Blended barycentric coordinates

Dmitry Anisimov · Daniele Panozzo · Kai Hormann

Abstract

Generalized barycentric coordinates are widely used to represent a point inside a polygon as an affine combination of the polygon's vertices, and it is desirable to have coordinates that are non-negative, smooth, and locally supported. Unfortunately, the existing coordinate functions that satisfy all these properties do not have a simple analytic expression, making them expensive to evaluate and difficult to differentiate. In this paper, we present a new closed-form construction of generalized barycentric coordinates, which are non-negative, smooth, and locally supported. Our construction is based on the idea of blending mean value coordinates over the triangles of the constrained Delaunay triangulation of the input polygon, which needs to be computed in a preprocessing step. We experimentally show that our construction compares favourably with other generalized barycentric coordinates, both in terms of quality and computational cost.

Citation Info

Journal
Computer Aided Geometric Design

Volume
52–53, March–April 2017

Pages
205–216

Note
Proceedings of GMP

1 Introduction

Given a planar n-sided simple polygon $P \subset \mathbb{R}^2$ with $n \geq 3$ vertices $v_1, \dots, v_n \in \mathbb{R}^2$, the n functions $\boldsymbol{b} = [b_1, \dots, b_n] \colon P \to \mathbb{R}^n$ are called *generalized barycentric coordinates* if they satisfy the *partition of unity* property

$$\sum_{i=1}^{n} b_i(v) = 1 \qquad \forall\, v \in P \tag{1}$$

and the *linear reproduction* property

$$\sum_{i=1}^{n} b_i(v) v_i = v \qquad \forall\, v \in P. \tag{2}$$

In addition, it is often desirable for the functions b_i to be *non-negative*,

$$b_i(v) \geq 0, \quad i = 1, \dots, n \qquad \forall\, v \in P, \tag{3}$$

and to have the *Lagrange property*

$$b_i(v_j) = \delta_{i,j} = \begin{cases} 0, & \text{if } i \neq j, \\ 1, & \text{if } i = j, \end{cases} \qquad i, j = 1, \dots, n, \tag{4}$$

where $\delta_{i,j}$ is the Kronecker delta. We also say that the coordinate functions are *smooth*, if

$$b_i \in C^k, \quad k > 0, \tag{5}$$

and *local*, if the functions b_i have small support

$$\operatorname{supp}(b_i) = \overline{\{v \in P : b_i(v) \neq 0\}}. \tag{6}$$

For $n = 3$, when P is a triangle, it is known [21] that the barycentric coordinates of v are uniquely determined by (1) and (2). These *classical barycentric coordinates* are linear functions that satisfy all properties above. However, for polygons with $n > 3$ vertices, the choice of b_i is no longer unique.

In recent years, many different constructions of barycentric coordinates for polygons with $n > 3$ vertices have been proposed. Some of these coordinates are successfully used for colour interpolation [20], image warping [12], shape deformation [15], and many other applications. Nevertheless, the question of finding a simple *closed-form* construction of the functions b_i, which satisfy all properties above is still open.

1.1 Related work

Wachspress [26] was one of the first who proposed a construction of barycentric coordinate functions for polygons with $n > 3$ vertices. These *Wachspress coordinates* are rational functions and have a simple closed form [20], but they are well-defined only for convex polygons. The same holds for *discrete harmonic coordinates* [22, 7], which arise from the standard piecewise linear finite element approximation of the Laplace equation.

The construction of discrete harmonic coordinates suggests that different properties of harmonic functions can be exploited to derive generalized barycentric coordinates, and Floater [8] uses the circumferential mean value property of harmonic functions to derive *mean value coordinates*, which have a simple closed form and are well-defined for any concave polygon [12]. They are positive inside the kernel of star-shaped polygons, satisfy the Lagrange property, and are smooth everywhere in the plane, except at the vertices v_i, where they are only C^0 continuous. Other barycentric coordinate functions with a closed-form definition, which are well-defined for concave polygons and possibly negative inside these polygons, are *metric* [18, 25], *Poisson* [16], and *Gordon–Wixom coordinates* [10, 2]. To the best of our knowledge, the only coordinate functions with a closed form that guarantee positivity inside arbitrary concave polygons are *positive mean value* [17] and *positive Gordon–Wixom coordinates* [19], but they are not smooth. Another recent generalization of barycentric coordinates with a closed form is based on power diagrams [3], but the computation of these coordinates is quite involved.

Neither of the above closed-form constructions results in positive and smooth coordinates for arbitrary concave polygons, but it is possible to obtain generalized barycentric coordinates with both properties numerically by solving certain optimization problems. Examples of such *computational* coordinates include *harmonic* [9, 15], *maximum entropy* [13], and *local barycentric coordinates* [28]. These coordinates possess all important properties above, but the usual approximation of harmonic and local barycentric coordinates involves a triangulation of the domain, and the obtained coordinates are only piecewise linear functions. Subdivision can be used to make these coordinates C^1 continuous [1], but this approach is computationally involved. Instead, maximum entropy coordinates are smooth and can be evaluated efficiently with Newton's method, but they are globally supported.

All these constructions have in common that they depend on the number of the polygon's vertices, and the time complexity for a single evaluation of the coordinate functions at some $v \in P$ is at least $O(n)$.

1.2 Contributions

We propose a novel construction of generalized barycentric coordinates for arbitrary simple polygons, which are non-negative, smooth, and locally supported. These coordinates have a closed-form definition and can be evaluated in constant time, due to the local support. Note that the constrained Delaunay triangulation of the polygon needs to be computed in a preprocessing step, and depending on the application, an additional cost on the order of $O(\log n)$ for each evaluation may apply, but even in this case, the computation time compares favourably to that of other coordinates.

The proposed construction is based on the idea of blending mean value coordinates over the triangles of the constrained Delaunay triangulation of the input polygon with appropriate blending functions. We describe this idea and the blending functions in detail in Section 2 and provide pseudocode for the efficient implementation of our coordinates in A. We further compare them with other coordinates in Section 3 and discuss the proposed approach and its limitations in Section 4.

2 Blended barycentric coordinates

It is known [14] that mean value coordinates are always positive inside any quadrilateral. We first show how this property can be exploited to obtain non-negative barycentric coordinates for pentagons (Section 2.1), before extending this construction to hexagons (Section 2.2) and to arbitrary polygons (Section 2.3).

