Analysis and New Constructions of Generalized Barycentric Coordinates in 2D

Doctoral Dissertation submitted to the
Faculty of Informatics of the Università della Svizzera italiana
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

presented by
Dmitry Anisimov

under the supervision of
Kai Hormann

May 23, 2017

Dissertation Committee

Kai Hormann	Università della Svizzera italiana, Switzerland
Matthias Hauswirth	Università della Svizzera italiana, Switzerland
Evanthia Papadopoulou	Università della Svizzera italiana, Switzerland
Michael Floater	University of Oslo, Norway
Pierre Alliez	INRIA, Sophia-Antipolis - Méditerranée, France

Dissertation accepted on May 23, 2017

Research Advisor	PhD Program Director
Kai Hormann	**Walter Binder**

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work, which has been carried out since the official commencement date of the approved research program.

Dmitry Anisimov
Lugano, May 23, 2017

To my parents

Nothingness is being and being nothingness. . . Our limited mind can not grasp or fathom this, for it joins infinity
– Azrael of Gerona

Abstract

Different coordinate systems allow to uniquely determine the position of a geometric element in space. In this dissertation, we consider a coordinate system that lets us determine the position of a two-dimensional point in the plane with respect to an arbitrary simple polygon. Coordinates of this system are called generalized barycentric coordinates in 2D and are widely used in computer graphics and computational mechanics.

There exist many coordinate functions that satisfy all the basic properties of barycentric coordinates, but they differ by a number of other properties. We start by providing an extensive comparison of all existing coordinate functions and pointing out which important properties of generalized barycentric coordinates are not satisfied by these functions. This comparison shows that not all of existing coordinates have fully investigated properties, and we complete such a theoretical analysis for a particular one-parameter family of generalized barycentric coordinates for strictly convex polygons. We also perform numerical analysis of this family and show how to avoid computational instabilities near the polygon's boundary when computing these coordinates in practice. We conclude this analysis by implementing some members of this family in the Computational Geometry Algorithm Library.

In the second half of this dissertation, we present a few novel constructions of non-negative and smooth generalized barycentric coordinates defined over any simple polygon. In this context, we show that new coordinates with improved properties can be obtained by taking convex combinations of already existing coordinate functions and we give two examples of how to use such convex combinations for polygons without and with interior points. These new constructions have many attractive properties and perform better than other coordinates in interpolation and image deformation applications.

Acknowledgements

When I was finishing my master program in Russia, I did not speak English well and could only dream about having a PhD abroad in this international language. However, my dream came true thanks to many different people who have been a great support for me in this endeavour.

My story began in 2010 when professor Valentina Petrova from my department at Saint-Petersburg State University sent me a link to the website of the Faculty of Informatics of Università della Svizzera italiana (USI) in Lugano and I am very grateful to her for this initiative! Then I would like to thank professor Laura Pozzi who, regardless of the distance between our countries, managed to see in me a promising candidate for PhD at USI and invited me here in 2011.

The second important step for finishing this PhD was a meeting with professor Kai Hormann in 2012, and I am very thankful to him for being my advisor, proposing this exact topic for my dissertation, and spending hours and hours every week teaching me how to approach complex problems, solve them, and explain to other people my solution in a simple and intuitive way. Since my PhD involved quite a lot of programming and experimenting, I would also like to say thank you to Randolf Schärfig for helping me a lot in all my programming tasks and to Teseo Schneider who was not only my collaborator in some scientific results but also taught me how to write understandable and reusable code.

Finally, an important part of my PhD from 2012 to 2017 is taken by my other collaborators David Bommes from the Aachen Institute for Advanced Study in Computational Engineering Science, Pierre Alliez from the National Institute for Research in Computer Science and Control (Inria), Daniele Panozzo from the Courant Institute of Mathematical Sciences, and Chongyang Deng from the Hangzhou Dianzi University. Without many ideas that they proposed and long discussions that we had, this dissertation would not be possible.

