Maximum likelihood coordinates

Qingjun Chang · Chongyang Deng · Kai Hormann

Abstract

Any point inside a d-dimensional simplex can be expressed in a unique way as a convex combination of the simplex's vertices, and the coefficients of this combination are called the barycentric coordinates of the point. The idea of barycentric coordinates extends to general polytopes with n vertices, but they are no longer unique if $n > d+1$. Several constructions of such generalized barycentric coordinates have been proposed, in particular for polygons and polyhedra, but most approaches cannot guarantee the non-negativity of the coordinates, which is important for applications like image warping and mesh deformation. We present a novel construction of non-negative and smooth generalized barycentric coordinates for arbitrary simple polygons, which extends to higher dimensions and can include isolated interior points. Our approach is inspired by maximum entropy coordinates, as it also uses a statistical model to define coordinates for convex polygons, but our generalization to non-convex shapes is different and based instead on the project-and-smooth idea of iterative coordinates. We show that our coordinates and their gradients can be evaluated efficiently and provide several examples that illustrate their advantages over previous constructions.

Citation Info

Journal
Computer Graphics Forum

Volume
42(5), August 2023

Pages
Article e14908
13 pages

Note
Proceedings of SGP

DOI
10.1111/cgf.14908

1 Introduction

Given a polytope Ω in $\mathbb{R}^d$ with $n \geq d+1$ vertices $v_1, \dots, v_n \in \mathbb{R}^d$, a set of functions $\lambda_i \colon \Omega \to \mathbb{R}$, $i = 1, \dots, n$ is called a set of *generalized barycentric coordinates*, if they allow to write any $v \in \Omega$ as an affine combination of the vertices v_i with coefficients $\lambda_i(v)$, that is,

$$\sum_{i=1}^{n} \lambda_i(v) v_i = v, \qquad \sum_{i=1}^{n} \lambda_i = 1. \tag{1}$$

For most applications, it is further indispensable that the coordinate functions λ_i satisfy the *Lagrange property*

$$\lambda_i(v_j) = \delta_{i,j}, \qquad i, j = 1, \dots, n, \tag{2}$$

since this implies that any data $f_1, \dots, f_n$ given at the vertices of Ω can be interpolated by the function $f(v) = \sum_{i=1}^{n} \lambda_i(v) f_i$. Another desirable property is that the generalized barycentric coordinates should be *non-negative* for any $v \in \Omega$,

$$\lambda_i(v) \geq 0, \qquad i = 1, \dots, n, \tag{3}$$

so that the interpolated values $f(v)$ are guaranteed to be inside the convex hull of the data. Moreover, all $\lambda_i(v)$ should depend smoothly both on v and the vertices v_i, and reduce to k-dimensional generalized barycentric coordinates, if restricted to the k-dimensional faces of Ω. In particular, they should be linear over the edges of a planar polygon and over the faces of a triangle mesh.

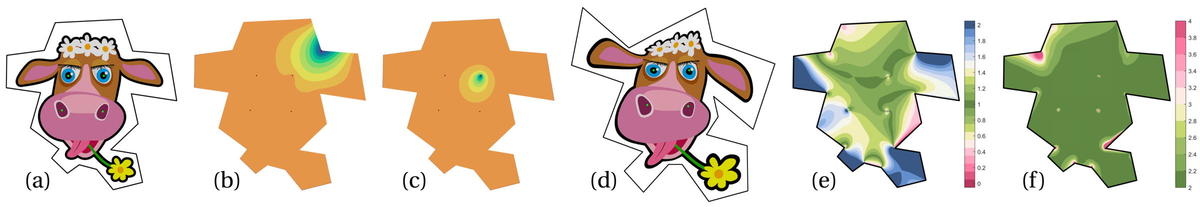


Figure 1: Maximum likelihood coordinates can be defined for arbitrary simple polygons with interior points (b, c). The explicit form of their gradient allows to compute the area and angle distortion (e, f) of the deformation of an image obtained by moving the polygon vertices (a, d).

Generalized barycentric coordinates have numerous applications, including geometric modelling [36, 30], mesh parameterization [11, 12], morphing [14], colour interpolation [39], rendering [23], image warping [49], image cloning [10], mesh deformation [27, 25, 33, 31, 34, 46, 45], finite element methods [47, 37, 18], and many more [22].

1.1 Related work

Barycentric coordinates were discovered by Möbius [41], who not only showed that they are unique if Ω is a d-dimensional simplex, but also derived an explicit formula for these simplex coordinates. Starting with the work of Kalman [29] and Wachspress [47], the idea of barycentric coordinates has been extended to non-convex polygons and polytopes, and many kinds of generalized barycentric coordinates were proposed. A good overview can be found in the surveys by Floater [13] and Anisimov [1].

The different constructions of generalized barycentric coordinates can mainly be divided in two groups. On the one hand, there are *closed-form* constructions, which provide the coordinates in terms of an algebraic expression that can be evaluated efficiently for any $v \in \Omega$. Particularly simple formulas are known for Wachspress [39] and discrete harmonic coordinates [42, 9]. These coordinates are well-defined only for convex polygons and convex polytopes [48, 49, 28, 26] and turn out to be special cases of a whole family of three-point coordinates [15, 2]. This family also includes mean value coordinates [12, 20], which come with the advantage of being well-defined for non-convex polygons and polyhedra, too [16, 27]. Whole families of barycentric coordinates for non-convex polygons and polyhedra were constructed by [7] and [50], but just like metric [37], Poisson [32], and Gordon–Wixom coordinates [6], they may take on negative values at certain $v \in \Omega$. Some constructions guarantee the non-negativity of the coordinates, but at the price of not depending smoothly on either $v \in \Omega$ [33, 38] or the vertices v_i [3]. An exception are iterative coordinates [8], which modify mean value coordinates iteratively until they are non-negative, which is proven to be the case after a finite number of iterations. However, the number of required iterations is not known *a priori* for a given polygon and may be on the order of $O(n^2)$, so that the evaluation of these coordinates becomes very slow.

On the other hand, there are *computational* constructions, where the coordinates are defined as the solution of a non-linear optimization problem with conditions (1), (2), and (3) as constraints. The resulting coordinates have all desired properties, but must be treated numerically. This includes harmonic [25] and local barycentric coordinates [51, 44], which are usually approximated by piecewise linear functions over a dense triangulation T of Ω and require to solve a *global* problem to determine the values of the functions at the vertices of T. Another construction from this category are maximum entropy coordinates [21], which can be evaluated at any $v \in \Omega$ by solving a *local* convex optimization problem, but this approach relies on the choice of certain prior functions, and it is not clear how to choose the latter for a given polygon, so that shape artefacts in the coordinates are avoided.

1.2 Contribution

We propose a novel construction of non-negative and smooth computational barycentric coordinates. Like maximum entropy coordinates (MEC), they are derived from a statistical model, but instead of maximizing the entropy of a discrete probability distribution, we propose to maximize the likelihood. In their basic form (Section 2), these *maximum likelihood coordinates* (MLC) satisfy (1) and are non-negative over the convex hull of the vertices v_i, but they satisfy the Lagrange property (2) only at the corners of the convex hull, that is, at the vertices of a convex polygon or polytope, akin to MEC with constant or Gaussian priors [43]. Another similarity with MEC is that MLC can be computed efficiently at any $v \in \operatorname{Int}\Omega$ after finding the minimum of a convex function in d variables with few iterations of Newton's method. As for MEC [40], this minimum can also be used to determine the gradients of the coordinates at v with a simple formula (Section 2.1).

To extend the construction of MLC to non-convex polygons, we borrow the project-and-smooth idea from iterative coordinates [8] (Sections 3.1 and 3.2), and we explain how to include interior points (Section 3.4) and how to generalize them to higher dimensions (Section 3.5). Our comparisons (Section 4) show that MLC outperform other coordinates in the context of image deformation and that the results get even better when using a novel scaling approach that takes the interior distances from $v \in \operatorname{Int}\Omega$ to the vertices v_i into account and improves the shape of the coordinate functions (Section 4.1). We conclude by discussing limitations and future work (Section 5).

2 Convex polygons

Let us start with the case when Ω is a planar convex polygon with n vertices $v_1,\ldots,v_n \in \mathbb{R}^2$. For any $v \in \operatorname{Int}\Omega$, we define the barycentric coordinates $\lambda = \lambda(v) \in \mathbb{R}^n$ by maximizing

$$\mathcal{L}(\lambda) = \prod_{i=1}^{n} \lambda_i, \tag{4}$$

subject to the constraints

$$\sum_{i=1}^{n} \lambda_i = 1, \qquad \sum_{i=1}^{n} \lambda_i v_i = v, \qquad \lambda_i \ge 0, \quad i = 1,\ldots,n. \tag{5}$$

The objective function $\mathcal{L}$ is motivated by the shape of *likelihood* functions which occur in statistics. To solve this nonlinear optimization problem, we first observe that $\mathcal{L}(\lambda) > 0$ if all coordinates λ_i are positive, while $\mathcal{L}(\lambda) = 0$ if at least one λ_i vanishes. Therefore, we can focus on the set of positive coordinates and maximize instead of $\mathcal{L}(\lambda)$ its logarithm,

$$\ell(\lambda) = \log \mathcal{L}(\lambda) = \sum_{i=1}^{n} \log \lambda_i, \tag{6}$$

because the logarithm is strictly increasing. This is somewhat simpler, because ℓ is strictly concave over the convex set of positive coordinates that satisfy the constraints in (5). Therefore, maximizing ℓ is equivalent to determining a constrained local extremum of ℓ, which in turn can be found with the method of Lagrange multipliers. The latter states that a local extremum of ℓ under the constraints in (5) is characterized by the existence of some $\phi_0 \in \mathbb{R}$ and $\phi = (\phi_1, \phi_2)^{\mathsf{T}} \in \mathbb{R}^2$, such that

$$\frac{\partial}{\partial \lambda_i} \ell(\lambda) = \frac{1}{\lambda_i} = \phi_0 + \phi^{\mathsf{T}}(v_i - v), \quad i = 1,\ldots,n.$$

Multiplying both sides of this identity by λ_i, summing over i, and using (5), we get

$$n = \sum_{i=1}^{n} \lambda_i(\phi_0 + \phi^{\mathsf{T}}(v_i - v)) = \phi_0 \sum_{i=1}^{n} \lambda_i + \phi^{\mathsf{T}} \sum_{i=1}^{n} \lambda_i (v_i - v) = \phi_0,$$

so that

$$\lambda_i = \frac{1}{n + \phi^{\mathsf{T}}(v_i - v)}, \quad i = 1,\ldots,n. \tag{7}$$

It remains to find ϕ, such that the second constraint in (5) holds, that is,

$$\sum_{i=1}^{n} \lambda_i (v_i - v) = \sum_{i=1}^{n} \frac{v_i - v}{n + \phi^{\mathsf{T}}(v_i - v)} = 0. \tag{8}$$

This condition, however, is equivalent to finding a stationary point of the function

$$F(\phi) = -\sum_{i=1}^{n} \log\bigl(n + \phi^{\mathsf{T}}(v_i - v)\bigr), \tag{9}$$

defined over the set

$$\Phi = \bigl\{\phi \in \mathbb{R}^2 : n + \phi^{\mathsf{T}}(v_i - v) > 0, i = 1,\ldots,n\bigr\}$$

of all ϕ that yield positive coordinates λ_i by (7). Note that Φ is the interior of the *polar dual* [28] of Ω with respect to v, scaled by $-n$, and therefore bounded and convex. Since F is strictly convex and diverges to ∞ at the boundary of Φ, it is clear that the unique stationary point of F is at the global minimum of F. Hence, instead of solving the original constrained nonlinear optimization problem in n variables λ_i, we just have to minimize a convex function in the two variables ϕ_1, ϕ_2, which can be done efficiently with few iterations of Newton's method, using $\phi^{(0)} = (0,0)^{\mathsf{T}} \in \Phi$ as initial guess (see Algorithm 1 in Appendix A.1). Once the optimal $\phi = \phi_\star$ is found, the *basic maximum likelihood coordinates* (BMLC) λ_i are computed using (7).