2.1 Coordinates for pentagons

Let $P = [v_1, \dots, v_5] \subset \mathbb{R}^2$ be a pentagon in the plane. Without loss of generality, we assume that it can be split into three triangles $\triangle_1 = [v_2, v_3, v_4]$, $\triangle_2 = [v_1, v_2, v_4]$, and $\triangle_3 = [v_1, v_4, v_5]$ (see Figure 1, left). Alternatively, this

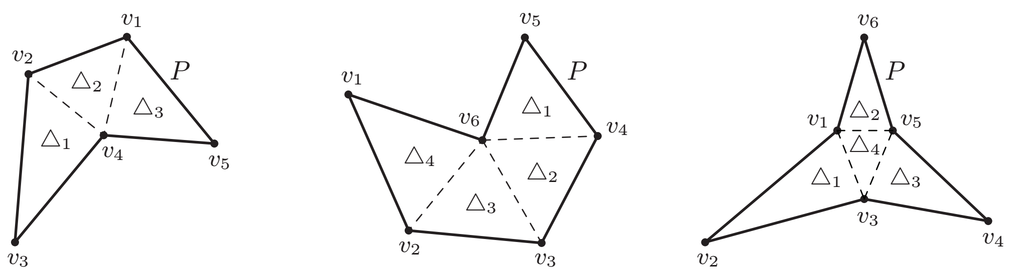

Figure 1: Pentagons can be split into three triangles (left), and hexagons can be split into four triangles (centre and right).

triangulation can be seen as two overlapping quadrilaterals

$$\square_1 = \triangle_1 \cup \triangle_2 = [v_1, v_2, v_3, v_4],$$
$$\square_2 = \triangle_2 \cup \triangle_3 = [v_1, v_2, v_4, v_5]$$

with the common triangle $\triangle_2$. We denote by

$$\boldsymbol{b}_1 = [b_1^1, b_2^1, b_3^1, b_4^1] \colon \square_1 \to \mathbb{R}^4$$

the mean value coordinates with respect to the quadrilateral $\square_1$ and by

$$\boldsymbol{b}_2 = [b_1^2, b_2^2, b_4^2, b_5^2] \colon \square_2 \to \mathbb{R}^4$$

those with respect to the quadrilateral $\square_2$. We also consider two blending functions

$$\mu_1, \mu_2 \colon \triangle_2 \setminus \{v_4\} \to [0,1]$$

such that

$$\mu_1(v) = \begin{cases} 1, & v \in [v_2, v_4), \\ 0, & v \in (v_4, v_1], \end{cases} \qquad \mu_2(v) = \begin{cases} 0, & v \in [v_2, v_4), \\ 1, & v \in (v_4, v_1], \end{cases}$$

and $\mu_1(v) + \mu_2(v) = 1$ for any $v \in \triangle_2 \setminus \{v_4\}$. Since μ_1 and μ_2 are not defined at v_4 and actually diverge as $v \to v_4$, we exclude this vertex from the definition, but, as we will show later, this does not affect the final construction of our barycentric coordinates.

To construct μ_1 and μ_2, we follow a simple procedure. Given the triangle $\triangle_2$, we first determine the classical barycentric coordinates $\lambda_{1,2} \colon \triangle_2 \to [0,1]$ corresponding to v_2 and v_1 as

$$\lambda_1(v) = \frac{\text{Area}[v_1, v, v_4]}{\text{Area}[v_1, v_2, v_4]}, \qquad \lambda_2(v) = \frac{\text{Area}[v, v_2, v_4]}{\text{Area}[v_1, v_2, v_4]} \qquad \forall\, v \in \triangle_2.$$

In order to guarantee smooth coordinates (see details below), we then choose a smooth monotonic function $q \colon [0,1] \to [0,1]$ and define the blending functions as

$$\mu_1 = \frac{\sigma_1}{\sigma_1 + \sigma_2}, \qquad \mu_2 = \frac{\sigma_2}{\sigma_1 + \sigma_2}, \qquad \sigma_1 = q \circ \lambda_1, \qquad \sigma_2 = q \circ \lambda_2. \tag{7}$$

Using the quadrilateral mean value coordinates $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ and the blending functions μ_1 and μ_2, we finally define the *blended coordinate functions* $b_i \colon P \setminus \{v_4\} \to \mathbb{R}$, $i = 1, \ldots, 5$ (see Figure 2) as

$$\begin{gathered} b_i(v) = \begin{cases} b_i^1(v), & v \in \triangle_1, \\ b_i^1(v)\mu_1(v) + b_i^2(v)\mu_2(v), & v \in \triangle_2 \setminus \{v_4\}, \\ b_i^2(v), & v \in \triangle_3, \end{cases} \qquad i = 1,2,4, \\ b_3(v) = \begin{cases} b_3^1(v), & v \in \triangle_1, \\ b_3^1(v)\mu_1(v), & v \in \triangle_2 \setminus \{v_4\}, \\ 0, & v \in \triangle_3, \end{cases} \qquad b_5(v) = \begin{cases} 0, & v \in \triangle_1, \\ b_5^2(v)\mu_2(v), & v \in \triangle_2 \setminus \{v_4\}, \\ b_5^2(v), & v \in \triangle_3. \end{cases} \end{gathered} \tag{8}$$

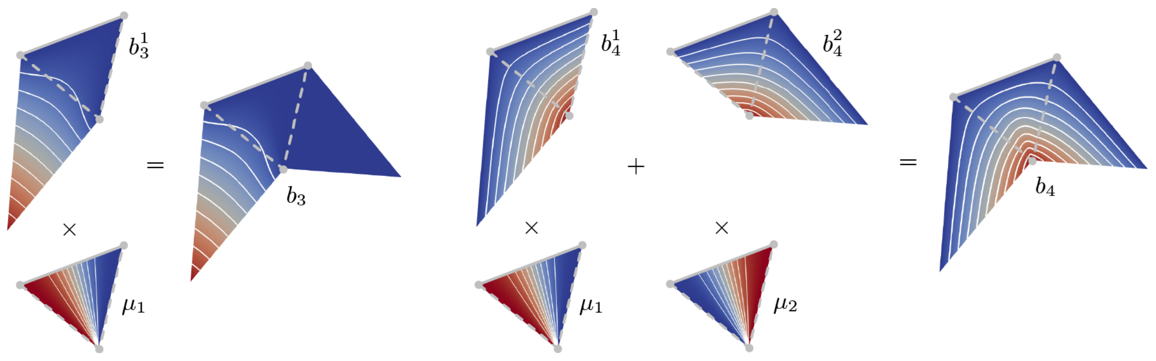

Figure 2: Construction of blended coordinates for the pentagon in Figure 1 (left). The coordinate function b_3 (left) is defined by multiplying the mean value coordinate function b_3^1 for $\square_1$ with the blending function μ_1 over $\triangle_2$ (outlined in grey). To construct b_4 (right), we likewise multiply the mean value coordinates b_4^1 and b_4^2 with μ_1 and μ_2, respectively, and add the results. The colour range shows function values between 0 (blue) and 1 (red), and the white curves are the contour lines at 0.1, 0.2, ..., 0.9.