I would also like to thank my colleagues Emiliano Cirillo, Filomena di Tomasso, and Elena Volontè for always keeping my spirit up in the office, the Dean's office of USI and especially Elisa Larghi and Janine Caggiano for always helping me with different documents and formal procedures, USI itself and all its professors

for being so international and sharing knowledge with the world, the committee members for finding time to read this long document, and all my friends, especially Elena Khramtcova, Jelena Milošević, Artiom Kovnatsky, and Dmitry Cherezov, who I met in Lugano and with whom I always had a lot of fun.

And of course, none of this would ever happen if it were not my father Vladimir Anisimov, my mother Nina Anisimova, and my wife Darya Kurpel who always support me in my life! (and Betmen if you know what I mean)

Contents

Chapter 1

Introduction

It is known since the days of *Peripatetic School* and usually contributed to *Archimedes* (c. 287 BC – c. 212 BC) [d. Monantheuil, 1599] that a lever $[v_1, v_2]$ with two weights w_1 and w_2 attached to its ends is balanced when a fulcrum is placed at the point $p \in [v_1, v_2]$ such that

$$w_1 l_1 = w_2 l_2, \tag{1.1}$$

where $l_1 = p - v_1$ and $l_2 = v_2 - p$ (see Figure 1.1). Equation (1.1) is called the *law of the lever* and the point of balance p is called the *centre of mass* of this lever or its *barycentre* (from Ancient Greek βάρος = "weight" and κέντρον = "centre"). The weights w_1 and w_2 are often called homogeneous, because multiplying them with a common non-zero scalar α does not change the equation. Rearranging terms, (1.1) can be written in the form

$$w_1(v_1 - p) + w_2(v_2 - p) = 0 \tag{1.2}$$

and further as

$$w_1 v_1 + w_2 v_2 = W p, \qquad W = w_1 + w_2.$$

Choosing the scalar $\alpha = \frac{1}{W}$, we can define the normalized weights $b_1 = \alpha w_1$ and $b_2 = \alpha w_2$ and write the barycentre p as an affine combination of the ends of the

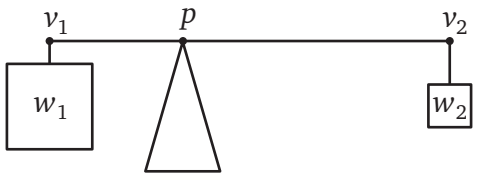


Figure 1.1. Law of the lever.

lever with these weights,

$$b_1 + b_2 = 1, \tag{1.3}$$
$$b_1 v_1 + b_2 v_2 = p. \tag{1.4}$$

The normalized weights b_1 and b_2 are called the *barycentric coordinates* of the point p with respect to the segment $[v_1, v_2]$.

While the problem above is about finding the barycentre p for the given weights, it is also interesting to study the opposite problem. Given the end points v_1 and v_2 of an arbitrary segment and some point p along this segment, how do we find the barycentric coordinates b_1 and b_2 of p with respect to this segment? It turns out that they are uniquely determined by (1.3) and (1.4) as ratios of lengths,

$$b_1 = \frac{l_2}{l}, \qquad b_2 = \frac{l_1}{l},$$

where $l = l_1 + l_2 = v_2 - v_1$ is the length of the segment.

1.1 Barycentric coordinates for simplices

In 1827, the German mathematician *August Ferdinand Möbius* (1790 – 1868) considered the problem of finding barycentric coordinates with respect to an arbitrary d-simplex in $d \in \mathbb{N}$ dimensions [Möbius, 1827]. For example, a 1-simplex is a line segment in 1D, a 2-simplex is a triangle in 2D, and a 3-simplex is a tetrahedron in 3D (see Figure 1.2).

Given a non-degenerate d-simplex $\triangle$ with $d+1$ vertices $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{d+1} \in \mathbb{R}^d$, we search for $d+1$ functions $\boldsymbol{b} = [b_1, \ldots, b_{d+1}] \colon \triangle \to \mathbb{R}^{d+1}$, which satisfy the *partition of unity property*

$$\sum_{i=1}^{d+1} b_i(\boldsymbol{p}) = 1 \qquad \forall \boldsymbol{p} \in \triangle \tag{1.5}$$