Like all barycentric coordinates that are non-negative over $\operatorname{Int}\Omega$, BMLC have a unique continuous extension to $\partial\Omega$, which is linear along the edges of Ω and satisfies the Lagrange property [15, Corollary 2.3]. Hence, as $v \in \operatorname{Int}\Omega$ converges to $v_\star \in \partial\Omega$, say $v_\star = (1-\mu)v_j + \mu v_{j+1}$ for some j and $0 \le \mu \le 1$, the BMLC of v converge to $\lambda_j = 1 - \mu$, $\lambda_{j+1} = \mu$, and $\lambda_i = 0$ for $i \ne j, j+1$.

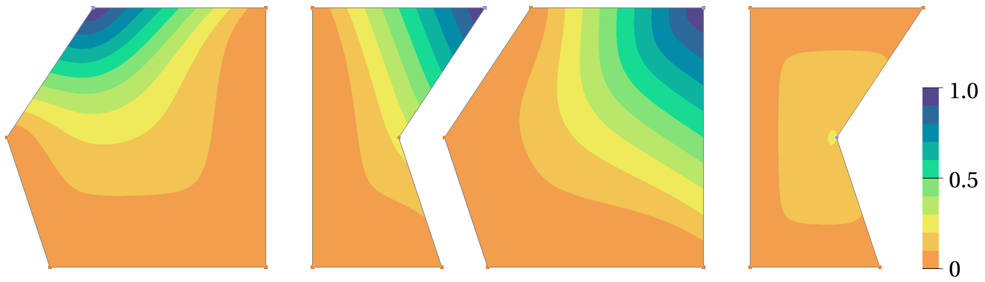


Figure 2: Examples of BMLC for a convex and a concave polygon.

2.1 Gradients

Since F and thus also the minimum of F depend smoothly on v, BMLC are smooth (i.e., C^∞) over $\operatorname{Int}\Omega$, and we can even determine their gradients without too much effort (see Algorithm 2 in Appendix A.1). To this end, let $\phi_\star = \phi(v)$ denote the minimum of F, as a function of v, and recall from (8) that

$$G(\phi, v) = \sum_{i=1}^{n} \frac{v_i - v}{n + \phi^\top (v_i - v)}$$

vanishes at $(\phi_\star, v)$ for any $v \in \operatorname{Int}\Omega$. Hence, the derivative of $G(\phi(v), v)$ with respect to v vanishes, too, and we get, by the multivariate chain rule,

$$\frac{\mathrm{d}}{\mathrm{d}v} G(\phi(v), v) = \frac{\partial}{\partial \phi} G(\phi_\star, v) \frac{\mathrm{d}}{\mathrm{d}v} \phi(v) + \frac{\partial}{\partial v} G(\phi_\star, v) = 0, \tag{10}$$

where

$$\frac{\partial}{\partial \phi} G(\phi, v) = \sum_{i=1}^{n} \frac{-(v_i - v)(v_i - v)^\top}{(n + \phi^\top (v_i - v))^2},$$
$$\frac{\partial}{\partial v} G(\phi, v) = \sum_{i=1}^{n} \frac{(v_i - v)\phi^\top - (n + \phi^\top (v_i - v)) I_2}{(n + \phi^\top (v_i - v))^2},$$

with I_k denoting the k-dimensional identity matrix. Likewise, considering the formula for λ_i in (7) as a function of ϕ and v, we have

$$\frac{\mathrm{d}}{\mathrm{d}v} \lambda_i(\phi(v), v) = \frac{\partial}{\partial \phi} \lambda_i(\phi_\star, v) \frac{\mathrm{d}}{\mathrm{d}v} \phi(v) + \frac{\partial}{\partial v} \lambda_i(\phi_\star, v), \tag{11}$$

where

$$\frac{\partial}{\partial \phi} \lambda_i(\phi, v) = \frac{-(v_i - v)^\top}{(n + \phi^\top (v_i - v))^2}, \qquad \frac{\partial}{\partial v} \lambda_i(\phi, v) = \frac{\phi^\top}{(n + \phi^\top (v_i - v))^2}.$$

Solving (10) for $\frac{\mathrm{d}}{\mathrm{d}v}\phi(v)$ and substituting the result in (11), we finally get, after some simplifications,

$$\nabla \lambda_i = \left(\frac{\mathrm{d}}{\mathrm{d}v} \lambda_i(\phi(v), v) \right)^\top = \lambda_i^2 \big(\phi_\star - G_v G_\phi^{-1} (v_i - v) \big), \tag{12}$$

where the two 2×2 matrices

$$G_\phi = \sum_{j=1}^{n} \lambda_j^2 (v_j - v)(v_j - v)^\top, \qquad G_v = \sum_{j=1}^{n} \lambda_j^2 \phi_\star (v_j - v)^\top - I_2$$

do not depend on i, so that $G_v G_\phi^{-1}$ needs to be computed only once for every $v \in \operatorname{Int}\Omega$. Note that G_ϕ is invertible, because $G_\phi = U^\top U$, where the rows of the matrix $U \in \mathbb{R}^{n\times 2}$ are $\lambda_i (v_i - v)^\top$, $i = 1, \dots, n$. Since these are the vertices of a non-degenerate polygon, it follows that G_ϕ is the *Gram matrix* of two linearly independent vectors $X, Y \in \mathbb{R}^n$, namely the two columns of $U = (X, Y)$.

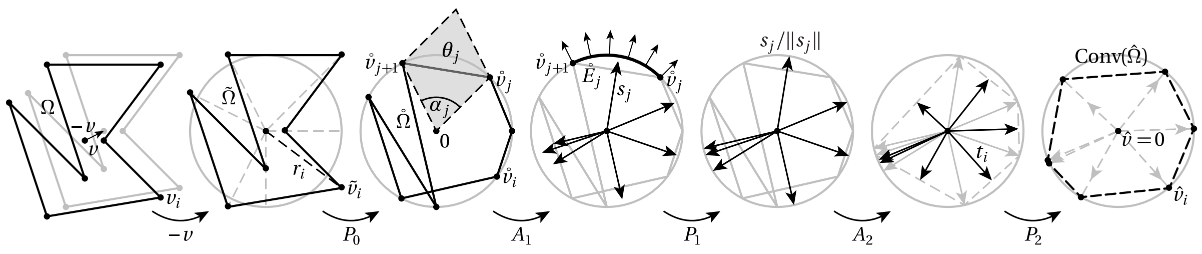


Figure 3: Individual steps of transforming Ω into $\hat\Omega$. To get the Lagrange property, we translate Ω by $-v$ and project the vertices to the unit circle (P_0). To get the linearity along the edges, we further smooth $\mathring\Omega$ with two averaging steps (A_1, A_2) and subsequent projections (P_1, P_2).

2.2 Higher dimensions

The formulation of BMLC naturally extends to convex polytopes in $\mathbb{R}^d$. The formula for the coordinates in (7) still applies, except that ϕ is d-dimensional in general and must be found by minimizing the d-variate analogue of the function F in (9). Likewise, the gradient of λ_i can be computed as above, but with $d \times d$ matrices G_ϕ and G_v.

3 Non-convex polygons

While BMLC satisfy all the key properties of generalized barycentric coordinates if Ω is convex and are well-defined and non-negative over the convex hull Conv(Ω) of a concave polygon, they loose the Lagrange property at the vertices and the linearity along the edges that do not belong to Conv(Ω) (see Figure 2). A similar behaviour is known for maximum entropy coordinates (MEC) with constant or Gaussian priors [43]. For MEC, this can be fixed by using special edge-aware prior functions [21], but we need a different strategy for extending BMLC to non-convex polygons. Our extension is based on the following procedure, which is inspired by the construction of iterative coordinates [8].

Let $V = (v_1, \dots, v_n) \in \mathbb{R}^{2\times n}$ be the matrix whose columns are the vertices of a non-convex polygon Ω. Given $v \in \operatorname{Int}\Omega$, we first translate Ω by $-v$ to get the polygon $\tilde\Omega$ with vertices $\tilde V = V - ve^\top$, where $e = (1, \dots, 1)^\top \in \mathbb{R}^n$. Since barycentric coordinates are invariant under translations, any barycentric coordinates of the origin w.r.t. $\tilde\Omega$ are also barycentric coordinates of v w.r.t. Ω and vice versa. We then apply some non-negative matrix $M \in \mathbb{R}^{n\times n}$, possibly depending on v and $v_1, \dots, v_n$, with at least one positive entry per row, to get $\hat\Omega$ with vertices $\hat V = \tilde V M$, which can also be seen as transforming the two vectors consisting of the x- and the y-coordinates of the vertices $\tilde v_i$ with $M^\top$. Any barycentric coordinates of the origin w.r.t. $\tilde\Omega$ can then be turned into barycentric coordinates of v w.r.t. Ω by transforming them with M and normalizing the result.

Lemma 1. *Given a simple polygon Ω with vertices V and $v \in \operatorname{Int}\Omega$, let $\hat\Omega$ be the polygon with vertices $\hat V = (V - ve^\top)M$ and $\hat\lambda$ be some positive barycentric coordinates of the origin $\hat v = 0$ with respect to $\hat\Omega$. Then $\lambda = w/W$, where $w = M\hat\lambda$ and $W = w_1 + \dots + w_n$, are positive barycentric coordinates of v with respect to Ω.*

Proof. It follows from the properties of M and $\hat\lambda$ that $w > 0$ and $W = e^\top w > 0$, hence $\lambda > 0$ and $\sum_{i=1}^n \lambda_i = e^\top\lambda = e^\top w/W = 1$. Moreover, since $\sum_{i=1}^n \hat\lambda_i \hat v_i = \hat V\hat\lambda = \hat v = 0$, we have

$$\sum_{i=1}^n \lambda_i v_i = V\lambda = V\frac{w}{W} = \frac{VM\hat\lambda}{W} = \frac{\hat V\hat\lambda}{W} + \frac{ve^\top M\hat\lambda}{W} = v\frac{e^\top w}{W} = v,$$

which concludes the proof. □

Lemma 1 provides a recipe for defining *maximum likelihood coordinates* (MLC) for non-convex polygons. For any $v \in \operatorname{Int}\Omega$,

1. transform Ω into $\hat\Omega$, using a suitable matrix M;
2. compute the BMLC $\hat\lambda$ of the origin w.r.t. $\hat\Omega$;
3. derive $\lambda = \lambda(v)$ from $\hat\lambda$ using again M.

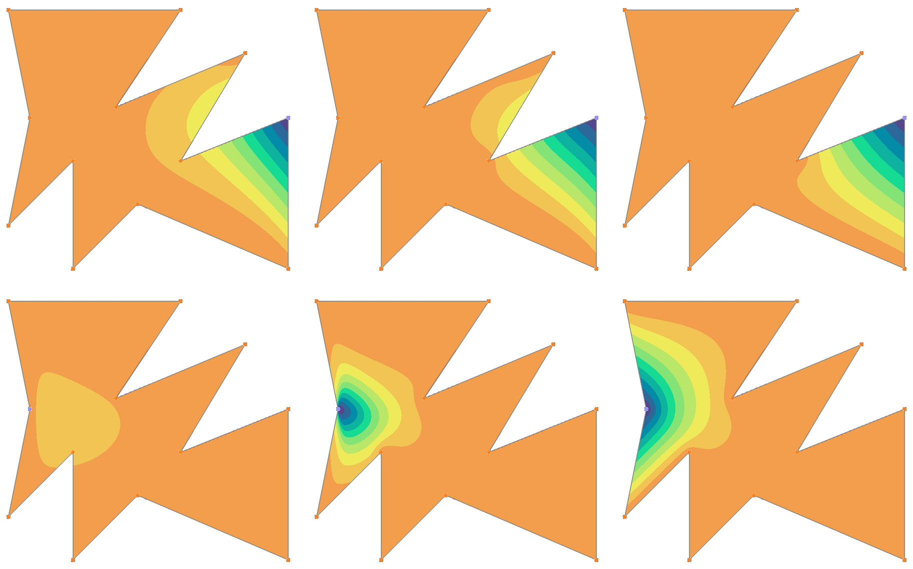


Figure 4: Examples of BMLC (left), MLC with projection (middle), and MLC with projection and smoothing (right).

While this approach guarantees that the coordinates λ are barycentric and positive, the difficult part is finding suitable matrices $M = M(v)$, such that the coordinates have the Lagrange property at the vertices and are piecewise linear along the edges of Ω. Both can be achieved by the project-and-smooth idea of iterative coordinates [8], which are basically defined by the same procedure.