These functions satisfy the partition of unity property (1), because

$$\begin{aligned}
v \in \triangle_1 \quad &\Rightarrow \quad \sum_{i=1}^{5} b_i(v) = \sum_{i=1,2,3,4} b_i^1(v) = 1,\\
v \in \triangle_2 \setminus \{v_4\} \quad &\Rightarrow \quad \sum_{i=1}^{5} b_i(v) = \sum_{i=1,2,3,4} b_i^1(v)\mu_1(v) + \sum_{i=1,2,4,5} b_i^2(v)\mu_2(v) = \mu_1(v) + \mu_2(v) = 1,\\
v \in \triangle_3 \quad &\Rightarrow \quad \sum_{i=1}^{5} b_i(v) = \sum_{i=1,2,4,5} b_i^2(v) = 1
\end{aligned}$$

and the linear reproduction property (2), because

$$\begin{aligned}
v \in \triangle_1 \quad &\Rightarrow \quad \sum_{i=1}^{5} b_i(v)v_i = \sum_{i=1,2,3,4} b_i^1(v)v_i = v,\\
v \in \triangle_2 \setminus \{v_4\} \quad &\Rightarrow \quad \sum_{i=1}^{5} b_i(v)v_i = \sum_{i=1,2,3,4} b_i^1(v)v_i\mu_1(v) + \sum_{i=1,2,4,5} b_i^2(v)v_i\mu_2(v) = v\mu_1(v) + v\mu_2(v) = v,\\
v \in \triangle_3 \quad &\Rightarrow \quad \sum_{i=1}^{5} b_i(v)v_i = \sum_{i=1,2,4,5} b_i^2(v)v_i = v.
\end{aligned}$$

It further follows directly from the definition that the functions b_i satisfy the non-negativity property (3) and have the Lagrange property (4) at all v_j for $j \neq 4$. To prove the Lagrange property at v_4, we first observe that

$$\min\{b_i^1(v), b_i^2(v)\} \le b_i^1(v)\mu_1(v) + b_i^2(v)\mu_2(v) \le \max\{b_i^1(v), b_i^2(v)\} \qquad \forall\, v \in \triangle_2 \setminus \{v_4\}$$

for $i = 1, 2, 4$, because $\mu_1(v) + \mu_2(v) = 1$, and

$$0 \le b_3^1(v)\mu_1(v) \le b_3^1(v), \qquad 0 \le b_5^2(v)\mu_2(v) \le b_5^2(v) \qquad \forall\, v \in \triangle_2 \setminus \{v_4\},$$

since $\mu_1(v), \mu_2(v) \in [0,1]$. As all lower and upper bounds converge to $\delta_{i,4}$ as v approaches v_4, we conclude that

$$\lim_{v \to v_4} b_i(v) = \delta_{i,4}$$

for all $i = 1, \dots, 5$.

If q has $k > 0$ vanishing derivatives at 0 and at 1, then the construction above guarantees that μ_1 and μ_2 blend with C^k continuity into the constant functions with values 0 or 1 along the edges $[v_2, v_4)$ and $(v_4, v_1]$, which in turn implies the C^k continuity of the coordinate functions b_i. The simplest choices of q for $k = 1$ and $k = 2$ (see Figure 3) are the polynomials

$$q_1(x) = 3x^2 - 2x^3 \qquad \text{and} \qquad q_2(x) = 6x^5 - 15x^4 + 10x^3 \qquad \forall\, x \in [0,1].$$

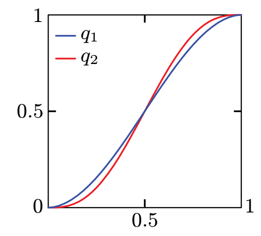

Figure 3: Examples of polynomial functions q_1 and q_2 used for the construction of C^1 and C^2 blended coordinates, respectively.

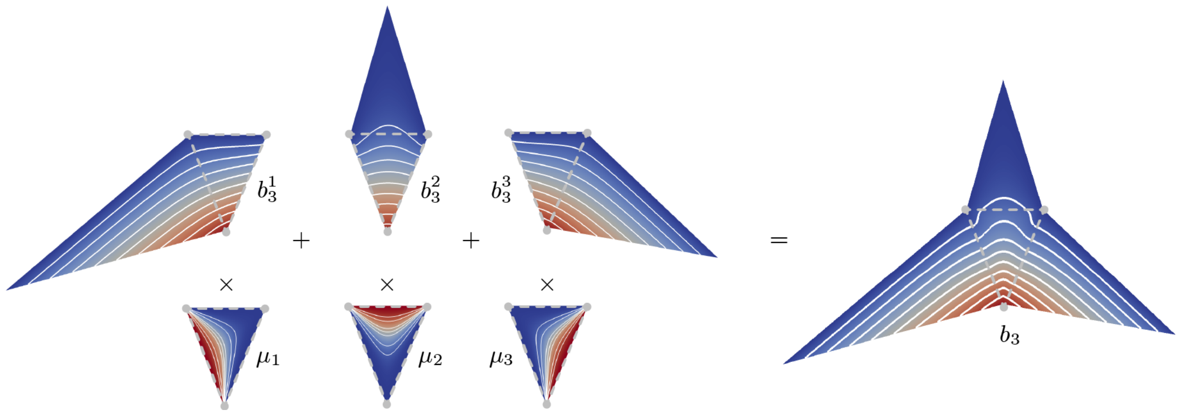

Figure 4: The construction of the blended coordinate function b_3 for the hexagon in Figure 1 (right) involves three blending functions over $\triangle_4$ (outlined in grey) for combining the quadrilateral mean value coordinates b_3^1, b_3^2, and b_3^3.

The function q_1 was used for the examples in Figures 2 and 4, and a comparison between C^1 and C^2 continuous coordinates, constructed with q_1 and q_2, respectively, can be found in Figures 6, 7, and 8.

2.2 Coordinates for hexagons

To construct blended coordinates for a planar hexagon, a similar approach can be used after splitting the hexagon into four triangles. If all triangles of the split have only one or two neighbouring triangles (see Figure 1, centre), then the blended coordinates are constructed as described in Section 2.1. However, it can also happen that one of the triangles has three neighbours (see Figure 1, right). Given such a hexagon $P = [v_1, \dots, v_6] \subset \mathbb{R}^2$ in the plane, let

$$\begin{aligned}\square_1 &= \triangle_1 \cup \triangle_4 = [v_1, v_2, v_3, v_5],\\ \square_2 &= \triangle_2 \cup \triangle_4 = [v_1, v_3, v_5, v_6],\\ \square_3 &= \triangle_3 \cup \triangle_4 = [v_1, v_3, v_4, v_5]\end{aligned}$$

be the quadrilaterals that overlap over the triangle $\triangle_4 = [v_1, v_3, v_5]$. As in Section 2.1, we first determine three sets of mean value coordinates $\boldsymbol{b}_1$, $\boldsymbol{b}_2$, and $\boldsymbol{b}_3$ for these quadrilaterals, respectively, and then consider the blending functions

$$\mu_1, \mu_2, \mu_3 \colon \triangle_4 \setminus \{v_1, v_3, v_5\} \to [0,1],$$

such that

$$\mu_1(v) = \begin{cases} 1, & v \in (v_1, v_3), \\ 0, & v \in (v_3, v_5) \cup (v_5, v_1), \end{cases} \quad \mu_2(v) = \begin{cases} 1, & v \in (v_5, v_1), \\ 0, & v \in (v_1, v_3) \cup (v_3, v_5), \end{cases} \quad \mu_3(v) = \begin{cases} 1, & v \in (v_3, v_5), \\ 0, & v \in (v_5, v_1) \cup (v_1, v_3), \end{cases}$$

and $\mu_1(v) + \mu_2(v) + \mu_3(v) = 1$ for any $v \in \triangle_4 \setminus \{v_1, v_3, v_5\}$. Again, μ_1, μ_2, and μ_3 are not defined at the vertices of $\triangle_4$, but this does not affect the final construction of generalized barycentric coordinates. Now, given the triangle $\triangle_4$, if we define the classical barycentric coordinates $\lambda_{1,2,3} \colon \triangle_4 \to [0,1]$ corresponding to v_5, v_3, and v_1 as

$$\lambda_1(v) = \frac{\text{Area}[v_1, v_3, v]}{\text{Area}[v_1, v_3, v_5]}, \qquad \lambda_2(v) = \frac{\text{Area}[v_1, v, v_5]}{\text{Area}[v_1, v_3, v_5]}, \qquad \lambda_3(v) = \frac{\text{Area}[v, v_3, v_5]}{\text{Area}[v_1, v_3, v_5]} \qquad \forall\, v \in \triangle_4,$$