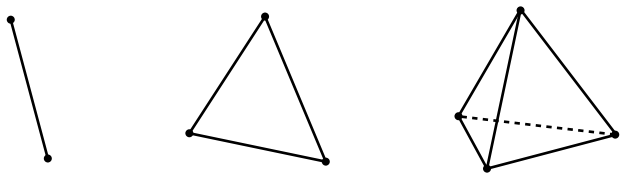


Figure 1.2. From left to right: 1-simplex, 2-simplex, 3-simplex.

and the *linear reproduction property*

$$\sum_{i=1}^{d+1} b_i(\boldsymbol{p})\boldsymbol{v}_i = \boldsymbol{p} \qquad \forall \boldsymbol{p} \in \triangle. \tag{1.6}$$

The functions $b_1, \ldots, b_{d+1}$ are called *barycentric coordinates* with respect to $\triangle$.

Analogously to the one-dimensional case that we discussed earlier, it turns out that these barycentric coordinates are uniquely determined by (1.5) and (1.6), and Möbius shows that they are ratios of volumes,

$$b_i(\boldsymbol{p}) = \frac{V_i(\boldsymbol{p})}{V}, \qquad i = 1, \ldots, d+1, \tag{1.7}$$

where $V_i(\boldsymbol{p}) = \mathrm{Vol}[\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{i-1}, \boldsymbol{p}, \boldsymbol{v}_{i+1}, \ldots, \boldsymbol{v}_{d+1}]$ are the volumes of the corresponding d-simplices and $V = V_1(\boldsymbol{p}) + \cdots + V_{d+1}(\boldsymbol{p}) = \mathrm{Vol}[\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{d+1}]$ is the volume of $\triangle$ and does not depend on $\boldsymbol{p}$.

For example, let simplex $\triangle$ be a triangle in the plane whose vertices $\boldsymbol{v}_1$, $\boldsymbol{v}_2$, and $\boldsymbol{v}_3$ are given in anticlockwise direction, and we treat the vertex indices cyclically, that is $\boldsymbol{v}_{i+3k} = \boldsymbol{v}_i$ for $i \in \{1, 2, 3\}$ and $k \in \mathbb{Z}$ (see Figure 1.3). If we let $\boldsymbol{p} = (x, y)$ and $\boldsymbol{v}_i = (x_i, y_i)$, we can find the signed areas $A_1(\boldsymbol{p})$, $A_2(\boldsymbol{p})$, and $A_3(\boldsymbol{p})$ of the triangles opposite to the vertices $\boldsymbol{v}_1$, $\boldsymbol{v}_2$, and $\boldsymbol{v}_3$, respectively, through the determinant definition of the signed triangle area

$$A_i(\boldsymbol{p}) = \mathrm{Area}(\boldsymbol{p}, \boldsymbol{v}_i, \boldsymbol{v}_{i+1}) = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ x & x_i & x_{i+1} \\ y & y_i & y_{i+1} \end{vmatrix}, \qquad i = 1, 2, 3.$$

Since $A_i(\boldsymbol{p})$ are exactly the volumes $V_i(\boldsymbol{p})$ used in (1.7) with $d = 2$, we obtain the barycentric coordinates with respect to $\triangle$ as

$$\lambda_i(\boldsymbol{p}) = \frac{A_i(\boldsymbol{p})}{A}, \qquad i = 1, 2, 3,$$

where $A = A_1(\boldsymbol{p}) + A_2(\boldsymbol{p}) + A_3(\boldsymbol{p})$ is the total area of $\triangle$ and λ_i is a special notation for barycentric coordinates computed over triangles.

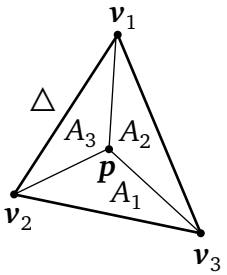


Figure 1.3. Notation used for barycentric coordinates with respect to a triangle.

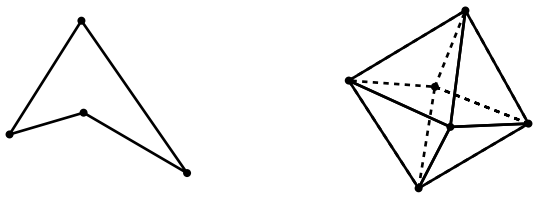


Figure 1.4. From left to right: 2-polytope, 3-polytope.

