closed-form solution for each step and without global linear system solving.

5. Experimental Results

In this section, we provide a series of examples to demonstrate the properties of the effectiveness of efficiency of our new solver, and comparison with the original LBC solver from Zhang et al. (2014).

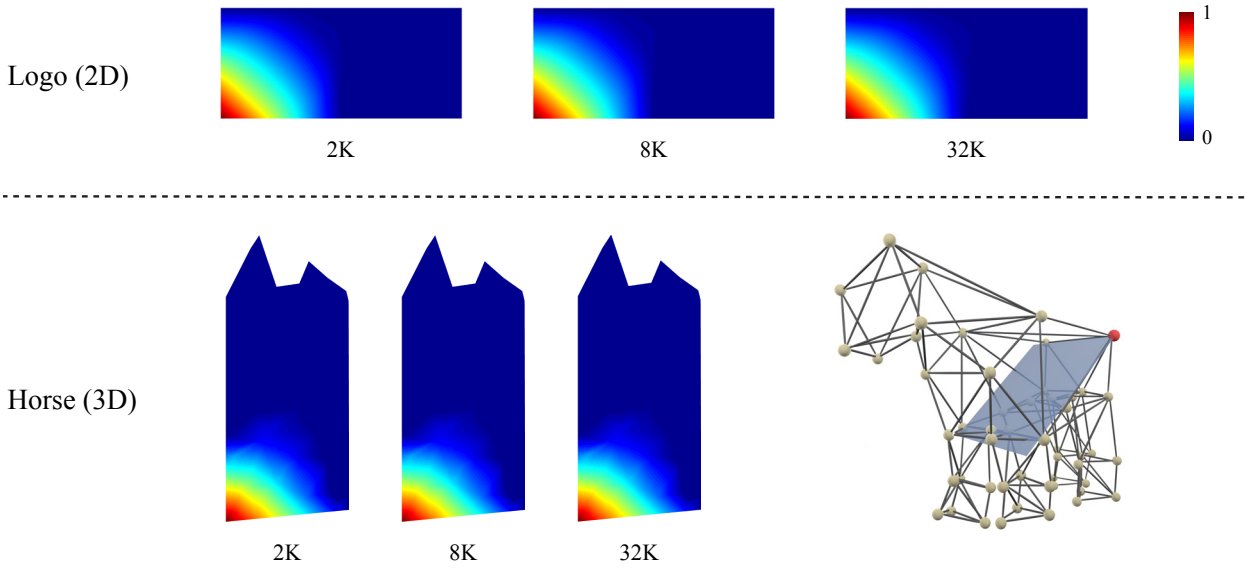


Figure 2: Color coding of the LBC function for one control point, computed using our new solver. For the 3D model, the color coding is shown on a planar slice through the interior of the cage, with the slicing plane shown in yellow. The numbers shown refer to the number of vertices in the respective tetrahedronization/triangulation. For the same model in different discretization resolution, our solver converges to results with similar accuracy.

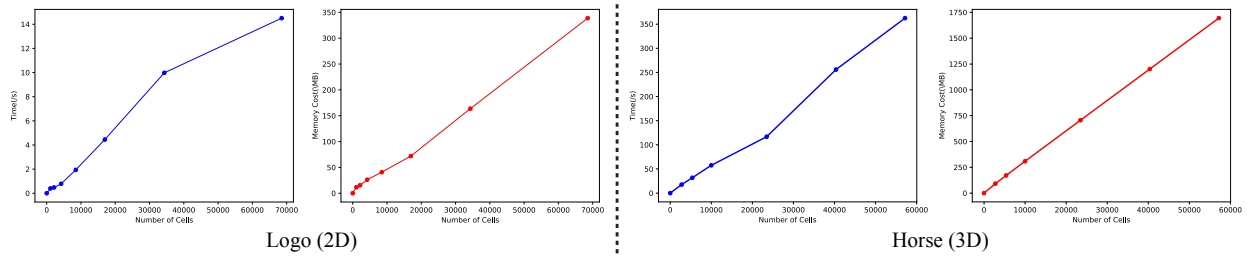


Figure 3: The plots of computational time (in blue) and memory consumption (in red) for our solver on the models in Fig. 2. Both exhibit approximately linear growth with respect to the model resolution.