However, iterative coordinates use the circumcentre coordinates of the origin w.r.t. $\hat{\Omega}$ in step 2. These are guaranteed to be positive only if $\hat{\Omega}$ is a convex cyclic polygon, which in turn may require $O(n^2)$ smoothing steps. For MLC, step 2 gives positive coordinates even if $\hat{\Omega}$ is a self-intersecting polygon, as long as $0 \in \operatorname{Conv}(\hat{\Omega})$, because the definition of BMLC actually depends only on the vertices of the polygon, but not on how they are connected by edges.

3.1 Projection

The Lagrange property at the vertices can be restored (see Figure 4) by projecting the translated vertices $\tilde{v}_i$ to the unit circle around the origin (see Figure 3), that is, by using as M the diagonal projection matrix

$$P_0 = \operatorname{diag}\left(\frac{1}{r_1}, \ldots, \frac{1}{r_n}\right),$$

where $r_i = \|\tilde{v}_i\| = \|v_i - v\|$.

Theorem 1. *The maximum likelihood coordinates $\lambda(v)$ defined by $M = P_0$ possess the Lagrange property at the vertices of Ω.*

Proof. If $v \in \operatorname{Int}\Omega$ converges to v_j, then r_j converges to 0, while the other r_k converge to $\|v_k - v_j\| > 0$. Consequently, the ratios r_j/r_k converge to $\delta_{k,j}$ for all $k = 1, \ldots, n$. Therefore,

$$\lim_{v\to v_j} \lambda_i(v) = \lim_{v\to v_j} \frac{\hat{\lambda}_i/r_i}{\sum_{k=1}^n \hat{\lambda}_k/r_k} = \frac{\lim_{v\to v_j}\bigl(\hat{\lambda}_i r_j/r_i\bigr)}{\sum_{k=1}^n \lim_{v\to v_j}\bigl(\hat{\lambda}_k r_j/r_k\bigr)} = \frac{\delta_{i,j}\lim_{v\to v_j}\hat{\lambda}_i}{\sum_{k=1}^n \delta_{k,j}\lim_{v\to v_j}\hat{\lambda}_k} = \delta_{i,j}\frac{\lim_{v\to v_j}\hat{\lambda}_i}{\lim_{v\to v_j}\hat{\lambda}_j} = \delta_{i,j},$$

because all $\hat{\lambda}_k$ are bounded and $\hat{\lambda}_j > 0$, even in the limit. □

3.2 Smoothing

To restore the linearity along the edges (see Figure 4), we apply the following smoothing process to the projected vertices $\mathring{v}_i = \tilde{v}_i/r_i$ (see Figure 3). We first average for each edge $E_j = [v_j, v_{j+1}]$ of Ω the outward-pointing unit normals of the circular arc $\mathring{E}_j$ between $\mathring{v}_j$ and $\mathring{v}_{j+1}$ in an integral sense. Basic calculations

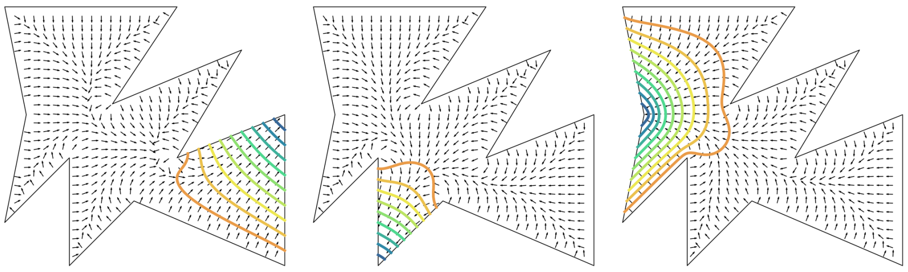


Figure 5: Examples of normalized gradient fields of MLC with projection and smoothing (cf. Figure 4).

reveal that this average can be expressed as a linear combination of the endpoints,

$$s_j = \int_{\mathring{E}_j} x\,\mathrm{d}x \bigg/ \int_{\mathring{E}_j} 1\,\mathrm{d}x = \sigma_{j,1}\mathring{v}_j + \sigma_{j,2}\mathring{v}_{j+1}, \tag{13}$$

with weights

$$\sigma_{j,1} = \sigma_{j,2} = \sigma_j = \frac{\tan\frac{\alpha_j}{2}}{\alpha_j} = \frac{1-\mathring{v}_j^{\mathsf{T}}\mathring{v}_{j+1}}{\alpha_j\theta_j},$$

where α_j is the length of $\mathring{E}_j$ and $\theta_j = \sin\alpha_j$ is the area of the parallelogram spanned by $\mathring{v}_j$ and $\mathring{v}_{j+1}$. Note that $s_j = \mathring{v}_j = \mathring{v}_{j+1}$ in the limit, as α_j converges to 0. Moreover, as α_j tends to π, σ_j diverges to $+\infty$, while all other σ_k converge to finite values, which is crucial for getting MLC that are linear along the edges. We then project the s_j back to the unit circle, average them with their predecessors, and project again. That is, we compute the vertices $\hat{v}_i$ of $\hat{\Omega}$ as

$$\hat{v}_i = t_i/\|t_i\|, \qquad t_i = \big(s_{i-1}/\|s_{i-1}\| + s_i/\|s_i\|\big)/2,$$

where indices are considered cyclically over the range $[1,\dots,n]$. This is equivalent to using as M the matrix $P_0A_1P_1A_2P_2$, where

$$A_1 = \begin{pmatrix} \sigma_{1,1} & 0 & \cdots & \sigma_{n,2} \\ \sigma_{1,2} & \sigma_{2,1} & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & \sigma_{n-1,2} & \sigma_{n,1} \end{pmatrix}, \quad A_2 = \frac{1}{2}\begin{pmatrix} 1 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & 1 \\ 1 & \cdots & 0 & 1 \end{pmatrix},$$

$P_1 = \operatorname{diag}(1/\|s_1\|,\dots,1/\|s_n\|)$, and $P_2 = \operatorname{diag}(1/\|t_1\|,\dots,1/\|t_n\|)$.

Theorem 2. *The maximum likelihood coordinates $\lambda(v)$ defined by $M = P_0A_1P_1A_2P_2$ are linear along the edges of Ω.*

Proof. If $v \in \operatorname{Int}\Omega$ converges to $v_\star = (1-\mu)v_j + \mu v_{j+1}$ for some j and $0 < \mu < 1$, then $\mathring{v}_j$ converges to $-\mathring{v}_{j+1}$ and α_j and θ_j converge to π and 0, respectively. And even though σ_j diverges to $+\infty$, s_j is well-defined, even in the limit, as the halfway vector between $\mathring{v}_j$ and $\mathring{v}_{j+1}$ with length $\|s_j\| = \frac{2}{\alpha_j}\sin\frac{\alpha_j}{2}$. Therefore, all $\|t_k\|$ converge to positive values, so that $\operatorname{Conv}(\hat{\Omega})$ is a well-defined convex cyclic polygon which contains the origin. It follows that $\hat{\lambda} > 0$, even in the limit, and also $u = P_1A_2P_2\hat{\lambda} > 0$.

We further notice that $\theta_j\sigma_j$ converges to $2/\pi$ as v approaches $v_\star$, while $\theta_j\sigma_k$ converges to 0 for $k \neq j$. Consequently, θ_jA_1u converges to the vector $\mathring{w} \in \mathbb{R}^n$ with $\mathring{w}_j = \mathring{w}_{j+1} = 2u_j/\pi$ and $\mathring{w}_k = 0$ for $k \neq j, j+1$. Since

$$\lim_{v\to v_\star}\lambda(v) = \lim_{v\to v_\star}\frac{\theta_jM\hat{\lambda}}{\theta_jEM\hat{\lambda}} = \lim_{v\to v_\star}\frac{P_0(\theta_jA_1u)}{EP_0(\theta_jA_1u)} = \frac{P_0\mathring{w}}{EP_0\mathring{w}},$$

this implies

$$\lim_{v\to v_\star}\lambda_i(v) = \frac{\dfrac{\mathring{w}_i}{r_i}}{\dfrac{\mathring{w}_j}{r_j} + \dfrac{\mathring{w}_{j+1}}{r_{j+1}}} = \begin{cases} \frac{1/\mu}{1/\mu+1/(1-\mu)} = 1-\mu, & i = j, \\ \frac{1/(1-\mu)}{1/\mu+1/(1-\mu)} = \mu, & i = j+1, \\ 0, & i \neq j, j+1, \end{cases}$$

because r_j and r_{j+1} converge to μe and $(1-\mu)e$, respectively, where $e = \|v_{i+1} - v_i\|$. □

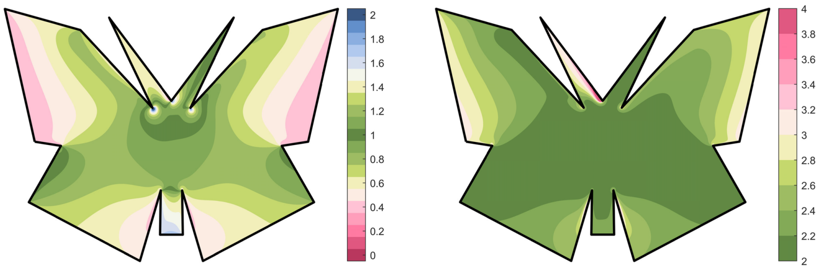


Figure 6: Area and angle distortion of the MLC deformation in Figure 9: Jacobian determinant (left) and MIPS energy (right).

3.3 Gradients

Since all ingredients of the projection and the smoothing process depend smoothly on v, it is clear that MLC with projection and smoothing are smooth over $\operatorname{Int}\Omega$, with a continuous extension to $\partial\Omega$ that is linear along the edges of Ω. The gradients of MLC can be derived, essentially with the same idea as in Section 2.1 and by carefully applying the chain rule several times to take care of the projection and smoothing operators. The details and pseudo code for computing MLC and their gradients can be found in Appendix A.2.

Figure 5 shows three examples of the normalized gradients of MLC with projection and smoothing at those nodes of a regular grid that are inside the non-convex polygon from Figure 4. Moreover, we can use the gradients to compute the distortion of the *barycentric mapping* from a source polygon Ω to a target polygon Ω',

$$f\colon \Omega \to \Omega', \quad f(v) = \sum_{i=1}^{n} \lambda_i(v) v_i', \tag{14}$$

at any point $v \in \operatorname{Int}\Omega$. For example, Figure 6 shows the determinant $\det(J_f)$ of the *Jacobian* matrix

$$J_f(v) = \sum_{i=1}^{n} v_i' \nabla^{\mathsf{T}} \lambda_i(v)$$

of f to visualize the area distortion (no area distortion corresponds to the value 1 and negative values would indicate fold-overs) of the MLC deformation in Figure 9 (note that the source image in Figure 9 was scaled by a factor of 2/3 w.r.t. the target images for layout reasons), as well as the MIPS energy $\operatorname{trace}({J_f}^{\mathsf{T}} J_f)/\det(J_f)$ to illustrate the angle distortion of this mapping (local conformality corresponds to the smallest possible value 2). Another example of such distortion plots can be found in Figure 1.

3.4 Interior points

An interesting feature of the MLC construction is that it allows to include isolated interior points. In fact, Theorem 1 does not depend on the vertices forming an actual polygon Ω and also holds for any set of isolated vertices. Hence, if we define the MLC for a given polygon Ω and m additional interior points $v_{n+1}, \dots, v_{n+m} \in \operatorname{Int}\Omega$ by projecting all vertices as in Section 3.1 and smoothing only the vertices of Ω as in Section 3.2, then we get generalized barycentric coordinates that satisfy all the key properties, except that they are only continuous at the interior points (see Figure 7).