1.2 Generalized barycentric coordinates

To the best of our knowledge, Kalman [1961] was the first to propose a generalization of barycentric coordinates to convex polyhedra, and in recent years there has been a growing interest in the problem of finding barycentric coordinates with respect to arbitrary polytopes (for example, 2-polytope is a polygon and 3-polytope is a polyhedron, as shown in Figure 1.4). Throughout this dissertation, we consider a non-degenerate polytope P with $n \geq d+1$ vertices $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n \in \mathbb{R}^d$. We denote the boundary of this polytope by ∂P, its interior, viewed as an open set, by $\Omega \subset \mathbb{R}^2$ and its closure by $\bar{\Omega}$, so that $\bar{\Omega}$ is the *convex hull* of the vertices.

Definition 1. Given the polytope P, the n functions $\boldsymbol{b} = [b_1, \ldots, b_n] \colon \bar{\Omega} \to \mathbb{R}^n$ are called *generalized barycentric coordinates* if they satisfy the *partition of unity property*

$$\sum_{i=1}^{n} b_i(\boldsymbol{p}) = 1 \qquad \forall \boldsymbol{p} \in \bar{\Omega} \tag{1.8}$$

and the *linear reproduction property*

$$\sum_{i=1}^{n} b_i(\boldsymbol{p}) \boldsymbol{v}_i = \boldsymbol{p} \qquad \forall \boldsymbol{p} \in \bar{\Omega}. \tag{1.9}$$

For $n = d+1$, the only functions that satisfy the properties in Definition 1 are the b_i in (1.7). For $n > d+1$, however, the b_i are no longer uniquely determined, which is the reason for the existence of the different constructions of generalized barycentric coordinates that we review in Chapter 2.

In addition to the defining properties (1.8) and (1.9), it is often desirable for the functions b_i to have a few extra properties,

- *Non-negativity*: $b_i(\boldsymbol{p}) \geq 0$ for any $\boldsymbol{p} \in \bar{\Omega}$; (1.10a)
- *Lagrange property*: $b_i(\boldsymbol{v}_j) = \delta_{ij}$, where δ_{ij} is the *Kronecker delta*; (1.10b)
- *Linearity on the boundary*: b_i is linear on each facet of P; (1.10c)
- *Smoothness*: $b_i \in C^\infty$. (1.10d)

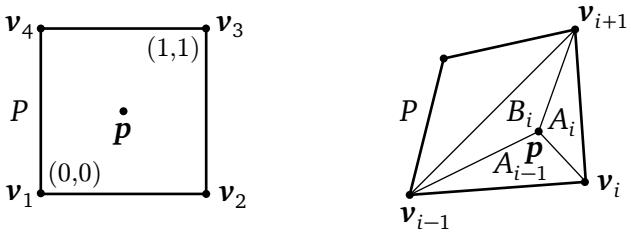


Figure 1.5. Notation used for barycentric coordinates with respect to the unit square (left) and to an arbitrary convex quadrilateral (right).

We remark that all these properties are satisfied in the case $n = d + 1$ for the linear functions b_i in (1.7).

For a simple example of generalized barycentric coordinates in 2D, consider the unit square $P = [0,1] \times [0,1]$ with the vertices $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_4$, which are given in anticlockwise direction, and we treat the vertex indices cyclically, that is $\boldsymbol{v}_{i+4k} = \boldsymbol{v}_i$ for $i \in \{1, \ldots, 4\}$ and $k \in \mathbb{Z}$ (see Figure 1.5, left). If we let $\boldsymbol{p} = (x, y)$, then we can define the barycentric coordinates with respect to P as

$$b_1(\boldsymbol{p}) = (1-x)(1-y), \quad b_2(\boldsymbol{p}) = x(1-y), \quad b_3(\boldsymbol{p}) = xy, \quad b_4(\boldsymbol{p}) = (1-x)y.$$

These *unit coordinates* (see Figure 1.6, left) clearly satisfy all properties above.