Implementation. Our solver is implemented in C++, using the EIGEN library Guennebaud et al. (2010) for linear algebra routines. 2D domains are triangulated using the TRIANGLE package Shewchuk (1996), and 3D domains are tetrahedronized using TETGEN Si (2015). All subproblems of our solver are solved in parallel using OpenMP. For all the test models, the residual convergence thresholds are set as $\varepsilon_{\mathrm{p}} = N_{\mathrm{p}} \times 10^{-7}$ and $\varepsilon_{\mathrm{d}} = N_{\mathrm{d}} \times 10^{-6}$, where N_{p} and N_{d} are the numbers of components in the residual matrices $\mathbf{r}_{\mathrm{p}}$ and $\mathbf{r}_{\mathrm{d}}$, respectively. The penalty parameter is set to $\mu = 100$ for 2D models and $\mu = 10000$ for 3D models. To compare with the original LBC solver, we use its implementation provided by the authors [1], and run it with OpenMP parallelization enabled. All experiments are run on a desktop PC with a quad-core Intel Core i5-4590 processor at 3.3 GHz and 16GB ram.

5.1. Space and Time Complexity

In Figs. 2 and 3, we show the scaling of our solver by running the algorithm on two models, each with increasing density of discretization. Fig. 2 shows the color coding of the resulting coordinate functions, where

[1] https://github.com/bldeng/LBC

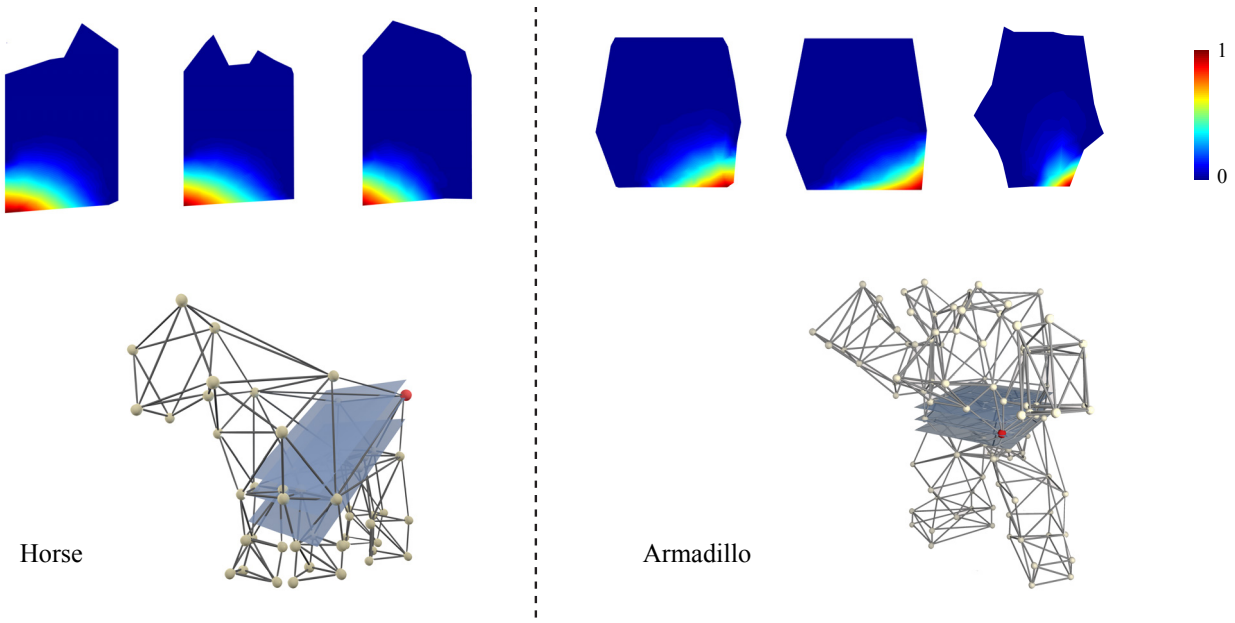


Figure 4: Color coding the LBC function for one control point (shown in red), computed using our new solver on two 3D models. For each model, three parallel planes are used to slice through the model to visualise the function values on the intersection. The tetrahedronization resolution, computational time and accuracy for each model are shown in Tab. 2.

we can see solver converges to similar accuracy across different resolutions. Fig. 3 plots the computational time and memory consumption for the examples in Fig. 2. We observe approximately linear growth of time and memory with respect to the model resolution.