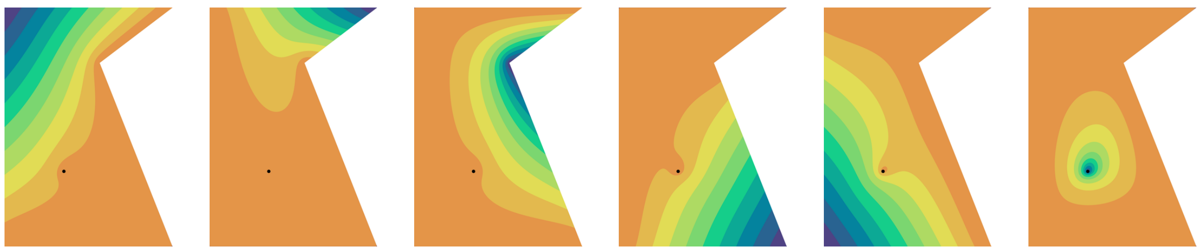


Figure 7: Examples of MLC for a polygon with an interior point.

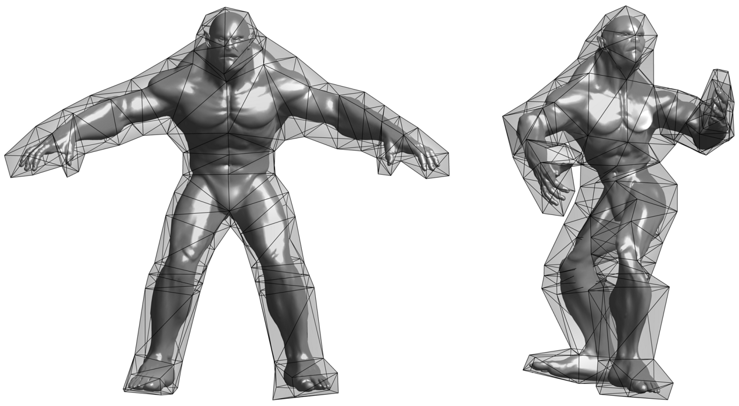


Figure 8: Deformation of a source mesh (left) with 3D MLC (right).

3.5 Higher dimensions

As for convex polygons, the construction of MLC extends to non-convex polytopes in $\mathbb{R}^d$. The generalizations of Lemma 1 and Theorem 1 from 2D to any dimension d is trivial, so let us focus on the smoothing process.

In the first step, we average the outward-pointing unit normals of the hyperspherical simplices that correspond to the simplicial faces of the polytope Ω and express them as a linear combination of the vertices that we get by projecting the faces of Ω. For example, if $T = [v_1, v_2, v_3]$ is a triangle of a polyhedron Ω in $\mathbb{R}^3$ and $\mathring{T}$ is the spherical triangle spanned by the projected vertices $\mathring{v}_1, \mathring{v}_2, \mathring{v}_3$, then we need to find the weights $\sigma_1, \sigma_2, \sigma_3$, such that

$$s = \int_{\mathring{T}} x \,\mathrm{d}x \bigg/ \int_{\mathring{T}} 1 \,\mathrm{d}x = \sigma_1 \mathring{v}_1 + \sigma_2 \mathring{v}_2 + \sigma_3 \mathring{v}_3. \tag{15}$$

With α denoting the area of $\mathring{T}$ and $\theta = |\mathring{v}_1 \cdot (\mathring{v}_2 \times \mathring{v}_3)|$ the volume of the parallelepiped spanned by $\mathring{v}_1, \mathring{v}_2, \mathring{v}_3$, it follows from [16, Theorem 2] that

$$\sigma_1 = \frac{\beta_{2,3} + \beta_{1,2}\, n_{1,2} \cdot n_{2,3} + \beta_{3,1}\, n_{3,1} \cdot n_{2,3}}{2\alpha\theta} \sin\beta_{2,3}$$

and similarly for σ_2 and σ_3, with $\beta_{r,s}$ denoting the (unsigned) angle between $\mathring{v}_r$ and $\mathring{v}_s$ and $n_{r,s} = (\mathring{v}_r \times \mathring{v}_s)/\|\mathring{v}_r \times \mathring{v}_s\|$ being a unit normal of the triangle $[0, \mathring{v}_r, \mathring{v}_s]$. The computation of the s_j for all m triangles $T_j = [v_{j_1}, v_{j_2}, v_{j_3}]$ of Ω can be expressed as $S = (s_1, \dots, s_m) = \mathring{V} A_1$, where the $n \times m$ matrix A_1 has exactly 3 non-zero entries per column, namely $(A_1)_{j_l, j} = \sigma_{j_l}$, for $l = 1, 2, 3$.

In the next step, we average, for each vertex v_i of Ω, the normalized vectors s_j of the m_i faces adjacent to v_i, with indices in N_i,

$$t_i = \frac{1}{m_i} \sum_{j \in N_i} s_j / \|s_j\|,$$

and normalize again to get $\hat{v}_i = t_i / \|t_i\|$. As in Section 3.2, this can be expressed as $\hat{V} = S P_1 A_2 P_2$. For example, if Ω is a polyhedron in $\mathbb{R}^3$, then the $m \times n$ matrix A_2 has the same structure as A_1^T, with 3 non-zero entries per row, namely $(A_2)_{j, j_l} = 1/m_{j_l}$, for $l = 1, 2, 3$.

At least in 3D, the proof of Theorem 2 can be generalized nicely. As $v \in \operatorname{Int}\Omega$ converges to a point inside a triangle $T_j = [v_{j_1}, v_{j_2}, v_{j_3}]$ of Ω, say $v_\star = \mu_1 v_{j_1} + \mu_2 v_{j_2} + \mu_3 v_{j_3}$ for some $\mu_1, \mu_2, \mu_3 > 0$ with $\mu_1 + \mu_2 + \mu_3 = 1$, the area α_j and the volume θ_j converge to 2π and 0, respectively, and while

$$\lim_{v \to v_\star} \theta_j \big(\sigma_{j_1}, \sigma_{j_2}, \sigma_{j_3}\big) = \frac{1}{2} \big(\sin\beta_{j_2, j_3}, \sin\beta_{j_3, j_1}, \sin\beta_{j_1, j_2}\big), \tag{16}$$

all other scaled weights $\theta_j \sigma_{k_l}$ for $k \neq j$ and $l = 1, 2, 3$ converge to 0. Consequently, $\theta_j A_1 P_1 A_2 P_2 \hat{\lambda}$ converges to a vector $\mathring{w} \in \mathbb{R}^n$, whose 3 non-zero entries are the values on the right hand side of (16), scaled by a common positive factor. Since these are, after normalization, the 2D barycentric coordinates of the origin 0 w.r.t. the triangle $\lim_{v \to v_\star} [\mathring{v}_{j_1}, \mathring{v}_{j_2}, \mathring{v}_{j_2}]$, it follows from Lemma 1 with $M = P_0$ that the MLC coordinates λ converge to the barycentric coordinates μ_1, μ_2, μ_3 of v w.r.t. T_j and are thus linear along the faces of Ω.

An example of a cage-based deformation using 3D MLC with projection and smoothing is shown in Figure 8.

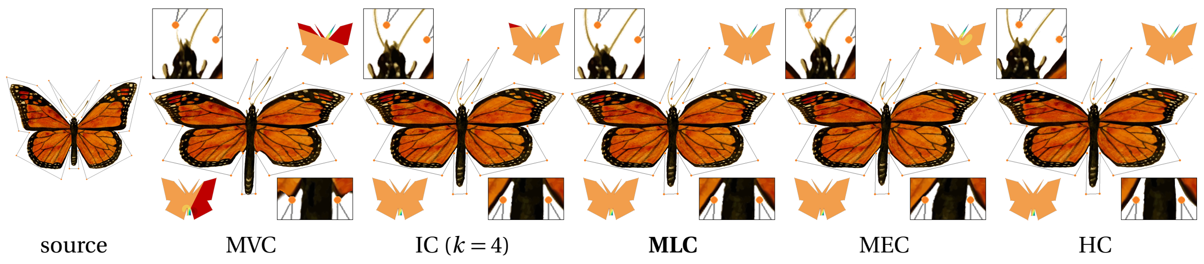


Figure 9: Deformation of a source image (left) with different barycentric coordinates, zoom-ins, and examples of basis functions (insets). While the MVC-based deformation has fold-overs, caused by negative functions values (bottom), and the MEC-based deformation has distortions, related to local maxima of some coordinate functions (top and bottom), the results using IC, MLC, and HC are free of artefacts.

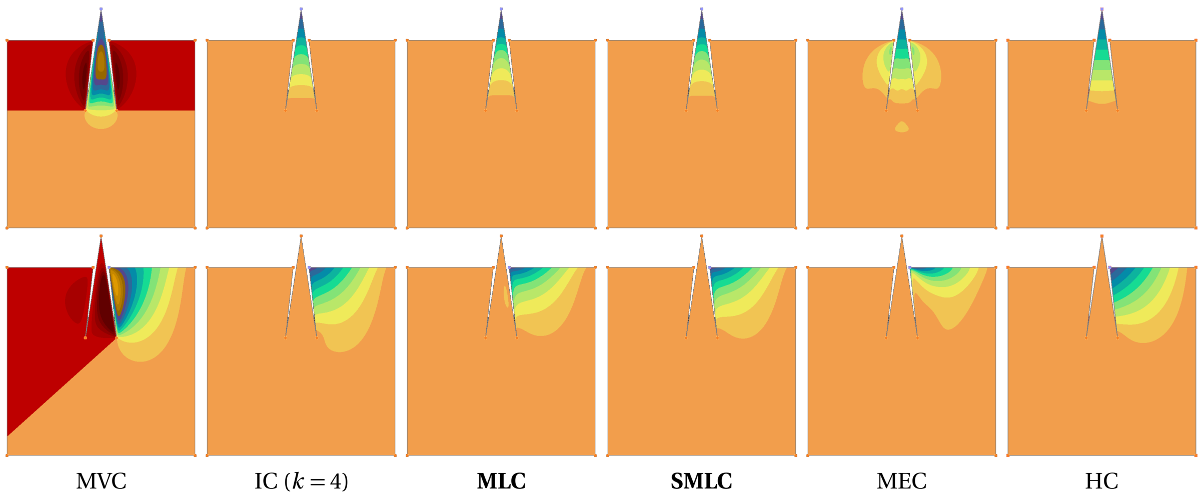


Figure 10: Comparison of different barycentric coordinate functions for a polygon with two very sharp concave vertices.

4 Results

One of the main applications of 2D barycentric coordinates is image deformation, which can be achieved by enclosing the source image with a source polygon Ω and then moving its vertices $v_1, \dots, v_n$ to get a target polygon Ω' with vertices $v'_1, \dots, v'_n$. The deformed target image is then generated by applying the barycentric mapping f in (14). Due to the properties of the barycentric coordinates λ, the mapping f is interpolatory, that is, $f(v_i) = v'_i$ for $i = 1, \dots, n$, and it linearly maps the edges of Ω to the edges of Ω'.

It is well-known that negative coordinates can lead to artefacts in the deformed image. For example, the negative values (in red) of the mean value coordinates (MVC) associated with the two vertices near the tail of the butterfly in Figure 9 cause the target image to fold over (bottom insets). For iterative coordinates (IC), both the negative values and the fold-overs disappear after 4 iterations. However, unexpected deformation results may also stem from shape artefacts of the coordinate functions. For example, the maximum entropy coordinates (MEC) associated with the two vertices near the tips of the butterfly's antennas have a local maximum of significant height (~ 0.13 in this case) inside the front wing, which lead to artefacts (top insets) that are not present in the MLC-based deformation, which in turn is similar in quality to the deformation obtained using harmonic coordinates (HC).

A more extreme example of this phenomenon can be seen in Figure 10. The MVC functions corresponding to vertices next to a concave vertex with a big interior angle (close to 2π) can have big local maxima (greater than 1, in brown) that lead to very negative local minima (in red) of other coordinate functions and may cause unwanted deformation results. These artefacts disappear for IC after 4 iterations. Local maxima (smaller than 1) also happen for MEC (top), but instead of causing local minima, they induce severe shape artefacts of neighbouring coordinate functions (bottom).