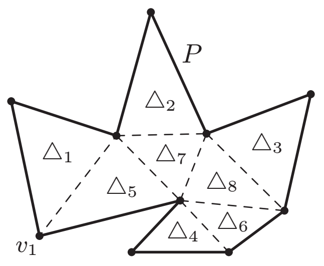

Figure 5: Example of a general polygon and its constrained Delaunay triangulation.

then we can construct the blending functions as

$$\mu_1 = \frac{\sigma_1}{\sigma_1+\sigma_2+\sigma_3}, \qquad \mu_2 = \frac{\sigma_2}{\sigma_1+\sigma_2+\sigma_3}, \qquad \mu_3 = \frac{\sigma_3}{\sigma_1+\sigma_2+\sigma_3} \tag{9}$$

with

$$\sigma_1 = (q\circ\lambda_2)(q\circ\lambda_3), \qquad \sigma_2 = (q\circ\lambda_3)(q\circ\lambda_1), \qquad \sigma_3 = (q\circ\lambda_1)(q\circ\lambda_2),$$

and q as defined in Section 2.1. The construction of the blended coordinate functions $b_i\colon P\setminus\{v_1, v_3, v_5\}\to\mathbb{R}$, $i = 1,\ldots,6$ (see Figure 4) is then analogous to the construction (8), and with the same arguments as above it can be shown that these functions satisfy the key properties (1) and (2), are non-negative, have the Lagrange property, even at the vertices v_1, v_3, and v_5, and are smooth.

2.3 Coordinates for arbitrary polygons

We are now ready to present the construction of blended barycentric coordinates for arbitrary simple polygons. Given the constrained Delaunay triangulation $\Delta = \{\triangle_1,\ldots,\triangle_m\}$ of a planar polygon $P = [v_1,\ldots,v_n]$ with $n > 6$ vertices, we consider all quadrilaterals defined by two triangles that share an interior edge, determine the mean value coordinates with respect to these quadrilaterals, and blend them as explained above. For example, for the polygon in Figure 5, the quadrilateral mean value coordinates are blended over the triangles $\triangle_1,\ldots,\triangle_6$ as in Section 2.1 and over the triangles $\triangle_7$ and $\triangle_8$ as in Section 2.2. In this way, we obtain coordinate functions with the same properties as before.

Since it is lengthy to write down the analytic expressions of the blended coordinate functions b_i as in (8), we do not present these formulas, but rather discuss how to evaluate them efficiently at any point v inside the polygon. Suppose we know the triangle $\triangle_j \in \Delta$ that contains v. We then compute the blended coordinates following a simple procedure with two steps. In the first step we find the k triangles in Δ that share an edge with $\triangle_j$. In the second step, depending on the number $k \in \{1,2,3\}$, we evaluate the $k+3$ functions b_i that correspond to the $k+3$ vertices of the k quadrilaterals that overlap at $\triangle_j$, using one of the three routines given in A. All other coordinates can be safely set to zero. On the one hand, this implies that the time complexity for the evaluation of all coordinates at any v is $O(1)$. On the other hand, it shows the locality of blended coordinates, because the support $\operatorname{supp}(b_i)$ of the coordinate function b_i is just the union of the triangles adjacent to v_i and their neighbouring triangles. For example, the support of b_1 in Figure 5 is $\operatorname{supp}(b_1) = \triangle_1 \cup \triangle_5 \cup \triangle_7$.

In addition to the constant cost of this evaluation procedure, the constrained Delaunay triangulation Δ of P needs to be computed in a preprocessing step in $O(n\log n)$ time, and, depending on the application, some time may be spent on finding the triangle $\triangle_j$ that contains v. We shall briefly discuss three possible scenarios. To generate the images in Figures 6, 7, and 8, we used seven linear subdivision steps to refine Δ, evaluated the coordinates at the vertices of this refined triangulation, and rendered the result. In this scenario, careful book-keeping during the subdivision process provides $\triangle_j$ for free, and the same holds in any application that allows to choose the evaluation points v per triangle of Δ. Another scenario is image warping, where the coordinates need to be evaluated for each pixel of the warped image. In this situation, $\triangle_j$ has to be found only once for the first pixel, and subsequently a local search with constant time complexity can be used to find $\triangle_j$ for the next pixel. Such a local search can also be used to evaluate the coordinates at the vertices of a (dense) triangulation of the polygon, if the vertices are visited, for example, by breadth-first traversal. In the worst case, if the application requires to compute coordinates at truly random points v, then $\triangle_j$ can be found in optimal $O(\log n)$ time [6] by using a hierarchy of Delaunay triangulations, and we report some timings for this situation below.

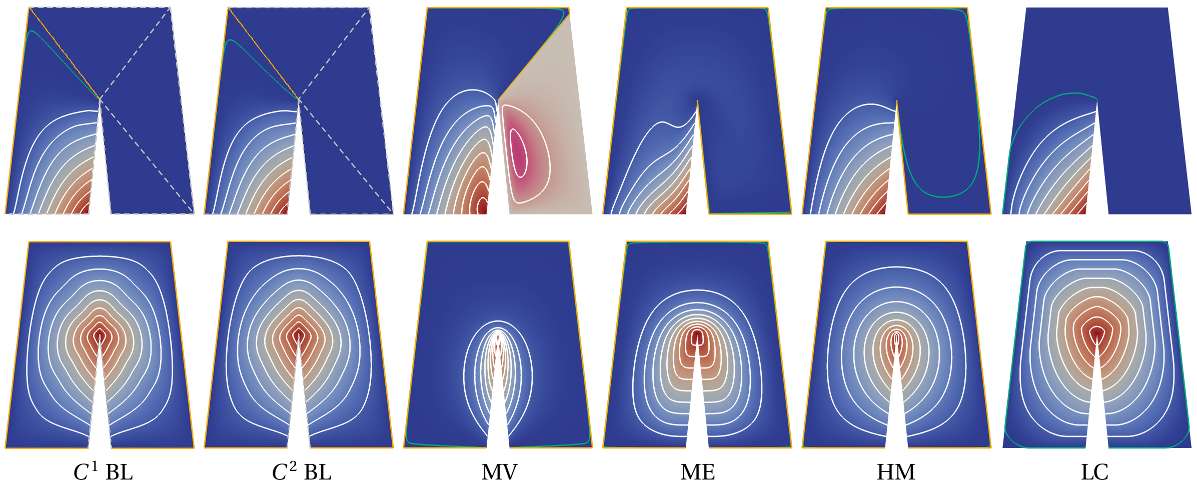

Figure 6: Comparison of different coordinate functions for a convex (top) and a concave (bottom) vertex. The white curves are the contour lines at $0.1, 0.2, \ldots, 0.9$ (and at -0.2 and -0.1 for mean value coordinates), the contour line at 10^{-4} is shown in green, and the orange line marks the support. The constrained Delaunay triangulation of the polygon for blended coordinates is outlined in grey, and the grey/magenta colour range shows the negative function values for mean value coordinates.