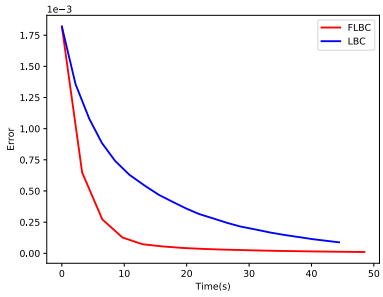
5.2. Comparison with LBC

In this subsection, we compare our new solver with the original LBC solver on a set of 2D and 3D models. To compare the accuracy between the solvers, we use MOSEK to compute a high-accuracy solution and use it as the ground truth. The solution from other solvers are compared to the ground truth, using their average squared distance defined in Equation (5) as the error measure.

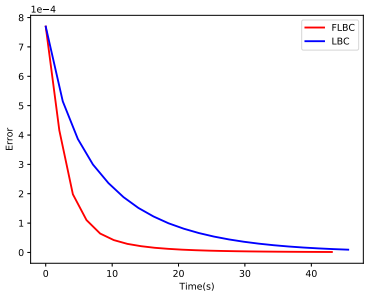
Tab. 2 summarises the computational time and accuracy between the two solvers on different models. In all cases, our solver significantly decreases the computational time than the original LBC solver while achieving better accuracy. Fig. 4 visualises the result functions computed using our method on the two 3D models.

Model	#Vertices	n	LBC (error)	LBC (time)	FLBC (error)	FLBC (time)
Logo	26823	6	3.18×10^{-4}	4.94	$\mathbf{5.72 \times 10^{-5}}$	**2.78**
Woody	10802	26	2.43×10^{-4}	19.30	$\mathbf{2.30 \times 10^{-4}}$	**12.43**
Cactus	10846	27	5.87×10^{-4}	22.60	$\mathbf{2.2 \times 10^{-4}}$	**16.98**
Gecko	10787	34	8.90×10^{-4}	25.06	$\mathbf{3.81 \times 10^{-4}}$	**16.53**
Horse	5845	50	1.66×10^{-5}	39.35	$\mathbf{1.21 \times 10^{-5}}$	**21.87**
Armadillo	6644	110	5.2×10^{-5}	134.65	$\mathbf{4.16 \times 10^{-5}}$	**120.85**

Table 2: Comparison of computational time (in seconds) and accuracy between the original LBC solver and our new solver (FLBC). #Vertices denotes the number of vertices in the triangulation or tetrahedronization of the cage. The ground truth is computed using MOSEK.

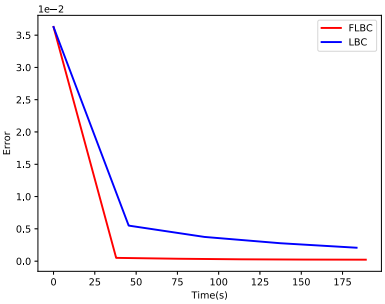


Gecko (2D)
#Vertices: 10787
#Control Points: 34

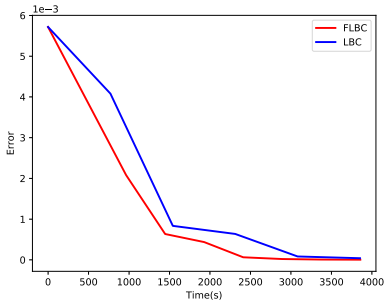


Horse (3D)
#Vertices: 5845
#Control Points: 50

Figure 5: Comparison of solution accuracy with respect to computational time during the iterations of our new solver (FLBC) and the original LBC solver, on two large models. The ground truth is computed using MOSEK.