The latter coordinates can be generalized to an arbitrary convex quadrilateral (see Figure 1.6, right) by viewing this quadrilateral as the image of a bilinear map from the unit square [Floater, 2015]. It then follows that for each $\boldsymbol{p}$ there exist some parameters $s, t \in (0,1)$ such that the coordinate functions b_i are defined as

$$b_1(\boldsymbol{p}) = (1-s)(1-t), \quad b_2(\boldsymbol{p}) = s(1-t), \quad b_3(\boldsymbol{p}) = st, \quad b_4(\boldsymbol{p}) = (1-s)t.$$

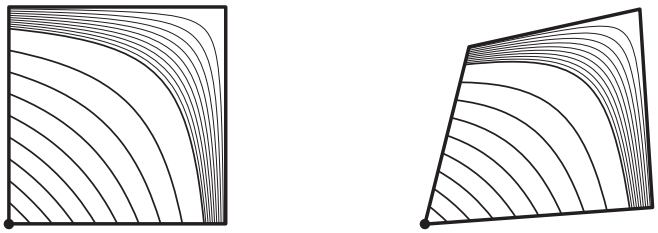


Figure 1.6. Unit coordinates (left) and bilinear coordinates (right) with respect to the marked vertex of the corresponding quadrilateral. The thick and the thin curves represent contours for the function values 0.0, 0.1, ..., 0.9 and 0.01, 0.02, ..., 0.09, respectively.

Floater [2015] shows how to find s and t as functions of $\boldsymbol{p}$ and presents a simple unified formula for these *bilinear coordinates*

$$b_i(\boldsymbol{p}) = \frac{4A_{i+1}(\boldsymbol{p})A_{i+2}(\boldsymbol{p})}{G_{i+1}(\boldsymbol{p})G_{i+2}(\boldsymbol{p})}, \qquad i = 1, \dots, 4,$$

where

$$G_i = 2A_i - B_i - B_{i+1} + \sqrt{B_1^2 + B_2^2 + 2A_1A_3 + 2A_2A_4}, \qquad i = 1, \dots, 4,$$

and A_i and B_i are the signed areas shown in Figure 1.5, right.

There are two types of generalized barycentric coordinates. On the one hand, there are coordinates with a *closed form*. Except for the direct definition as of unit and bilinear coordinates above, analogously to (1.2), the closed-form definition is often based on certain *weight functions* $\boldsymbol{w} = [w_1, \dots, w_n] \colon \Omega \to \mathbb{R}^n$, which satisfy

$$\sum_{i=1}^{n} w_i(\boldsymbol{p})(\boldsymbol{v}_i - \boldsymbol{p}) = \boldsymbol{0} \qquad \forall \boldsymbol{p} \in \Omega. \tag{1.11}$$

These weight functions are also called *homogeneous coordinates*, because normalizing them gives the generalized barycentric coordinates

$$b_i(\boldsymbol{p}) = \frac{w_i(\boldsymbol{p})}{W(\boldsymbol{p})}, \qquad W(\boldsymbol{p}) = \sum_{j=1}^{n} w_j(\boldsymbol{p}), \qquad i = 1, \dots, n. \tag{1.12}$$

On the other hand, there are *computational coordinates* that can only be obtained numerically, for example, by solving an optimization problem. We show more examples of closed-form and different constructions of computational generalized barycentric coordinates in Chapter 2.

1.3 Applications

In general, the main application of generalized barycentric coordinates is *interpolation*. It follows from (1.8) and (1.9) that the function $\boldsymbol{f} : \bar{\Omega} \to \mathbb{R}^d$ with

$$\boldsymbol{f}(\boldsymbol{p}) = \sum_{i=1}^{n} b_i(\boldsymbol{p})\boldsymbol{f}_i \tag{1.13}$$

reproduces affine functions and from (1.10b) that this function interpolates the data $\boldsymbol{f}_1, \dots, \boldsymbol{f}_n \in \mathbb{R}^d$ at the vertices $\boldsymbol{v}_1, \dots, \boldsymbol{v}_n$. This *barycentric interpolant* $\boldsymbol{f}$ is also linear along the facets of P if the coordinates b_i satisfy the property (1.10c),

lies in the convex hull of the data $\boldsymbol{f}_i$ if the coordinates are non-negative, and is as smooth as the coordinates. Throughout this dissertation, we use *colour interpolation* [Meyer et al., 2002] and *image deformation* [Hormann and Floater, 2006; Warren et al., 2007] (see also Sections 2.3.1 and 2.3.2 for more details) as two example applications of the interpolant $\boldsymbol{f}$ to compare different generalized barycentric coordinates.