One explanation for this problem is that the prior functions, which are used to guarantee the linearity of MEC along the edges of Ω, are normalized products of functions ρ_i that are zero on the edge E_i of Ω and grow

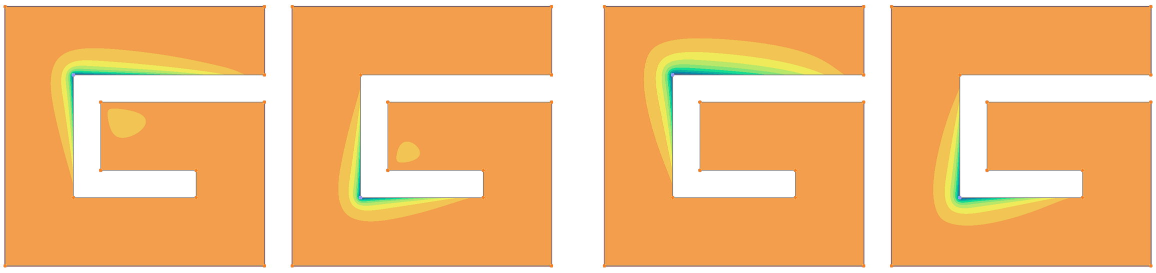


Figure 11: Comparison of MLC (left) and SMLC (right) associated with a concave vertex of a non-convex polygon.

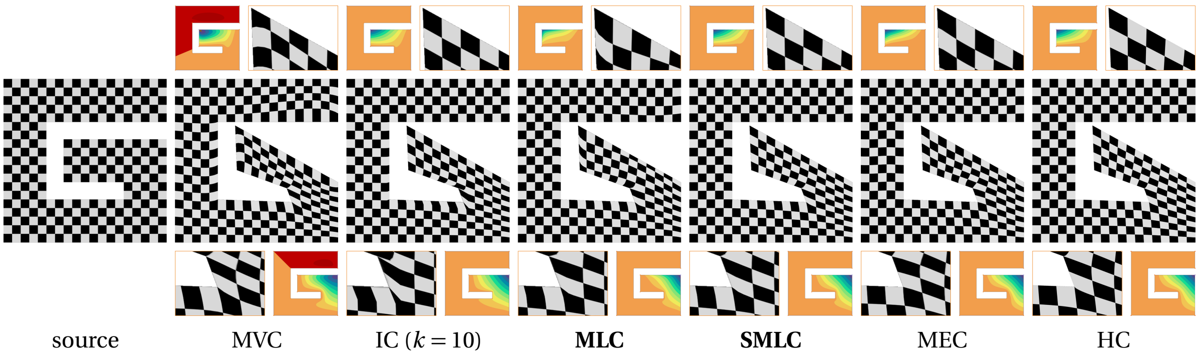


Figure 12: Deformation of a source image (left) with different barycentric coordinates, zoom-ins, and examples of basis functions (insets). The top row shows artefacts caused by basis functions with too much (MVC) or too little (MLC) local influence, which disappear after few iterations (IC) or with additional scaling (SMLC), respectively. Instead, the artefact of the MVC-based deformation in the bottom row is amplified as the number of iterations grows, and the scaling step is not as effective in improving the result of the MLC-based deformation.

with increasing distance to E_i, independently of the other edges. Due to this locality, MEC are not "aware" of the global shape of Ω. For MLC, the situation is similar, but to a lesser extent, and it can be improved with a simple, yet effective approach.

4.1 Scaling

To enrich MLC with a certain global shape awareness, we add another scaling step to the project-and-smooth procedure from Section 3. These *scaled maximum likelihood coordinates* (SMLC) are defined by using the matrix $M = P_0 A_1 P_1 A_2 P_2 D$ in the basic MLC recipe, where $D = \operatorname{diag}(1/d_1, \dots, 1/d_n)$ with d_i denoting some interior distance between v and v_i. In our examples, we used biharmonic distances [35], but also the geodesic distance inside Ω [5] could be used. Note that the additional scaling of the projected and smoothed vertices does not change the fact that $0 \in \operatorname{Conv}(\hat{\Omega})$, so that we can still compute valid BMLC $\hat{\lambda}$ of the origin w.r.t. $\hat{\Omega}$.

Using the same arguments as before, it is clear that SMLC are as smooth as the functions used to compute the distances d_i, and as long as the derivatives of these distances can be evaluated in some way, then we can also compute the derivatives of SMLC, with minor changes to the code provided in Appendix A.2.

Figure 11 shows that the scaling approach is effective in removing local maxima, or at least in significantly decreasing their magnitude. It also reduces another shape artefact that some MLC functions may have, namely falling off very quickly in the vicinity of the associated vertex, which in turn implies too little local impact of these coordinate functions. As shown in Figure 12 (top insets), this can lead to artefacts in MLC-based deformations, which disappear when using SMLC. MVC functions associated with concave vertices also tend to suffer from this phenomenon. This causes other neighbouring coordinate functions to grow too quickly close to concave vertices and thus having too much impact on the deformation. The resulting artefacts (bottom insets in Figure 12) get even worse when using IC with an increasing number of iterations. While the MLC-based deformation has similar problems, which are improved only marginally by the additional scaling step, the deformation using MEC may be considered more natural in this case. Compared to the HC-based deformation, we observe that the result using SMLC is the most similar among all shown deformations.

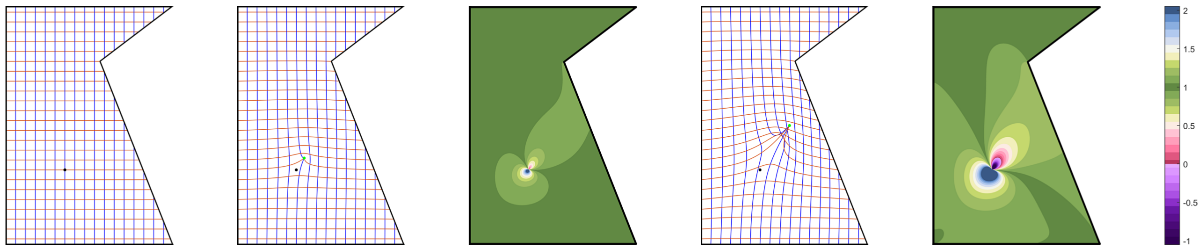


Figure 13: Deformation of a source image (left) with MLC for a polygon with an interior point (cf. Figure 7). The Jacobian determinant plots indicate the lack of C^1 continuity at the interior point, and fold-overs may occur if the displacement is too large (right).

4.2 Interior points

In the context of image deformation, the possibility to add interior points to the polygon (see Section 3.4) can be useful for *constrained* deformation, where certain points in the interior of the image should not be affected by the deformation (see Figure 1). However, as MLC are not C^1 at interior points, so neither is the deformation, which can lead to visual artefacts, except if the point constraints are placed inside a region of constant colour, as in Figure 1.

Moreover, interior points do not provide an effective *control handle*, since moving interior points can lead to fold-overs in the deformation, unless the displacement is small (see Figure 13). A similar behaviour is known for HC-based deformation [24, Figure 9].

5 Conclusion

Despite the intense research on generalized barycentric coordinates over the last two decades, few constructions give smooth and non-negative coordinates that are well-defined for arbitrary simple polygons, and with MLC we introduce a new animal to this zoo that comes with several advantages, but also some limitations. A comparison of the properties of different 2D constructions can be found in Appendix B.

In contrast to harmonic and local barycentric coordinates, MLC and their gradients can be evaluated efficiently with arbitrary accuracy at any point inside the polygon by solving a local convex optimization problem. In this respect, MLC are very similar to MEC and even better, since the gradient and the Hessian of the function F in (9), which are needed for minimizing F with Newton's method, are considerably simpler than those needed for computing MEC. However, there is one important *conceptual* difference between MLC and MEC.

The construction of MEC for non-convex polygons relies on the choice of suitable prior functions and both options (MEC-1 and MEC-2) presented in [21] suffer from lack of global shape awareness, which can cause severe deformation artefacts (see Figure 9). Note that we used MEC-1 in our examples, because the results obtained with MEC-2 were worse. In the case of MLC, the extension to non-convex polygons is based on the project-and-smooth idea from iterative coordinates, which seems better suited for creating natural-looking deformations and can easily be extended to become more aware of the global shape of the polygon. However, while the scaling step proposed in Section 4.1 effectively reduces unwanted local maxima of the coordinate functions, it does not remove all shape artefacts (see Figure 12), and future work is needed for refining and improving this approach.

Compared to iterative coordinates [8], one major advantage of MLC is speed. In our experiments, we observed that MLC are as expensive to compute as IC with $k = 4$ iterations, but for certain polygons, $k \in O(n^2)$ iterations are needed to guarantee the non-negativity of IC (see Figure 8 in [8]). Moreover, we are not aware of any algorithm for computing the gradients of IC, a problem that appears to increase in difficulty with the number of iterations.

Another advantage of our construction regards the extension to higher dimensions. While our smoothing procedure in Section 3.2 is essentially identical (up to an index shift) to a double-smoothing step of IC, expressing and viewing it the way we do (a first averaging step to get vectors s_j associated with edges and a second averaging step to get back to vectors t_i associated with vertices), renders the extension to 3D and beyond straightforward. However, for $d \geq 4$ it remains future work to work out concrete formulas for the weights $\sigma_1, \ldots, \sigma_d$ that allow us to express the average of the outward-pointing unit normals of a general hyperspherical simplex in terms of a linear combination of the projected vertices $\mathring{v}_1, \dots, \mathring{v}_d$ as in (13) and (15).

On the theoretical side, our construction reveals that the key steps for constructing non-negative barycentric coordinates that satisfy the Lagrange property and are linear along the edges are the translation by $-v$, the projection P_0 onto the unit circle, and the first averaging step A_1. In principle, we can use the basic MLC recipe with $M = P_0 A_1 \hat{M}$ for any non-negative matrix $\hat{M}$ and using any scheme for computing non-negative barycentric coordinates $\hat{\lambda}$ of the origin w.r.t. the points $\hat{V} = (V - ve^\top)M = \mathring{V} A_1 \hat{M}$. For example, one could use MEC without priors for the latter, or consider non-uniform averaging methods to replace our second averaging step A_2.

Another interesting line of future research relates to the rather simple and explicit expressions for the gradients, especially in the case of BMLC for convex polygons. A deeper analysis, similar to the one in [17], may reveal conditions for guaranteeing the injectivity of the barycentric mapping f in (14).

One inherent limitation of MLC is that they are only continuous across the boundary of the polygon and not defined outside the convex hull of the polygon. While this is sufficient for cage-based deformation, it rules out skeleton-based deformation as in [50]. Another limitation is the lack of smoothness at interior points, and it remains future work to find generalized barycentric coordinates that are at least C^1 at interior points.

Acknowledgements

We thank the anonymous reviewers for their valuable comments and suggestions, which helped to improve this paper. This work was supported by the Swiss National Science Foundation (SNF) under project number 188577 and the National Natural Science Foundation of China (NSFC) under project number 61872121.

References

[1] D. Anisimov. Barycentric coordinates and their properties. In Hormann and Sukumar [22], chapter 1, pages 3–22.

[2] D. Anisimov, K. Hormann, and T. Schneider. Behaviour of exponential three-point coordinates at the vertices of convex polygons. *Journal of Computational and Applied Mathematics*, 350:114–129, Apr. 2019. [PDF]

[3] D. Anisimov, D. Panozzo, and K. Hormann. Blended barycentric coordinates. *Computer Aided Geometric Design*, 52–53:205–216, Mar.–Apr. 2017. [PDF]

[4] L. Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics*, 16(1):1–3, Nov. 1966.

[5] B. Aronov. On the geodesic Voronoi diagram of point sites in a simple polygon. *Algorithmica*, 4(1):109–140, 1989.

[6] A. Belyaev. On transfinite barycentric coordinates. In A. Sheffer and K. Polthier, editors, *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, pages 89–99, Cagliari, June 2006.

[7] M. Budninskiy, B. Liu, Y. Tong, and M. Desbrun. *Power coordinates*: a geometric construction of barycentric coordinates on convex polytopes. *ACM Transactions on Graphics*, 35(6):Article 241, 11 pages, Nov. 2016. Proceedings of SIGGRAPH Asia.

[8] C. Deng, Q. Chang, and K. Hormann. Iterative coordinates. *Computer Aided Geometric Design*, 79:Article 101861, 13 pages, May 2020. [PDF]

[9] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In S. G. Mair and R. Cook, editors, *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 173–182, Los Angeles, Sept. 1995.

[10] Z. Farbman, G. Hoffer, Y. Lipman, D. Cohen-Or, and D. Lischinski. Coordinates for instant image cloning. *ACM Transactions on Graphics*, 28(3):Article 67, 9 pages, Aug. 2009. Proceedings of SIGGRAPH.