Logo (2D)
#Vertices: 1.07M
#Control Points: 6



Horse (3D)
#Vertices: 506K
#Control Points: 50

Figure 6: Comparison of solution accuracy with respect to computational time during iterations of our new solver (FLBC) and the original LBC solver, on two large models. The ground truth is computed by running the LBC solver until full convergence.

Fig. 5 further compares between the two solvers by plotting the relation between their solution accuracy and computational time on the Gecko and Horse models. On both models, our solver improves the accuracy faster than the original LBC solver.

Finally, Fig. 6 plots the solution accuracy with respect to computational time for our solver and the original LBC solver on two models with high-resolution triangulation/tetrahedronization. As MOSEK runs out of memory on these models, we run the original LBC solver until full convergence and takes the result as ground truth. Our solver is more efficient than the original LBC solver when both achieve the same accuracy.

6. Discussion and Conclusion

In this paper, we propose a new algorithm for computing local barycentric coordinates. By formulating it as a gradient optimization problem, our new ADMM solver avoids expensive global linear system solving, with improved scalability and efficiency than the original LBC solver. Experimental results verify the effectiveness of our new approach.

Given that there are other generalized barycentric coordinates schemes that rely on numerical optimization, a natural direction for further research is to extend our approach to other optimization-based barycentric coordinates. The main challenge here is to deal with constraints that cannot be directly formulated using gradients, such as bound constraints for the coordinates. One potential way to do so is to relate internal coordinate values with boundary values via line integral of gradients, thereby transforming the bound constraints for coordinates as an integral constraints for their gradients. Further investigation will be done in a future work.

Although our reformulation comes with improved scalability, its asymptotic convergence rate is still the same as other ADMM solvers. Recently, there are research efforts to accelerate the convergence of such

first-order solvers Wang (2015); Peng et al. (2018). These techniques are complementary to our gradient-based approach, and as a future work they can be integrated to further improve the efficiency of LBC computation.

Finally, our technique is built upon the observation described in Conjecture 1. Proving the conjecture in a rigorous way will be an interesting future work.

Acknowledgments

This work was supported by National Natural Science Foundation of China (No. 61672481), and Youth Innovation Promotion Association CAS (No. 2018495).

Appendix A. Intuition for Conjecture 1

Suppose one of the functions w_k^* has a local minimum point q in the interior of Ω. Then we can always find a convex domain $\Gamma \subset \Omega$ that contains point q in its interior, such that the value of w_k^* along its boundary $\partial\Gamma$ is strictly larger than $w_k^*(q)$. Since we assume that the functions $\{w_i^*\}$ solve the optimization problem (1) without the non-negativity constraint, they must satisfy all other constraints of problem (1). We can then construct another set of functions $\{w_i^\dagger\}$ with the following properties:

- $w_i^\dagger$ and w_i^* has the same value on $\Omega \setminus \Gamma$ and $\partial\Gamma$.
- The value of $w_i^\dagger$ in the interior of Γ is computed by interpolating the w_i^* along $\partial\Gamma$ using a transfinite barycentric kernel $b_i(x, y)$ ($x \in \Gamma$, $y \in \partial\Gamma$), so that $b_i(x, y) \geq 0$, and $w_i^\dagger(x) = \int_\alpha^\beta b_i(x, r(t))w_i^*(r(t))dt$ where $r : [\alpha, \beta] \mapsto \partial\Gamma$ is a parameterization of $\partial\Gamma$ Belyaev (2006); Kosinka and Bartoň (2016).

Then by construction, $\{w_i^\dagger\}$ also satisfy all constraints of problem (1) other than non-negativity. Intuitively, by finding appropriate kernels $\{b_i\}$, we can ensure that each $w_i^\dagger$ blends the values of w_i^* along $\partial\Gamma$ and with less variation than w_i^* on Γ, so that $\{w_i^\dagger\}$ produce a smaller target function for problem (1) than $\{w_i^*\}$. We then arrive at a contradiction that $\{w_i^*\}$ cannot be the solution to (1) without non-negativity constraint.