Except for colour interpolation and image deformation, generalized barycentric coordinates are also used in a variety of other applications in geometry processing, computer graphics, and computational mechanics. To name a few in the context of geometry processing and computer graphics, they can be used for geometric modelling [Loop and DeRose, 1989], mesh parameterization [Floater, 1997], rendering quadrilaterals [Hormann and Tarini, 2004], shape deformation [Ju, Schaefer and Warren, 2005], polygon shading [Hormann and Floater, 2006], deformation transfer [Ben-Chen et al., 2009], image compositing [Farbman et al., 2009], texture mapping [Desbrun et al., 2002] and texture synthesis [Takayama et al., 2010], and gradient mesh simplification [Li et al., 2013]. In the context of computational mechanics, they are widely used as basis functions in polygonal finite elements and for solving high-order partial differential equations (see [Sukumar and Malsch, 2006] for more details).

1.4 Contributions

The main focus of this dissertation is on closed-form generalized barycentric coordinates in 2D. As mentioned in the previous section, these coordinates can be applied in many different contexts due to their particular properties. If we consider interpolation as main application of generalized barycentric coordinates, the non-negativity property (1.10a) ensures that the interpolated result stays in the convex hull of the given data, and the interpolation property itself is guaranteed by the Lagrange property (1.10b). Linearity along edges (see Equation (1.10c)) is important for image deformation, and for smooth results in all applications, the coordinates have to satisfy the smoothness property (1.10d). In some applications, as we will show later, we also need to have coordinate functions, which are locally supported (see Chapter 5 for more details), but since this property is not usually required for other applications, we do not list it above. Finally, the coordinate functions have to be well-defined for any simple polygon, where by a simple polygon we mean any convex or concave polygon without self-intersections, and they need to have a simple analytic expression, so that we could compute them easily. Hence, the main goal of this dissertation is to

study different properties of already existing coordinate functions and present new constructions that compare favourably with other generalized barycentric coordinates, both in terms of quality and computational cost.

In Chapter 2, we start from discussing the most-known constructions of barycentric coordinates with respect to polygons and polyhedra, where we also present an extensive comparison of 2D coordinates. From this comparison it becomes clear that none of existing constructions leads to coordinates, which comply with all necessary properties of generalized barycentric coordinates. For example, all non-negative coordinates either do not have a simple closed form or are not smooth functions. It is then important to construct new coordinate functions that satisfy all properties from Section 1.2, and we present such constructions in Chapters 5 and 6. This chapter, as well as the beginning of the introductory chapter, are based on:

Anisimov, D. [2017]. *Generalized Barycentric Coordinates in Computer Graphics and Computational Mechanics*, Chapter 1, *in* K. Hormann and N. Sukumar (eds), to appear, CRC Press, Florida.

After studying all existing coordinate functions in Chapter 2, we realized that there are coordinates, whose properties are not fully investigated, and, in Chapter 3, we perform theoretical analysis of a particular one-parameter family of generalized barycentric coordinates for strictly convex polygons. This family of coordinates includes three well-known coordinate functions, that are used in many different applications and posses many nice properties. Hence it is also important to understand which properties of generalized barycentric coordinates are satisfied by other members of this family. In particular, we managed to show that all members of this family are continuous generalized barycentric coordinates in the closure of any strictly convex polygon. This result is based on the following publication:

Anisimov, D., Hormann, K., and Schneider, T. [2017]. Behaviour of exponential three-point coordinates at the vertices of convex polygons, to appear.

After the theoretical analysis of the family above, we also perform its numerical analysis and show that while in theory these coordinates are well-defined everywhere inside the polygon, in practice, the computation is not stable in the vicinity of the polygon's boundary. In this context, we show how to avoid such instabilities when computing these coordinates and implement some important members of the family in the Computational Geometry Algorithm Library [CGAL, 2016]. These results are based on the following publication:

Anisimov, D., Bommes, D., Hormann, K., and Alliez, P. [2016]. 2D generalized barycentric coordinates. In CGAL User and Reference Manual. CGAL Editorial Board, 4.9 Edition.