3 Comparison

Figures 6, 7, and 8 show a comparison of C^1 and C^2 continuous blended (BL) coordinates with mean value (MV), maximum entropy (ME), harmonic (HM), and local (LC) barycentric coordinates for three different polygons (with 7, 13, and 39 vertices, respectively) and coordinate functions associated to convex and concave vertices. While mean value coordinates can be negative inside the polygon, blended coordinates are always positive by definition. Maximum entropy and harmonic coordinates are also positive inside the polygon, but they are globally supported and can be computed only numerically. Instead, blended coordinates are local and have a closed form. Local barycentric coordinates also fulfill the locality requirement, but they can again be computed only numerically, and the exact support of these coordinate functions is not known. The numerical solver used to compute them generates small function values even outside the probable support,

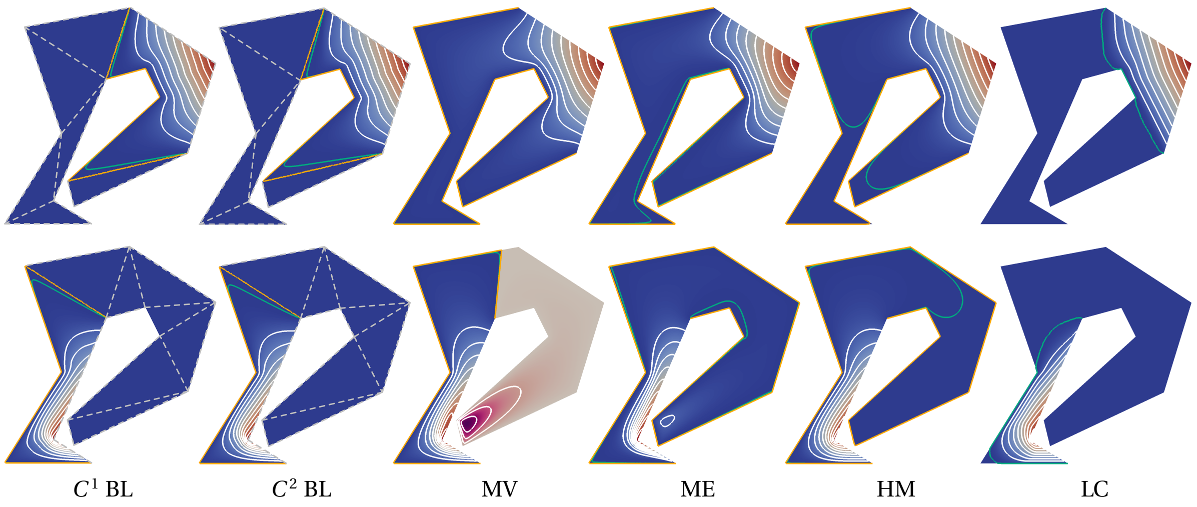

Figure 7: Comparison of different coordinate functions for a convex (top) and a concave (bottom) vertex. The white curves are the contour lines at $0.1, 0.2, \ldots, 0.9$ (and at $-0.3, -0.2, -0.1$ for mean value coordinates), the contour line at 10^{-4} is shown in green, and the orange line marks the support. The constrained Delaunay triangulation of the polygon for blended coordinates is outlined in grey, and the grey/magenta colour range shows the negative function values for mean value coordinates.

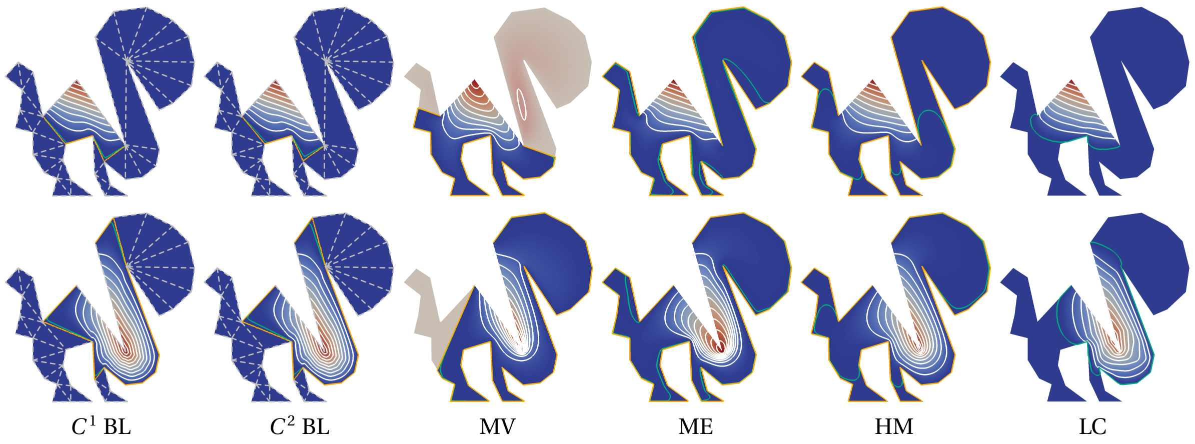

Figure 8: Comparison of different coordinate functions for a convex (top) and a concave (bottom) vertex. The white curves are the contour lines at $0.1, 0.2, \ldots, 0.9$ (and at -0.1 for mean value coordinates), the contour line at 10^{-4} is shown in green, and the orange line marks the support. The constrained Delaunay triangulation of the polygon for blended coordinates is outlined in grey, and the grey/magenta colour range shows the negative function values for mean value coordinates.

and Zhang et al. [28] suggest to treat all values below 10^{-4} as numerically zero. In turn, the exact support of blended coordinate functions is clearly defined.

We implemented all coordinates above in C++ on a MacBook Pro with 2.4 GHz Intel Core i7 processor and 8 GB RAM. For blended coordinates, we first build the constrained Delaunay triangulation of the polygon with *Triangle* [24] and then evaluate the coordinate functions as explained above, using the pseudocode in A. For mean value and maximum entropy coordinates, we implemented the pseudocodes given in [12, Section 5] and [13, Section 5], respectively. For harmonic coordinates, we use the sparse Cholesky decomposition in *Eigen* [11] to solve the linear system arising from the standard finite element discretization of the Laplace equation with Dirichlet boundary conditions, and local barycentric coordinates are computed with the code provided by Deng and Liu [5].