Appendix B. Proof for Theorem 1

First, we prove by contradiction that the solution functions $\{w_i^*\}$ without the non-negativity constraint are indeed non-negative over the whole domain Ω. Suppose there exists a function w_i^* that is not non-negative over Ω. Due to the Lagrange and linearity properties, w_i^* must be non-negative along the boundary of Ω. Then there must exists a local minimum point $\mathbf{p}$ for w_i^* where $w_i^*(\mathbf{p}) < 0$, which violates Theorem 1. Therefore, each function w_i^* must be non-negative over the whole domain.

Suppose the solution functions $\{w_i^*\}$ without non-negativity constraints are not the solution to the original problem (1). Then because each w_i^* is non-negative, the actual solution $\{\overline{w}_i^*\}$ to (1) must produce a lower target function value than $\{w_i^*\}$ while satisfying all the constraints. This will violate the assumption that $\{w_i^*\}$ are the solution without the non-negativity constraint. Thus $\{w_i^*\}$ must also be a solution to the original optimization problem (1).

References

Anisimov, D., Deng, C., Hormann, K., 2016. Subdividing barycentric coordinates. Computer Aided Geometric Design 43, 172–185.

Anisimov, D., Panozzo, D., Hormann, K., 2017. Blended barycentric coordinates. Computer Aided Geometric Design 52, 205–216.

Belyaev, A., 2006. On transfinite barycentric coordinates, in: Proceedings of the Fourth Eurographics Symposium on Geometry Processing, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland. pp. 89–99.

Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends in Machine Learning 3, 1–122.