When we analyzed some properties of already existing coordinate functions, we tried to construct new coordinates with favourable properties. Our first attempt, which becomes a motivation for our main results in Chapters 5 and 6, is based on the idea of a convex combination of two members from the one-parameter family of coordinates above. By taking such a convex combination, we construct new coordinates with some improved properties that we discuss in Chapter 4, which is based on the following poster:

Anisimov, D. [2012]. Blended barycentric coordinates. Poster in NSF Workshop on Barycentric Coordinates in Geometry Processing and Finite/Boundary Element Methods, New York, USA.

We further elaborate on the idea of blending barycentric coordinates using convex combinations in Chapter 5 and present, to the best of our knowledge, the first closed-form coordinates for arbitrary simple polygons that satisfy all properties from Section 1.2 and solve the main problem of how to construct non-negative and smooth coordinate functions with a closed form for any simple polygon. This chapter is mainly based on the following publication:

Anisimov, D., Panozzo, D., and Hormann, K. [2017]. Blended barycentric coordinates. Computer Aided Geometric Design, to appear, 2017.

Although our construction in Chapter 5 has all necessary properties of generalized barycentric coordinates, such blended coordinates are not well-defined for polygons with interior points and so, in Chapter 6, we show how to construct non-negative and smooth coordinate functions for any simple polygon with interior points. We construct such coordinates using subdivision and show that new coordinates have many attractive properties in interpolation and image deformation applications. This chapter is based on the following publication:

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

We finally conclude and point out some interesting future directions in Chapter 7 and present pseudocodes for computing some closed-form generalized barycentric coordinates in Appendix A.

Chapter 2

State of the art

To offer a full overview of existing generalized barycentric coordinates in different dimensions, we first present the most important 2D barycentric coordinates (see Section 2.1). We then also describe the most important 3D constructions and their extensions to higher dimensions (see Section 2.2) and conclude the chapter with several examples of applying the coordinates to colour interpolation and image deformation (see Section 2.3).

2.1 2D coordinates

We first focus on the 2D case and consider a simple polygon P. Without loss of generality, we assume the vertices $\boldsymbol{v}_i$ of this polygon to be given in anticlockwise direction, and we treat the vertex indices cyclically, that is, $\boldsymbol{v}_{i+kn} = \boldsymbol{v}_i$ for $i \in \{1, \dots, n\}$ and $k \in \mathbb{Z}$. Note that all the coordinates in this section, except for Hermite and complex, are generalized barycentric coordinates in the sense of Definition 1.

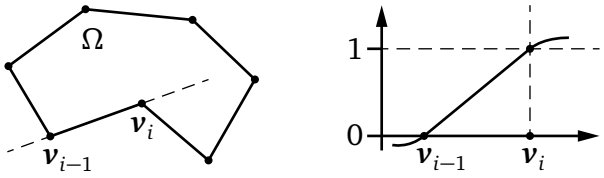


Figure 2.1. Consider the barycentric coordinate function for the concave vertex $\boldsymbol{v}_i$ and the cross section along the line defined by this vertex and the neighbouring vertex $\boldsymbol{v}_{i-1}$ (dashed line). If this function is C^1 at $\boldsymbol{v}_i$, then it must be greater than one in Ω, which implies that another coordinate function is negative.

The earliest generalizations of barycentric coordinates in 2D were closed-form constructions and restricted to convex polygons. In this setting, the key objective is finding smooth and positive weights w_i, which satisfy (1.11). It is then known [Floater et al., 2006] that the normalized weights b_i in (1.12) are well-defined generalized barycentric coordinates with respect to P and satisfy all properties in (1.10). However, most of these constructions lead to negative weights w_i at certain points inside non-convex polygons. Even worse, the denominator W may vanish, so that the b_i are not necessarily well-defined for all $\boldsymbol{p} \in \Omega$. So far, no closed-form construction of positive weights for arbitrary non-convex polygons is known, and even if it exists, the resulting coordinates would not be more than C^0 at concave corners (see Figure 2.1). We discuss different closed-form coordinates in Sections 2.1.1–2.1.7.