[11] M. S. Floater. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14(3):231–250, Apr. 1997.

[12] M. S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20(1):19–27, Mar. 2003.

[13] M. S. Floater. Generalized barycentric coordinates and applications. *Acta Numerica*, 24:161–214, May 2015.

[14] M. S. Floater and C. Gotsman. How to morph tilings injectively. *Journal of Computational and Applied Mathematics*, 101(1–2):117–129, Jan. 1999.

[15] M. S. Floater, K. Hormann, and G. Kós. A general construction of barycentric coordinates over convex polygons. *Advances in Computational Mathematics*, 24(1–4):311–331, Jan. 2006. [PDF]

[16] M. S. Floater, G. Kós, and M. Reimers. Mean value coordinates in 3D. *Computer Aided Geometric Design*, 22(7):623–631, Oct. 2005.

[17] M. S. Floater and J. Kosinka. On the injectivity of Wachspress and mean value mappings between convex polygons. *Advances in Computational Mathematics*, 32(2):163–174, Feb. 2010.

[18] A. Gillette, A. Rand, and C. Bajaj. Constructions of scalar and vector finite element families on polygonal and polyhedral meshes. *Computer Methods in Applied Mechanics and Engineering*, 16(4):667–683, Oct. 2016.

[19] W. J. Gordon and J. A. Wixom. Pseudo-harmonic interpolation on convex domains. *SIAM Journal on Numerical Analysis*, 11(5):909–933, 1974.

[20] K. Hormann and M. S. Floater. Mean value coordinates for arbitrary planar polygons. *ACM Transactions on Graphics*, 25(4):1424–1441, Oct. 2006. [PDF]

[21] K. Hormann and N. Sukumar. Maximum entropy coordinates for arbitrary polytopes. *Computer Graphics Forum*, 27(5):1513–1520, July 2008. Proceedings of SGP. [PDF]

[22] K. Hormann and N. Sukumar, editors. *Generalized Barycentric Coordinates in Computer Graphics and Computational Mechanics*. Taylor & Francis, CRC Press, Boca Raton, 2017. ISBN 978-1-4987-6359-2.

[23] K. Hormann and M. Tarini. A quadrilateral rendering primitive. In T. Akenine-Möller and M. McCool, editors, *Graphics Hardware 2004*, Eurographics Symposium Proceedings, pages 7–14, Grenoble, France, Aug. 2004.

[24] A. Jacobson, I. Baran, J. Popović, and O. Sorkine. Bounded biharmonic weights for real-time deformation. *ACM Transactions on Graphics*, 30(4):Article 78, 8 pages, July 2011. Proceedings of SIGGRAPH.

[25] P. Joshi, M. Meyer, T. DeRose, B. Green, and T. Sanocki. Harmonic coordinates for character articulation. *ACM Transactions on Graphics*, 26(3):Article 71, 9 pages, July 2007. Proceedings of SIGGRAPH.

[26] T. Ju, P. Liepa, and J. Warren. A general geometric construction of coordinates in a convex simplicial polytope. *Computer Aided Geometric Design*, 24(3):161–178, Apr. 2007.

[27] T. Ju, S. Schaefer, and J. Warren. Mean value coordinates for closed triangular meshes. *ACM Transactions on Graphics*, 24(3):561–566, July 2005. Proceedings of SIGGRAPH.

[28] T. Ju, S. Schaefer, J. Warren, and M. Desbrun. A geometric construction of coordinates for convex polyhedra using polar duals. In M. Desbrun and H. Pottmann, editors, *Proceedings of the Third Eurographics Symposium on Geometry Processing*, SGP '05, pages 181–186, Vienna, July 2005.

[29] J. A. Kalman. Continuity and convexity of projections and barycentric coordinates in convex polyhedra. *Pacific Journal of Mathematics*, 11(3):1017–1022, Fall 1961.

[30] T. Langer and H.-P. Seidel. Mean value Bézier surfaces. In R. Martin, M. Sabin, and J. Winkler, editors, *Mathematics of Surfaces XII*, volume 4647 of *Lecture Notes in Computer Science*, pages 263–274. Springer, Berlin, Heidelberg, 2007. Proceedings of the 12th IMA International Conference.

[31] T. Langer and H.-P. Seidel. Higher order barycentric coordinates. *Computer Graphics Forum*, 27(2):459–466, Apr. 2008. Proceedings of Eurographics.

[32] X.-Y. Li and S.-M. Hu. Poisson coordinates. *IEEE Transactions on Visualization and Computer Graphics*, 19(2):344–352, Feb. 2013.

[33] Y. Lipman, J. Kopf, D. Cohen-Or, and D. Levin. GPU-assisted positive mean value coordinates for mesh deformations. In A. Belyaev and M. Garland, editors, *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, SGP '07, pages 117–123, Barcelona, Spain, July 2007.

[34] Y. Lipman, D. Levin, and D. Cohen-Or. Green coordinates. *ACM Transactions on Graphics*, 27(3):Article 78, 10 pages, Aug. 2008. Proceedings of SIGGRAPH.

[35] Y. Lipman, R. Rustamov, and T. Funkhouser. Biharmonic distance. *ACM Transactions on Graphics*, 29(3):Article 27, 11 pages, June 2010.

[36] C. T. Loop and T. D. DeRose. A multisided generalization of Bézier surfaces. *ACM Transactions on Graphics*, 8(3):204–234, July 1989.

[37] E. A. Malsch, J. J. Lin, and G. Dasgupta. Smooth two dimensional interpolants: A recipe for all polygons. *Journal of Graphics Tools*, 10(2):27–39, 2005.

[38] J. Manson, K. Li, and S. Schaefer. Positive Gordon–Wixom coordinates. *Computer-Aided Design*, 43(11):1422–1426, Nov. 2011.

[39] M. Meyer, A. Barr, H. Lee, and M. Desbrun. Generalized barycentric coordinates on irregular polygons. *Journal of Graphics Tools*, 7(1):13–22, 2002.

[40] D. Millán, N. Sukumar, and M. Arroyo. Cell-based maximum-entropy approximants. *Computer Methods in Applied Mechanics and Engineering*, 284:712–731, Feb. 2015.

[41] A. F. Möbius. *Der barycentrische Calcul*. Johann Ambrosius Barth, Leipzig, 1827.

[42] U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2(1):15–36, 1993.

[43] N. Sukumar and R. W. Wright. Overview and construction of meshfree basis functions: from moving least squares to entropy approximants. *International Journal for Numerical Methods in Engineering*, 70(2):181–205, Apr. 2007.

[44] J. Tao, B. Deng, and J. Zhang. A fast numerical solver for local barycentric coordinates. *Computer Aided Geometric Design*, 70:46–58, Mar. 2019.

[45] J.-M. Thiery and T. Boubekeur. Green coordinates for triquad cages in 3D. In *SIGGRAPH Asia 2022 Conference Proceedings*, pages 1–8, Article 38, Daegu, Republic of Korea, Dec. 2022.

[46] J.-M. Thiery, P. Memari, and T. Boubekeur. Mean value coordinates for quad cages in 3D. *ACM Transactions on Graphics*, 37(6):Article 229, 14 pages, Dec. 2018. Proceedings of SIGGRAPH Asia.

[47] E. L. Wachspress. *A Rational Finite Element Basis*, volume 114 of *Mathematics in Science and Engineering*. Academic Press, New York, 1975. ISBN 978-0-12-728950-2.

[48] J. Warren. Barycentric coordinates for convex polytopes. *Advances in Computational Mathematics*, 6(1):97–108, Dec. 1996.

[49] J. Warren, S. Schaefer, A. N. Hirani, and M. Desbrun. Barycentric coordinates for convex sets. *Advances in Computational Mathematics*, 27(3):319–338, Oct. 2007.

[50] Z. Yan and S. Schaefer. A family of barycentric coordinates for co-dimension 1 manifolds with simplicial facets. *Computer Graphics Forum*, 38(5):75–83, Aug. 2019. Proceedings of SGP.

[51] J. Zhang, B. Deng, Z. Liu, G. Patanè, S. Bouaziz, K. Hormann, and L. Liu. Local barycentric coordinates. *ACM Transactions on Graphics*, 33(6):Article 188, 12 pages, Nov. 2014. Proceedings of SIGGRAPH Asia. [PDF]

A Implementation

To facilitate the implementation of our work, this appendix provides further details and pseudo-code for the evaluation of the coordinate functions and their gradients.

A.1 Basic maximum likelihood coordinates

The pseudo-code in Algorithm 1 for computing the BMLC $\lambda(v)$ of a point v inside a convex polytope Ω in $\mathbb{R}^d$ closely follows the explanation in Section 2 and explicitly states how to minimize the convex function F in (9) using Newton's method with Armijo line search. Usually, 3 to 7 iterations suffice to reach the convergence

Algorithm 1 Basic maximum likelihood coordinates

Input: vertices $v_1,\dots,v_n$ of a polytope Ω in $\mathbb{R}^d$ and point $v \in \mathrm{Int}(\Omega)$
Output: coordinates $\lambda = (\lambda_1,\dots,\lambda_n)$ of v w.r.t. Ω and optimal $\phi \in \mathbb{R}^d$

```
 1: function BMLC($v_1,\dots,v_n,v$)
 2:     initialize $\phi = 0$, $\epsilon = 10^{-10}$, $\rho = 0.55$, $\tau = 0.4$, and $K = 20$
 3:     for $i = 1$ to $n$ do                                   ▷ translate all $v_i$ by $-v$
 4:         $v_i := v_i - v$
 5:     $F := -n \log n$                                       ▷ compute value of $F$ in (9) at initial $\phi = 0$
 6:     while true do
 7:         $F_{\text{new}} := 0, \quad g := 0 \in \mathbb{R}^d, \quad H := 0 \in \mathbb{R}^{d\times d}$
 8:         for $i = 1$ to $n$ do                               ▷ compute gradient $g$ of $F$ at $\phi$
 9:             $c_i := n + \phi^{\mathsf{T}} v_i, \quad g := g - v_i / c_i$
10:         if $\|g\|_2 < \epsilon$ then
11:             goto line 22                                    ▷ exit while loop
12:         for $i = 1$ to $n$ do                               ▷ compute Hessian $H$ of $F$ at $\phi$
13:             $H := H + v_i v_i^{\mathsf{T}} / c_i^2$
14:         $q := -H^{-1} g$                                    ▷ search direction
15:         for $k = 0$ to $K$ do                               ▷ Armijo line search method [4]
16:             $\phi_{\text{new}} := \phi + \rho^k q$
17:             for $i = 1$ to $n$ do                           ▷ compute value of $F$ at $\phi_{\text{new}}$
18:                 $F_{\text{new}} := F_{\text{new}} - \log(n + \phi_{\text{new}}^{\mathsf{T}} v_i)$
19:             if $F_{\text{new}} < F + \tau \rho^k g^{\mathsf{T}} q$ then
20:                 goto line 21                                ▷ exit for loop
21:         $\phi := \phi_{\text{new}}, \quad F := F_{\text{new}}$
22:     $\lambda := (1/c_1,\dots,1/c_n)$                        ▷ see (7)
23:     return $[\lambda, \phi]$
```

Algorithm 2 Gradient of basic maximum likelihood coordinates

Input: vertices $v_1,\dots,v_n$ of a polytope Ω in $\mathbb{R}^d$ and point $v \in \mathrm{Int}(\Omega)$
Output: gradients $\nabla\lambda_1,\dots,\nabla\lambda_n$ of BMLC at v

```
1: function GradientBMLC($v_1,\dots,v_n,v$)
2:     initialize $G_\phi := 0 \in \mathbb{R}^{d\times d}$, $G_v := -I_d = -\mathrm{diag}(1,\dots,1) \in \mathbb{R}^{d\times d}$
3:     $[\lambda,\phi] := \mathrm{BMLC}(v_1,\dots,v_n,v)$                ▷ BMLC $\lambda$ of $v$ w.r.t. $\Omega$
4:     for $i=1$ to $n$ do
5:         $v_i := \lambda_i(v_i - v)$,   $G_\phi := G_\phi + v_i v_i^\mathsf{T}$,   $G_v := G_v + \lambda_i \phi\, v_i^\mathsf{T}$
6:     $G := G_v G_\phi^{-1}$                ▷ recall from Section 2.1 that $G_\phi$ is invertible
7:     for $i=1$ to $n$ do
8:         $\nabla\lambda_i := \lambda_i(\lambda_i\phi - G v_i)$                ▷ see (12)
9:     return $(\nabla\lambda_1,\dots,\nabla\lambda_n)$
```

threshold $\epsilon = 10^{-10}$, but up to 15 iterations may be required, if v is very close (at a distance of 10^{-4} times the diameter of Ω) to the boundary.