To compare the performance, we created two sets of test polygons. The first set is shown in Figure 9 and consists of 5 concave polygons with $n = 6l + 2$ vertices that are composed by concatenating $l = 2^p$ copies of the piece shown on the left, for $p = 0, \ldots, 4$. The constrained Delaunay triangulations of these polygons have exactly 2 triangles with one neighbour, $6l - 2$ triangles with two neighbours, and no triangle with three neighbours. The second set in Figure 10 consists of 5 isotoxal star polygons with the same numbers of vertices. In this case, the constrained Delaunay triangulations have $3l + 1$ triangles with one neighbour, no triangles with two neighbours, and $3l - 1$ triangles with three neighbours.

For all test polygons, we evaluated the different coordinates at the 50,000 interior vertices of a dense Delaunay triangulation, and the results are summarized in Figures 11 and 12. In case of blended coordinates, the given times correspond to the construction of the constrained Delaunay triangulation, which is less than 0.0004 sec, even for $n = 98$, plus the evaluation of the n coordinate functions at all evaluation points, where the difference between using q_1 or q_2 was not noticeable. For mean value and maximum entropy coordinates, we report the pure evaluation times, and for harmonic coordinates we added the times for assembling the matrix, factorizing it, and solving the linear system for all n coordinates with back substitution. In case of local barycentric coordinates, the solver did not converge for such a dense triangulation, and so we decided to use instead a Delaunay triangulation with only 500 interior vertices. The given times include the initialization of the solver and running it for a fixed number of 500 iterations, which is barely enough to guarantee convergence for $n = 8$. Even for this comparatively small number of interior vertices, the timings are about one order of magnitude slower, and they would be even worse if the solver would be allowed to run until convergence. To give an idea about the worst-case scenario for blended coordinates regarding the additional cost for finding the triangles that contain the query points, we also report the time needed for this triangle search (TS) with the hierarchical Delaunay triangulation strategy implemented in CGAL [27], and show the overall cost (BL+TS) in the plots.

The data confirms that the evaluation of blended coordinates has constant time complexity, with an additional $O(\log n)$ cost for the triangle search in the worst case, while for all other coordinates the evaluation

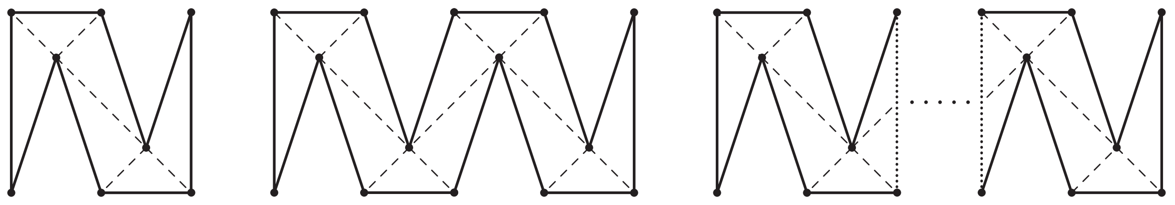
Figure 9: Concave test polygons, composed of 1, 2, and l pieces with 8, 14, and $n = 6l + 2$ vertices, respectively.

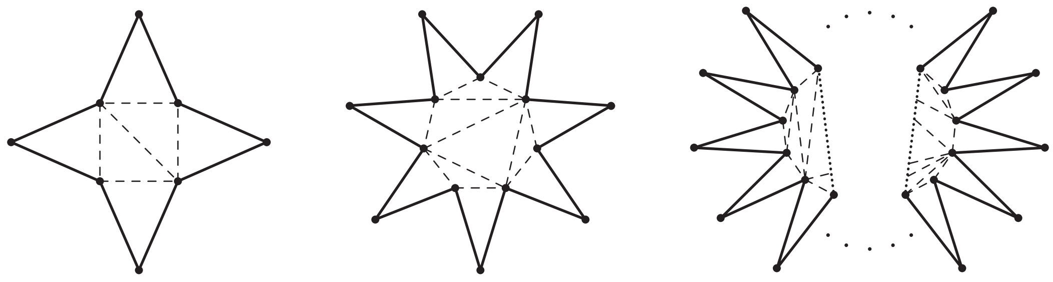
Figure 10: Isotoxal star polygons with 8, 14, and n vertices, respectively.

time depends linearly on n. It also shows that blended coordinates are on par with mean value coordinates, the fastest competitor, even for small values of n. In a nutshell, the pure evaluation of blended coordinates is faster than that of all other coordinates for polygons with $n \geq 10$ vertices, and with triangle search the break-even point is around $n = 35$.

Another observation is that blended coordinates have approximately the same timings for both sets of test polygons. To explain this behaviour, remember that for the polygons in Figure 9, most of the query points lie in triangles with two neighbours, while for the polygons in Figure 10, about half the query points are contained in triangles with one neighbour, and the other half in triangles with three neighbours. Counting instructions, it now turns out that two calls to Pseudocode 2 are about as expensive as executing Pseudocode 1 and Pseudocode 3 once each. Similar timings are actually to be expected for other polygons with the same number of vertices, because of a simple fact that follows from Euler's formula. If we denote the number of triangles with $k = 1, 2, 3$ neighbours in the constrained Delaunay triangulation of P by m_k, then it follows that $m_3 = m_1 - 2$, that is, triangles with one and three neighbours always come in pairs. Consequently, Pseudocodes 1 and 3 are always called similarly often for random evaluation points inside an arbitrary

	$n=8$	$n=14$	$n=26$	$n=50$	$n=98$
BL	0.025	0.026	0.025	0.026	0.026
MV	0.024	0.029	0.039	0.061	0.104
ME	0.058	0.083	0.135	0.237	0.448
HM	0.163	0.163	0.174	0.218	0.315
LC	*0.290*	*0.535*	*1.028*	*2.206*	*5.706*
TS	0.019	0.019	0.020	0.023	0.028

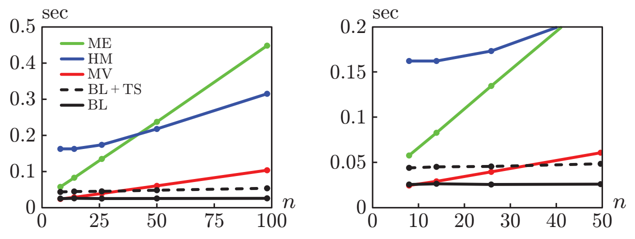

Figure 11: Timings for the polygons in Figure 9.

	$n=8$	$n=14$	$n=26$	$n=50$	$n=98$
BL	0.026	0.025	0.026	0.026	0.026
MV	0.024	0.029	0.040	0.061	0.104
ME	0.058	0.083	0.136	0.240	0.451
HM	0.202	0.194	0.207	0.244	0.326
LC	*0.268*	*0.501*	*0.964*	*2.437*	*8.501*
TS	0.019	0.019	0.019	0.021	0.022

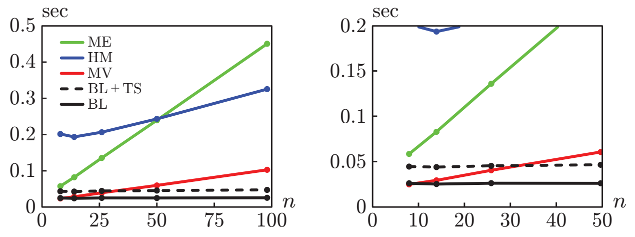

Figure 12: Timings for the polygons in Figure 10.