Crane, K., Weischedel, C., Wardetzky, M., 2013. Geodesics in heat: A new approach to computing distance based on heat flow. ACM Trans. Graph. 32, 152:1–152:11.
Farbman, Z., Hoffer, G., Lipman, Y., Cohen-Or, D., Lischinski, D., 2009. Coordinates for instant image cloning. ACM Trans. Graph. 28, 67:1–67:9.
Floater, M.S., 1997. Parametrization and smooth approximation of surface triangulations. Computer aided geometric design 14, 231–250.
Floater, M.S., 2003. Mean value coordinates. Computer aided geometric design 20, 19–27.
Floater, M.S., 2015. Generalized barycentric coordinates and applications. Acta Numerica 24, 161–214.
Floater, M.S., Kós, G., Reimers, M., 2005. Mean value coordinates in 3d. Computer Aided Geometric Design 22, 623–631.
Guennebaud, G., Jacob, B., et al., 2010. Eigen v3. http://eigen.tuxfamily.org.
Hormann, K., Floater, M.S., 2006. Mean value coordinates for arbitrary planar polygons. ACM Trans. Graph. 25, 1424–1441.
Hormann, K., Sukumar, N., 2008. Maximum entropy coordinates for arbitrary polytopes. Comput. Graph. Forum 27, 1513–1520.
Hormann, K., Sukumar, N. (Eds.), 2017. Generalized Barycentric Coordinates in Computer Graphics and Computational Mechanics. CRC Press.
Jacobson, A., Baran, I., Popović, J., Sorkine, O., 2011. Bounded biharmonic weights for real-time deformation. ACM Trans. Graph. 30, 78:1–78:8.
Joshi, P., Meyer, M., DeRose, T., Green, B., Sanocki, T., 2007. Harmonic coordinates for character articulation. ACM Trans. Graph. 26.
Ju, T., Schaefer, S., Warren, J., 2005. Mean value coordinates for closed triangular meshes. ACM Trans. Graph. 24, 561–566.
Ju, T., Zhou, Q.Y., van de Panne, M., Cohen-Or, D., Neumann, U., 2008. Reusable skinning templates using cage-based deformations. ACM Trans. Graph. 27, 122:1–122:10.
Kazhdan, M., Bolitho, M., Hoppe, H., 2006. Poisson surface reconstruction, in: Proceedings of the Fourth Eurographics Symposium on Geometry Processing, Eurographics Association. pp. 61–70.
Kazhdan, M., Hoppe, H., 2013. Screened poisson surface reconstruction. ACM Trans. Graph. 32, 29:1–29:13.
Kosinka, J., Bartoň, M., 2016. Convergence of barycentric coordinates to barycentric kernels. Computer Aided Geometric Design 43, 200–210.
Landreneau, E., Schaefer, S., 2010. Poisson-based weight reduction of animated meshes. Computer Graphics Forum 29, 1945–1954.
Li, X.Y., Hu, S.M., 2013. Poisson coordinates. IEEE Transactions on Visualization and Computer Graphics 19, 344–352.
Li, X.Y., Ju, T., Hu, S.M., 2013. Cubic mean value coordinates. ACM Trans. Graph. 32, 126:1–126:10.
Lipman, Y., Kopf, J., Cohen-Or, D., Levin, D., 2007. GPU-assisted positive mean value coordinates for mesh deformations, in: Proceedings of the Fifth Eurographics Symposium on Geometry Processing, Eurographics Association. pp. 117–123.
Lipman, Y., Levin, D., Cohen-Or, D., 2008. Green coordinates. ACM Trans. Graph. 27, 78:1–78:10.
Manson, J., Schaefer, S., 2010. Moving least squares coordinates. Computer Graphics Forum 29, 1517–1524.
Möbius, A.F., 1827. Der barycentrische Calcul: ein neues Hülfsmittel zur analytischen Behandlung der Geometrie. Barth.
MOSEK ApS, 2017. The MOSEK optimization software. URL: http://www.mosek.com/.
Peng, Y., Deng, B., Zhang, J., Geng, F., Qin, W., Liu, L., 2018. Anderson acceleration for geometry optimization and physics simulation. ACM Trans. Graph. 37, 42:1–42:14.
Rabinovich, M., Poranne, R., Panozzo, D., Sorkine-Hornung, O., 2017. Scalable locally injective mappings. ACM Trans. Graph. 36.
Rustamov, R.M., Lipman, Y., Funkhouser, T., 2009. Interior distance using barycentric coordinates. Computer Graphics Forum 28, 1279–1288.
Shewchuk, J.R., 1996. Triangle: Engineering a 2d quality mesh generator and delaunay triangulator, in: Lin, M.C., Manocha, D. (Eds.), Applied Computational Geometry Towards Geometric Engineering, Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 203–222.
Si, H., 2015. TetGen, a Delaunay-based quality tetrahedral mesh generator. ACM Trans. Math. Softw. 41, 11:1–11:36.
Sukumar, N., Malsch, E.A., 2006. Recent advances in the construction of polygonal finite element interpolants. Archives of Computational Methods in Engineering 13, 129–163.
Tao, J., Zhang, J., Deng, B., Fang, Z., Peng, Y., He, Y., 2018. Parallel and scalable heat methods for geodesic distance computation. arXiv e-prints arXiv:1812.06060.
Wachspress, E.L., 1975. A Rational Finite Element Basis. Elsevier.
Wang, H., 2015. A chebyshev semi-iterative approach for accelerating projective and position-based dynamics. ACM Trans. Graph. 34, 246:1–246:9.
Weber, O., Ben-Chen, M., Gotsman, C., 2009. Complex barycentric coordinates with applications to planar shape deformation. Computer Graphics Forum 28, 587–597.
Weber, O., Poranne, R., Gotsman, C., 2012. Biharmonic coordinates. Comput. Graph. Forum .
Xu, W., Zhou, K., Yu, Y., Tan, Q., Peng, Q., Guo, B., 2007. Gradient domain editing of deforming mesh sequences. ACM Trans. Graph. 26.
Yu, Y., Zhou, K., Xu, D., Shi, X., Bao, H., Guo, B., Shum, H.Y., 2004. Mesh editing with poisson-based gradient field manipulation. ACM Trans. Graph. 23, 644–651.
Zhang, J., Deng, B., Liu, Z., Patanè, G., Bouaziz, S., Hormann, K., Liu, L., 2014. Local barycentric coordinates. ACM Trans. Graph. 33, 188:1–188:12.