Similarly, the pseudo-code in Algorithm 2 for computing the gradients of BMLC sticks to the formulas derived in Section 2.1, which generalize straightforwardly to any dimension d. Note that since Algorithm 1 must be executed in line 3 as part of the gradient computation, one could easily modify the code to return both λ and $\nabla\lambda$.

A.2 Maximum likelihood coordinates

The pseudo-code in Algorithm 3 for computing the MLC $\lambda(v)$ of a point v inside an arbitrary simple polygon Ω in $\mathbb{R}^2$ closely follows the derivation in Section 3, but with a few modifications that help to simplify and speed up the code.

We first compute the projected vertices $\mathring{v}_i$ in lines 2–3 as in Section 3, but then deviate from (13) for the computation of the s_j in lines 5–6. More precisely, we omit the multiplications with σ_j, since these factors cancel out when we normalize the s_j. In other words, we use the fact that $A_1P_1 = \bar{A}_1\bar{P}_1$, where

$$\bar{A}_1 = \begin{pmatrix} 1 & 0 & \cdots & 1 \\ 1 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & 1 \end{pmatrix}, \qquad \bar{P}_1 = \mathrm{diag}\left(\frac{1}{\bar{s}_1},\dots,\frac{1}{\bar{s}_n}\right),$$

with $\bar{s}_j = \|\mathring{v}_j + \mathring{v}_{j+1}\|$. We similarly omit the factor $1/2$ when computing t_i and the projected vertices $\hat{v}_i$ in lines 8–9. Note that the statements in lines 4, 7, and 10 (and likewise in lines 12 and 15) serve to avoid modulo operations to keep the indices in the range $[1,\dots,n]$ in the subsequent loops.

After computing the BMLC $\hat{\lambda}$ of the origin w.r.t. $\hat{\Omega}$ in line 11, we split the multiplication of $\hat{\lambda}$ with M into three steps. We start by determining the non-zero entries $C_{i,i} = \alpha_i$ and $C_{i,i+1} = \beta_i$ of the matrix $C = \bar{P}_1 A_2 P_2$ in lines 13–14 and then compute first the weights $\mathring{w} = \bar{A}_1\bar{P}_1 A_2 P_2 \hat{\lambda}$ and finally $w = P_0\mathring{w}$ in lines 17–19. In the end, we get the MLC λ by normalizing w in line 20.

To understand the pseudo-code in Algorithm 4 for computing the gradient of MLC, let us parse it from bottom to top. First recall that

$$\lambda_i = \frac{w_i}{\sum_{j=1}^n w_j}, \qquad w_i = \frac{\mathring{w}_i}{r_i}, \qquad r_i = \|v_i - v\|,$$

so that, by the quotient rule,

$$\nabla\lambda_i = \frac{\nabla w_i \sum_{j=1}^n w_j - w_i \sum_{j=1}^n \nabla w_j}{\left(\sum_{j=1}^n w_j\right)^2} \tag{17}$$

and

$$\nabla w_i = \frac{\nabla\mathring{w}_i r_i - \mathring{w}_i \nabla r_i}{r_i^2} = \frac{\nabla\mathring{w}_i r_i + \mathring{w}_i \mathring{v}_i}{r_i^2}, \tag{18}$$

as implemented in lines 29–34.

Algorithm 3 Maximum likelihood coordinates

Input: vertices $v_1,\dots,v_n$ of a polygon Ω in $\mathbb{R}^2$ and point $v \in \mathrm{Int}(\Omega)$
Output: coordinates $\lambda = (\lambda_1,\dots,\lambda_n)$ of v w.r.t. Ω

```
 1: function MLC($v_1,\dots,v_n,v$)
 2:     for $i=1$ to $n$ do                                   ▷ translate by $-v$ and project with $P_0$
 3:         $\tilde v_i := v_i - v,\quad r_i := \|\tilde v_i\|,\quad \mathring{v}_i := \tilde v_i / r_i$
 4:     $\mathring{v}_{n+1} := \mathring{v}_1$
 5:     for $j=1$ to $n$ do                                   ▷ first averaging step $A_1$ and projection $P_1$
 6:         $s_j := \mathring{v}_j + \mathring{v}_{j+1},\quad \bar s_j := \|s_j\|,\quad \mathring{s}_j := s_j/\bar s_j$      ▷ normalized $\mathring{s}$
 7:     $\mathring{s}_0 := \mathring{s}_n$
 8:     for $i=1$ to $n$ do                                   ▷ second averaging step $A_2$ and projection $P_2$
 9:         $t_i := \mathring{s}_{i-1} + \mathring{s}_i,\quad \bar t_i := \|t_i\|,\quad \hat v_i := t_i/\bar t_i$
10:     $\bar t_{n+1} := \bar t_1$
11:     $[\hat\lambda,\phi] :=$ BMLC($\hat v_1,\dots,\hat v_n,0$)                ▷ BMLC $\hat\lambda$ of the origin w.r.t. $\hat\Omega$
12:     $\hat\lambda_0 := \hat\lambda_n,\quad \hat\lambda_{n+1} := \hat\lambda_1$
13:     for $i=1$ to $n$ do                                   ▷ entries of the matrix $\bar P_1 A_2 P_2$
14:         $\alpha_i := 1/(\bar s_i \bar t_i),\quad \beta_i := 1/(\bar s_i \bar t_{i+1})$
15:     $\alpha_0 := \alpha_n,\quad \beta_0 := \beta_n$
16:     $W := 0$
17:     for $i=1$ to $n$ do
18:         $\mathring{w}_i := \alpha_{i-1}\hat\lambda_{i-1} + (\alpha_i+\beta_{i-1})\hat\lambda_i + \beta_i\hat\lambda_{i+1}$      ▷ $\mathring{w} = \bar A_1 \bar P_1 A_2 P_2 \hat\lambda$
19:         $w_i := \mathring{w}_i / r_i,\quad W := W + w_i$                  ▷ $w = P_0 \mathring{w}$
20:     $\lambda := (w_1/W,\dots,w_n/W)$                          ▷ normalize $w$ to get $\lambda$
21:     return $\lambda$
```

The gradient of $\mathring{w}_i$ is then derived by applying the product rule to the formula in line 18 of Algorithm 3,

$$\begin{aligned}\nabla \mathring{w}_i = {} & \alpha_{i-1}\nabla\hat\lambda_{i-1} + \hat\lambda_{i-1}\nabla\alpha_{i-1} \\ & + (\alpha_i+\beta_{i-1})\nabla\hat\lambda_i + \hat\lambda_i(\nabla\alpha_i + \nabla\beta_{i-1}) \\ & + \beta_i\nabla\hat\lambda_{i+1} + \hat\lambda_{i+1}\nabla\beta_i.\end{aligned} \tag{19}$$

To find the gradients of $\alpha_i = 1/(\bar s_i \bar t_i)$ and $\beta_i = 1/(\bar s_i \bar t_{i+1})$, we recall the definition of $\bar s_i$ and $\bar t_i$ in lines 6 and 9 of Algorithm 3 and realize that

$$\alpha_i = \frac{1}{\|\mathring{v}_i + \mathring{v}_{i+1}\| \cdot \|\mathring{s}_{i-1} + \mathring{s}_i\|}, \qquad \beta_i = \frac{1}{\|\mathring{v}_i + \mathring{v}_{i+1}\| \cdot \|\mathring{s}_{i+1} + \mathring{s}_i\|}$$

are functions that depend on several variables $\mathring{v}_j$ and $\mathring{s}_j$. Hence, we apply the chain rule to express their derivatives w.r.t. v as

$$\begin{aligned}\nabla^{\mathsf{T}}\alpha_i &= \frac{\mathrm{d}\alpha_i}{\mathrm{d}v} = \frac{\partial\alpha_i}{\partial\mathring{v}_i}\frac{\mathrm{d}\mathring{v}_i}{\mathrm{d}v} + \frac{\partial\alpha_i}{\partial\mathring{v}_{i+1}}\frac{\mathrm{d}\mathring{v}_{i+1}}{\mathrm{d}v} + \frac{\partial\alpha_i}{\partial\mathring{s}_{i-1}}\frac{\mathrm{d}\mathring{s}_{i-1}}{\mathrm{d}v} + \frac{\partial\alpha_i}{\partial\mathring{s}_i}\frac{\mathrm{d}\mathring{s}_i}{\mathrm{d}v}, \\ \nabla^{\mathsf{T}}\beta_i &= \frac{\mathrm{d}\beta_i}{\mathrm{d}v} = \frac{\partial\beta_i}{\partial\mathring{v}_i}\frac{\mathrm{d}\mathring{v}_i}{\mathrm{d}v} + \frac{\partial\beta_i}{\partial\mathring{v}_{i+1}}\frac{\mathrm{d}\mathring{v}_{i+1}}{\mathrm{d}v} + \frac{\partial\beta_i}{\partial\mathring{s}_i}\frac{\mathrm{d}\mathring{s}_i}{\mathrm{d}v} + \frac{\partial\beta_i}{\partial\mathring{s}_{i+1}}\frac{\mathrm{d}\mathring{s}_{i+1}}{\mathrm{d}v}.\end{aligned}$$

The partial derivatives of α_i and β_i w.r.t. the relevant $\mathring{v}_j$ and $\mathring{s}_j$ are found using the quotient rule,

$$\begin{aligned}\frac{\partial\alpha_i}{\partial\mathring{v}_i} &= \frac{\partial\alpha_i}{\partial\mathring{v}_{i+1}} = \frac{-\alpha_i\mathring{s}_i^{\mathsf{T}}}{\bar s_i}, & \frac{\partial\alpha_i}{\partial\mathring{s}_i} &= \frac{\partial\alpha_i}{\partial\mathring{s}_{i-1}} = \frac{-\alpha_i\hat v_i^{\mathsf{T}}}{\bar t_i}, \\ \frac{\partial\beta_i}{\partial\mathring{v}_i} &= \frac{\partial\beta_i}{\partial\mathring{v}_{i+1}} = \frac{-\beta_i\mathring{s}_i^{\mathsf{T}}}{\bar s_i}, & \frac{\partial\beta_i}{\partial\mathring{s}_i} &= \frac{\partial\beta_i}{\partial\mathring{s}_{i+1}} = \frac{-\beta_i\hat v_{i+1}^{\mathsf{T}}}{\bar t_{i+1}},\end{aligned}$$

so that

$$\nabla\alpha_i = -\alpha_i\left(\frac{X_i\mathring{s}_i}{\bar s_i} + \frac{Y_i\hat v_i}{\bar t_i}\right), \qquad \nabla\beta_i = -\beta_i\left(\frac{X_i\mathring{s}_i}{\bar s_i} + \frac{Y_{i+1}\hat v_{i+1}}{\bar t_{i+1}}\right),$$

where

$$X_i = \nabla\mathring{v}_i + \nabla\mathring{v}_{i+1}, \qquad Y_i = \nabla\mathring{s}_{i-1} + \nabla\mathring{s}_i.$$