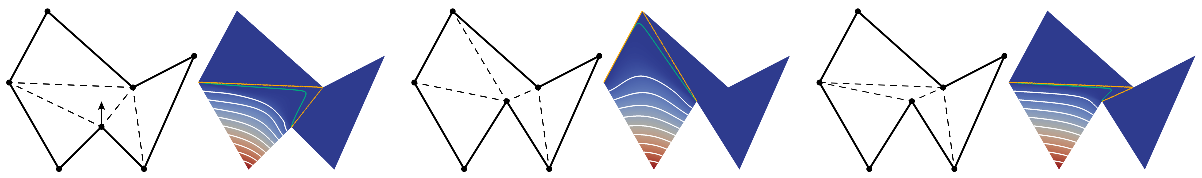
Figure 13: Moving a vertex of the polygon (left) can lead to a different constrained Delaunay triangulation and a discontinuous change of the coordinate functions (middle). This can be overcome by keeping the triangulation (right), as long as no triangle folds over, even if the triangulation becomes non-Delaunay.

polygon, and the average evaluation time is therefore close to that of Pseudocode 2. Comparing the number of instructions in this routine with those for mean value coordinates further explains the break-even point at around $n = 10$.

4 Conclusion

Blending approaches are frequently used in geometric modelling as a promising recipe for getting interpolants that inherit certain global properties from corresponding local properties and are efficient to evaluate. Our construction follows this idea and can actually be seen as a bivariate variant of Catmull–Rom splines [4], with the quadrilateral mean value coordinates taking the role of the local polynomial interpolants and the compactly supported B-spline blending functions replaced by the blending functions μ_i per triangle. Our construction has four crucial ingredients. First, the non-negativity and partition of unity property of the blending functions guarantees that the properties of the local quadrilateral mean value coordinates carry over to the blended coordinate functions for the whole polygon. Second, the $k > 0$ vanishing derivatives of the blending functions across the edges of the blending region provide C^k continuity of the coordinates b_i. Third, the non-negativity of mean value coordinates for arbitrary quadrilaterals is the key for obtaining b_i that are non-negative globally. And finally, the locality of the construction leads to favourable computational cost. In principle, one could use other barycentric coordinates as the main building blocks of our construction, as long as they are well-defined and non-negative for arbitrary quadrilaterals. However, to the best of our knowledge, mean value coordinates are the only known coordinates with these properties and a closed-form definition, and using computational coordinates for this purpose would compromise the efficiency of the approach.

Blended coordinates also have a few drawbacks. First, they do not depend continuously on the vertices of the polygon, because even a small perturbation of some v_i can lead to a different constrained Delaunay triangulation and a discontinuous change of the coordinate functions b_i (see Figure 13, middle). This problem can be overcome to some extent by keeping the triangulation, because the construction clearly works for non-Delaunay triangulations, too (see Figure 13, right), but only as long as the triangles of the triangulation do not flip. However, in many applications it is the data associated with the vertices of v_i that changes, while the polygon and its triangulation are fixed, and then the behaviour of the barycentric interpolant is smooth. For example, interactively changing the vertices of the target polygon in an image warping application will not introduce any unexpected, discontinuous behaviour. Second, even for perfectly symmetric polygons, like the ones in Figure 10, the coordinate functions are not symmetric, because the constrained Delaunay triangulation is not. One way to resolve this problem would be to average the coordinate functions with respect to all possible triangulations of the polygon, but since the number of such triangulations grows exponentially with n, this approach is computationally feasible only for polygons with a small number of vertices. Another option is to add interior points to the triangulation of P before defining the blended coordinates. Unfortunately, our coordinates are discontinuous at interior points, and it remains future work to come up with a different blending construction that can deal with interior points. This would then also open the door to an extension to 3D, where interior points may be necessary for triangulating the given polyhedron, as in the case of the Schönhardt polyhedron [23].

Acknowledgements

We thank the anonymous reviewers for their valuable comments and suggestions, which helped to improve this paper. We further thank Teseo Schneider and Olga Sorkine-Hornung for useful discussions.

References

[1] D. Anisimov, C. Deng, and K. Hormann. Subdividing barycentric coordinates. *Computer Aided Geometric Design*, 43:172–185, 2016.

[2] A. Belyaev. On transfinite barycentric coordinates. In *Proceedings of SGP*, pages 89–99, 2006.

[3] 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, 2016.

[4] E. Catmull and R. Rom. A class of local interpolating splines. In *Computer Aided Geometric Design*, pages 317–326. Academic Press, New York, 1974.

[5] B. Deng and Z. Liu. Local barycentric coordinates solver. `https://github.com/bldeng/LBC`, Mar. 2015. [Online; accessed 28-September-2016].

[6] O. Devillers. Improved incremental randomized Delaunay triangulation. In *Proceedings of SoCG*, pages 106–115, 1998.

[7] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *Proceedings of SIGGRAPH*, pages 173–182, 1995.

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

[9] 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, 2006.

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

[11] G. Guennebaud, B. Jacob, et al. Eigen 3.2.9. `http://eigen.tuxfamily.org`, July 2016. [Online; accessed 28-September-2016].

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

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

[14] K. Hormann and M. Tarini. A quadrilateral rendering primitive. In *Proceedings of Graphics Hardware*, pages 7–14, 2004.

[15] 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, 2007.

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

[17] Y. Lipman, J. Kopf, D. Cohen-Or, and D. Levin. GPU-assisted positive mean value coordinates for mesh deformations. In *Proceedings of SGP*, pages 117–123, 2007.

[18] 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.

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

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

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

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

[23] E. Schönhardt. Über die Zerlegung von Dreieckspolyedern in Tetraeder. *Mathematische Annalen*, 98(1):309–312, 1928.

[24] J. R. Shewchuk. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In M. C. Lin and D. Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer, Berlin, 1996.

[25] N. Sukumar and E. A. Malsch. Recent advances in the construction of polygonal finite element interpolants. *Archives of Computational Methods in Engineering*, 13(1):129–163, 2006.

[26] E. L. Wachspress. *A Rational Finite Element Basis*, volume 114 of *Mathematics in Science and Engineering*. Academic Press, New York, 1975.

[27] M. Yvinec. CGAL 4.9: 2D Triangulation. `http://doc.cgal.org/4.9/Manual/packages.html#PkgTriangulation2Summary`, Sept. 2016. [Online; accessed 28-September-2016].

[28] 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, 2014.

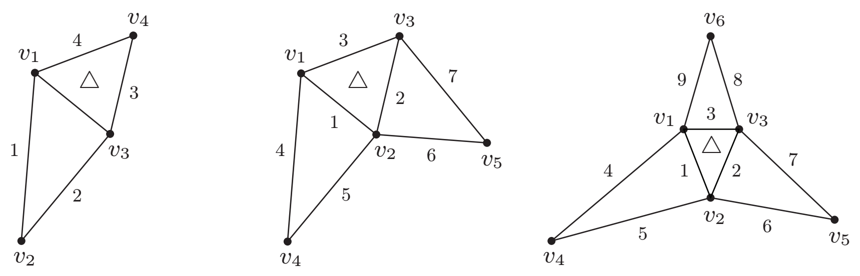

Figure 14: Local vertex and edge indices used in the Pseudocodes 1 (left), 2 (centre), and 3 (right).

A Efficient evaluation of blended coordinates

In this appendix we provide the pseudocodes for the efficient evaluation of the non-zero blended coordinate functions b_i at a given point v inside a triangle $\triangle$ with one, two, or three overlapping quadrilaterals. In all three pseudocodes, we use i and j to denote the vertex and edge indices as shown in Figure 14, and k is reserved for indexing the related variables inside each quadrilateral. For the sake of clarity, we decided to use only local indices in the pseudocodes, and the superindices "+" and "−" refer to the "next" and "previous" local indices. Therefore, an actual implementation must take special care of correctly mapping all local indices to the corresponding global indices. For example, in Pseudocode 2, the local indices i and i^+ for $j = 4$ in the second loop over all edges are the local vertex indices 1 and 4 (see Figure 14, centre), which in turn refer to certain global vertex indices, depending on $\triangle$.

Regarding notation, we follow [12] and denote the vector from the query point v to the vertex v_i by s_i, the length of this vector by r_i, the signed area of the triangle $[v_i, v_{i^+}, v]$ by $A_i/2$, and the dot product of s_i and s_{i^+} by D_i. However, instead of computing the tangents of the half-angles between s_i and s_{i^+} via

$$t_i = \tan(\alpha_i/2) = \frac{1-\cos(\alpha_i)}{\sin(\alpha_i)} = \frac{r_i\, r_{i^+} - D_i}{A_i},$$

we use the formula

$$t_i = \tan(\alpha_i/2) = \frac{\sin(\alpha_i)}{1+\cos(\alpha_i)} = \frac{A_i}{r_i\, r_{i^+} + D_i},$$

which works without exceptions even for interior query points with $A_i = 0$.

Pseudocode 1 Case: one quadrilateral

```
1:  function b(v)
2:      for $i = 1$ to 4 do                              ▷ Iterate over all vertices
3:          $s_i := v_i - v$
4:          $r_i := \|s_i\|$
5:      for $j = 1$ to 4 do                              ▷ Iterate over all edges
6:          $A_j := \det(s_i, s_{i^+})$
7:          $D_j := \langle s_i, s_{i^+} \rangle$
8:          $t_j := A_j / (r_i\, r_{i^+} + D_j)$
9:      $W := 0$                                          ▷ Mean value weights
10:     for $k = 1$ to 4 do
11:         $w_k := (t_{j^-} + t_j) / r_i$
12:         $W := W + w_k$
13:     for $i = 1$ to 4 do                              ▷ Blended coordinates
14:         $b_i := w_k / W$
```

Pseudocode 2 Case: two overlapping quadrilaterals

```
function $\boldsymbol{b}(v)$
    for $i = 1$ to $5$ do                                                    ▷ Iterate over all vertices
        $s_i := v_i - v$
        $r_i := \|s_i\|$
    for $j = 1$ to $7$ do                                                    ▷ Iterate over all edges
        $A_j := \det(s_i, s_{i^+})$
        $D_j := \langle s_i, s_{i^+} \rangle$
        $t_j := A_j / (r_i r_{i^+} + D_j)$
    $A := A_1 + A_2 + A_3$,   $a_1 := A_1 / A$,   $a_2 := A_2 / A$              ▷ Blending functions
    $\sigma_1 := q(a_2)$,   $\sigma_2 := q(a_1)$,   $\Sigma := \sigma_1 + \sigma_2$
    $\mu_1 := \sigma_1 / \Sigma$,   $\mu_2 := \sigma_2 / \Sigma$
    $W_1 := 0$,   $W_2 := 0$                               ▷ Mean value weights for both quadrilaterals
    for $k = 1$ to $4$ do
        $w_k^1 := (t_{j^-}^1 + t_j^1) / r_i^1$,   $w_k^2 := (t_{j^-}^2 + t_j^2) / r_i^2$
        $W_1 := W_1 + w_k^1$,   $W_2 := W_2 + w_k^2$
    for $k = 1$ to $4$ do                                                    ▷ Mean value coordinates
        $b_k^1 := w_k^1 / W_1$,   $b_k^2 := w_k^2 / W_2$
    for $i = 1$ to $3$ do                                                    ▷ Blended coordinates
        $b_i := b_k^1 \mu_1 + b_k^2 \mu_2$
    $b_4 := b_4^1 \mu_1$,   $b_5 := b_4^2 \mu_2$
```

Pseudocode 3 Case: three overlapping quadrilaterals

```
function $\boldsymbol{b}(v)$
    for $i = 1$ to $6$ do                                                    ▷ Iterate over all vertices
        $s_i := v_i - v$
        $r_i := \|s_i\|$
    for $j = 1$ to $9$ do                                                    ▷ Iterate over all edges
        $A_j := \det(s_i, s_{i^+})$
        $D_j := \langle s_i, s_{i^+} \rangle$
        $t_j := A_j / (r_i r_{i^+} + D_j)$
    $A := A_1 + A_2 + A_3$,   $a_1 := A_1 / A$,   $a_2 := A_2 / A$,   $a_3 := A_3 / A$,   ▷ Blending functions
    $\sigma_1 := q(a_2) q(a_3)$,   $\sigma_2 := q(a_1) q(a_2)$,   $\sigma_3 := q(a_1) q(a_3)$,   $\Sigma := \sigma_1 + \sigma_2 + \sigma_3$
    $\mu_1 := \sigma_1 / \Sigma$,   $\mu_2 := \sigma_2 / \Sigma$,   $\mu_3 := \sigma_3 / \Sigma$
    $W_1 := 0$,   $W_2 := 0$,   $W_3 := 0$                ▷ Mean value weights for all three quadrilaterals
    for $k = 1$ to $4$ do
        $w_k^1 := (t_{j^-}^1 + t_j^1) / r_i^1$,   $w_k^2 := (t_{j^-}^2 + t_j^2) / r_i^2$,   $w_k^3 := (t_{j^-}^3 + t_j^3) / r_i^3$
        $W_1 := W_1 + w_k^1$,   $W_2 := W_2 + w_k^2$,   $W_3 := W_3 + w_k^3$
    for $k = 1$ to $4$ do                                                    ▷ Mean value coordinates
        $b_k^1 := w_k^1 / W_1$,   $b_k^2 := w_k^2 / W_2$,   $b_k^3 := w_k^3 / W_3$
    for $i = 1$ to $3$ do                                                    ▷ Blended coordinates
        $b_i := b_k^1 \mu_1 + b_k^2 \mu_2 + b_k^3 \mu_3$
    $b_4 := b_4^1 \mu_1$,   $b_5 := b_4^2 \mu_2$,   $b_6 := b_4^3 \mu_3$
```