Algorithm 4 Gradient of maximum likelihood coordinates

Input: vertices $v_1, \dots, v_n$ of a polygon Ω in $\mathbb{R}^2$ and point $v \in \operatorname{Int}(\Omega)$
Output: gradients $\nabla\lambda_1, \dots, \nabla\lambda_n$ of MLC at v

```
 1: function GRADIENTMLC($v_1, \dots, v_n, v$)
 2:     use Algorithm 3 to get $\phi \in \mathbb{R}^2$, $r, \bar{s}, \bar{t}, \hat{\lambda}, \alpha, \beta, \mathring{w}, w \in \mathbb{R}^n$, $\mathring{v}, \mathring{s}, \hat{v} \in (\mathbb{R}^2)^n$
 3:     for $i = 1$ to $n$ do                                        ▷ compute $\nabla\mathring{v}$
 4:         $\nabla\mathring{v}_i := (\mathring{v}_i \mathring{v}_i^\mathsf{T} - I_2)/r_i$              ▷ see (20)
 5:     $\nabla\mathring{v}_{n+1} := \nabla\mathring{v}_1$
 6:     for $i = 1$ to $n$ do                    ▷ compute $\nabla\mathring{s}$ and auxiliary variables
 7:         $X_i := \nabla\mathring{v}_i + \nabla\mathring{v}_{i+1}$,   $\mathring{S} := (I_2 - \mathring{s}_i \mathring{s}_i^\mathsf{T})/\bar{s}_i$
 8:         $\nabla\mathring{s}_i := X_i \mathring{S}$                                       ▷ see (21)
 9:     $\nabla\mathring{s}_0 := \nabla\mathring{s}_n$
10:     for $i = 1$ to $n$ do                    ▷ compute $\nabla\hat{v}$ and auxiliary variables
11:         $Y_i := \nabla\mathring{s}_{i-1} + \nabla\mathring{s}_i$,   $\hat{V} := (I_2 - \hat{v}_i \hat{v}_i^\mathsf{T})/\bar{t}_i$
12:         $\nabla\hat{v}_i := Y_i \hat{V}$                                        ▷ see (22)
13:     $G_\phi := 0 \in \mathbb{R}^{2\times 2}$,   $\nabla\phi := 0 \in \mathbb{R}^{2\times 2}$
14:     for $i = 1$ to $n$ do
15:         $G_\phi := G_\phi + \hat{\lambda}_i^2 \hat{v}_i \hat{v}_i^\mathsf{T}$,   $G_{\hat{v}} := \hat{\lambda}_i I_2 - \hat{\lambda}_i^2 \phi\, \hat{v}_i^\mathsf{T}$
16:         $\nabla\phi := \nabla\phi + \nabla\hat{v}_i\, G_{\hat{v}}$
17:     $\nabla\phi := \nabla\phi\, G_\phi^{-1}$                                     ▷ see (24)
18:     for $i = 1$ to $n$ do
19:         $\nabla\hat{\lambda}_i := -\hat{\lambda}_i^2(\nabla\phi\, \hat{v}_i + \nabla\hat{v}_i\, \phi)$                  ▷ see (23)
20:     for $i = 1$ to $n$ do                  ▷ auxiliary variables for $\nabla\alpha$ and $\nabla\beta$
21:         $x_i = X_i \mathring{s}_i / \bar{s}_i$,   $y_i = Y_i \hat{v}_i / \bar{t}_i$
22:     $x_0 := x_n$,   $\beta_0 := \beta_n$
23:     for $i = 1$ to $n$ do                          ▷ auxiliary variables for $\nabla\mathring{w}$
24:         $a_i = \alpha_i \nabla\hat{\lambda}_i - \hat{\lambda}_i \alpha_i (x_i + y_i)$,           ▷ $-\alpha_i(x_i + y_i) = \nabla\alpha_i$
25:         $b_i = \beta_{i-1} \nabla\hat{\lambda}_i - \hat{\lambda}_i \beta_{i-1}(x_{i-1} + y_i)$       ▷ $-\beta_i(x_i + y_{i+1}) = \nabla\beta_i$
26:     $a_0 := a_n$,   $b_{n+1} := b_1$
27:     for $i = 1$ to $n$ do                                        ▷ compute $\nabla\mathring{w}$
28:         $\nabla\mathring{w}_i := a_{i-1} + a_i + b_i + b_{i+1}$                    ▷ see (19)
29:     $W := 0$,   $\nabla W := 0 \in \mathbb{R}^2$
30:     for $i = 1$ to $n$ do                              ▷ compute $\nabla w$ and $\nabla W$
31:         $\nabla w_i := (\nabla\mathring{w}_i r_i + \mathring{w}_i \mathring{v}_i)/r_i^2$                ▷ see (18)
32:         $W := W + w_i$,   $\nabla W := \nabla W + \nabla w_i$
33:     for $i = 1$ to $n$ do                                        ▷ compute $\nabla\lambda$
34:         $\nabla\lambda_i := (\nabla w_i W - w_i \nabla W)/W^2$                   ▷ see (17)
35:     return $(\nabla\lambda_1, \dots, \nabla\lambda_n)$
```

Before working out the formulas for $\nabla\mathring{v}_i$, $\nabla\mathring{s}_i$, and $\nabla\hat{\lambda}_i$, observe that the implementation of (19) is found in lines 20–28 of Algorithm 4.

Let us now derive the gradients of

$$\mathring{v}_i = \frac{v_i - v}{\|v_i - v\|}, \qquad \mathring{s}_i = \frac{\mathring{v}_i + \mathring{v}_{i+1}}{\|\mathring{v}_i + \mathring{v}_{i+1}\|}, \qquad \hat{v}_i = \frac{\mathring{s}_{i-1} + \mathring{s}_i}{\|\mathring{s}_{i-1} + \mathring{s}_i\|}.$$

By the quotient rule, we immediately have

$$\nabla\mathring{v}_i = \frac{\mathring{v}_i \mathring{v}_i^\mathsf{T} - I_2}{r_i}, \tag{20}$$

as well as

$$\frac{\partial\,\mathring{s}_i}{\partial\,\mathring{v}_i} = \frac{\partial\,\mathring{s}_i}{\partial\,\mathring{v}_{i+1}} = \frac{I_2 - \mathring{s}_i \mathring{s}_i^\mathsf{T}}{\bar{s}_i}, \qquad \frac{\partial\,\hat{v}_i}{\partial\,\mathring{s}_{i-1}} = \frac{\partial\,\hat{v}_i}{\partial\,\mathring{s}_i} = \frac{I_2 - \hat{v}_i \hat{v}_i^\mathsf{T}}{\bar{t}_i}.$$

Using the chain rule, we then get

$$\nabla^\mathsf{T} \mathring{s}_i = \frac{\mathrm{d}\mathring{s}_i}{\mathrm{d}v} = \frac{\partial\,\mathring{s}_i}{\partial\,\mathring{v}_i}\frac{\mathrm{d}\mathring{v}_i}{\mathrm{d}v} + \frac{\partial\,\mathring{s}_i}{\partial\,\mathring{v}_{i+1}}\frac{\mathrm{d}\mathring{v}_{i+1}}{\mathrm{d}v} \tag{21}$$

and

$$\nabla^{\mathsf{T}}\hat{v}_i = \frac{\mathrm{d}\hat{v}_i}{\mathrm{d}v} = \frac{\partial\hat{v}_i}{\partial\mathring{s}_{i-1}}\frac{\mathrm{d}\mathring{s}_{i-1}}{\mathrm{d}v} + \frac{\partial\hat{v}_i}{\partial\mathring{s}_i}\frac{\mathrm{d}\mathring{s}_i}{\mathrm{d}v}. \tag{22}$$

Lines 3-12 implement the formulas for these three gradients.

It remains to determine $\nabla\hat{\lambda}_i$. To this end, recall that $\hat{\lambda}$ are the BMLC of the origin w.r.t. the vertices $\hat{v}_1,\dots,\hat{v}_n$, hence, by (7),

$$\hat{\lambda}_i(\phi,\hat{v}_i) = \frac{1}{n+\phi^{\mathsf{T}}\hat{v}_i},$$

where $\phi = \phi(\hat{v}_1,\dots,\hat{v}_n)$ denotes the minimum of the function $\hat{F}(\phi) = -\sum_{i=1}^n \log(n+\phi^{\mathsf{T}}\hat{v}_i)$. Since the partial derivatives of $\hat{\lambda}_i$ are

$$\frac{\partial\hat{\lambda}_i}{\partial\phi} = -\hat{\lambda}_i^2\hat{v}_i^{\mathsf{T}}, \qquad \frac{\partial\hat{\lambda}_i}{\partial\hat{v}_i} = -\hat{\lambda}_i^2\phi^{\mathsf{T}},$$

we find, by applying the chain rule, that

$$\nabla\hat{\lambda}_i = -\hat{\lambda}_i^2(\nabla\phi\,\hat{v}_i + \nabla\hat{v}_i\,\phi). \tag{23}$$

Using the chain rule again, we express the derivative of ϕ w.r.t. v as

$$\frac{\mathrm{d}\phi}{\mathrm{d}v} = \sum_{i=1}^n \frac{\partial\phi}{\partial\hat{v}_i}\frac{\mathrm{d}\hat{v}_i}{\mathrm{d}v},$$

and to get the partial derivatives of ϕ w.r.t. $\hat{v}_i$, we exploit the fact (cf. Section 2.1) that the function

$$G(\phi,\hat{v}_1,\dots,\hat{v}_n) = \sum_{i=1}^n \frac{\hat{v}_i}{n+\phi^{\mathsf{T}}\hat{v}_i}$$

as well as its variations w.r.t. $\hat{v}_i$ vanish, that is,

$$\frac{\mathrm{d}G}{\mathrm{d}\hat{v}_i} = \frac{\partial G}{\partial\phi}\frac{\mathrm{d}\phi}{\mathrm{d}\hat{v}_i} + \frac{\partial G}{\partial\hat{v}_i} = 0.$$

Solving these equations for $\partial\phi/\partial\hat{v}_i = \mathrm{d}\phi/\mathrm{d}\hat{v}_i$, it follows that

$$\nabla\phi = \sum_{i=1}^n \nabla\hat{v}_i G_{\hat{v}_i} G_\phi^{-1}, \tag{24}$$

where

$$G_\phi = \sum_{i=1}^n \hat{\lambda}_i^2\hat{v}_i\hat{v}_i^{\mathsf{T}}, \qquad G_{\hat{v}_i} = \hat{\lambda}_i I_2 - \hat{\lambda}_i^2\phi\,\hat{v}_i^{\mathsf{T}}.$$

The implementation of (23) and (24) can be found in lines 13–19.

B Comparison of properties

Table 1 provides a summary of the properties of all generalized barycentric coordinates for simple planar polygons that we are aware of. The first group consists of coordinates that are well-defined only for convex polygons, and the last group are computational coordinates that do not have a closed form.

For the family of 3-point coordinates [15], note that not all coordinates in this family are non-negative.

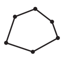
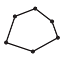
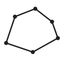
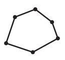
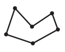
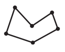
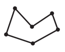
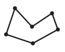
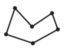
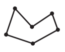
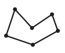
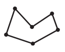
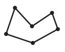
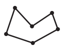
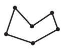
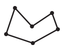
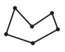
coordinates	domain	non-negative	exact pointwise evaluation		smoothness	comments
			coordinates	gradient		
Wachspress [47, 39]		✓	✓	✓	C^∞	
discrete harmonic [42, 9]		✗	✓	✓	C^∞	also known as "cotangent weights"
3-point family [15]		(✓)	✓	✓	C^∞	includes Wachspress, discrete harmonic, and mean value
power [7]		✓	✓	✓	C^∞	extension to non-convex polygons C^0 only
mean value [12, 20]		✗	✓	✓	C^∞	positive inside convex polygons
positive mean value [33]		✓	✓	✓	C^0	
metric [37]		✗	✓	✓	C^∞	polygon can have isolated interior points and holes
Poisson [32]		✗	✓	✓	C^∞	
Gordon–Wixom [19, 6]		✗	✓	✓	C^0	positive and C^∞ inside convex polygons
positive Gordon–Wixom [38]		✓	✓	✓	C^0	
blended [3]		✓	✓	✓	C^k	not continuous with respect to the polygon's vertices
5-point family [50]		✗	✓	✓	C^∞	polygon can be degenerate
iterative [8]		✓	✓	✗	C^∞	may need $O(n^2)$ iterations for non-negativity
harmonic [25]		✓	✗	✗	C^∞	usually approximated with FEM or BEM
local [51, 44]		✓	✗	✗	C^0	
maximum entropy [21]		✓	✓	✓	C^∞	proposed prior functions lack global shape awareness
maximum likelihood		✓	✓	✓	C^∞	SMLC extension has a certain global shape awareness

Table 1: Properties of generalized barycentric coordinates for simple planar polygons.