It was later realized that b_i can also be obtained numerically as the solution of an optimization problem subject to the constraints given by the required properties of the coordinates. This optimization problem can either be global or local, and we present different kinds of such computational coordinates in Sections 2.1.8–2.1.10.

If the points $\boldsymbol{v}_i$ are not given as vertices of a polygon, but rather as scattered points $\boldsymbol{p}_i$, Definition 1 and properties (1.10b) and (1.10d) still make sense. In this setting, we can define generalized barycentric coordinates b_i with respect to the set $\Pi = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\} \in \mathbb{R}^2$ of $n \geq 3$ scattered points, and we review three different constructions in Sections 2.1.11–2.1.13.

We also give a short overview of coordinates that generalize Definition 1 in some other way. In particular, we briefly discuss Hermite and complex coordinates in Sections 2.1.14 and 2.1.15.

2.1.1 Wachspress coordinates

Wachspress [1975] was one of the first who suggested a generalization of barycentric coordinates to a polygon with $n > 3$ vertices. Later, Meyer et al. [2002] pro-

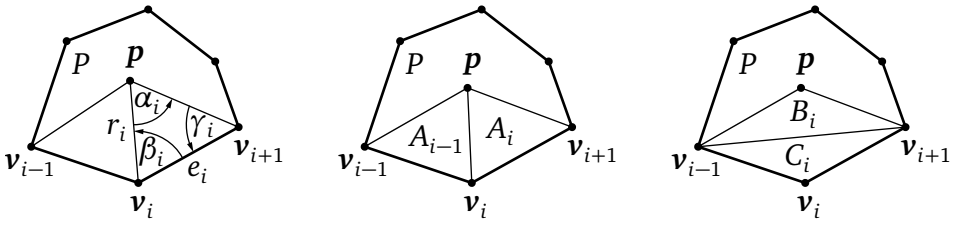


Figure 2.2. Notation used for signed angles, distances, and signed areas in a polygon P.

Figure 2.3. Zero level set (dashed) of the denominator W for the Wachspress (left) and the discrete harmonic (right) coordinates with respect to a concave polygon. In both cases, the zero level curves cross the polygon's interior and hence the coordinates are not well-defined at the corresponding points.

pose a simple local formulation of these *Wachspress coordinates*, which is given by the normalization (1.12) of the weight functions

$$w_i = \frac{\cot \gamma_{i-1} + \cot \beta_i}{r_i^2}, \qquad i = 1, \dots, n, \tag{2.1}$$

where γ_{i-1} and β_i are the angles shown in Figure 2.2 and $r_i = \|\boldsymbol{v}_i - \boldsymbol{p}\|$. These coordinates are rational functions with numerator and denominator of degrees at most $n-2$ and $n-3$, respectively, which are the minimal possible degrees [Warren, 2003].

For strictly convex polygons, it is clear that all $w_i(\boldsymbol{p}) > 0$ for any $\boldsymbol{p} \in \Omega$, so that Wachspress coordinates are well-defined and satisfy all properties in (1.10). They are also affine invariant. For non-convex polygons, the coordinates are not well-defined at some points in the polygon's interior, because the denominator W vanishes (see Figure 2.3), but they can be generalized to weakly convex polygons [Malsch and Dasgupta, 2004].

2.1.2 Discrete harmonic coordinates

Discrete harmonic coordinates [Pinkall and Polthier, 1993; Eck et al., 1995] arise from the standard piecewise linear finite element approximation to the Laplace equation and are given by the weight functions

$$w_i = \cot \beta_{i-1} + \cot \gamma_i, \qquad i = 1, \dots, n, \tag{2.2}$$

where β_{i-1} and γ_i are the angles shown in Figure 2.2. Interestingly, it turns out that for all polygons, whose vertices lie on a common circle, these coordinates are identical to Wachspress coordinates and therefore possess the same properties [Floater et al., 2006]. For other strictly convex polygons, they are still well-defined and satisfy all properties in (1.10), except for non-negativity. For

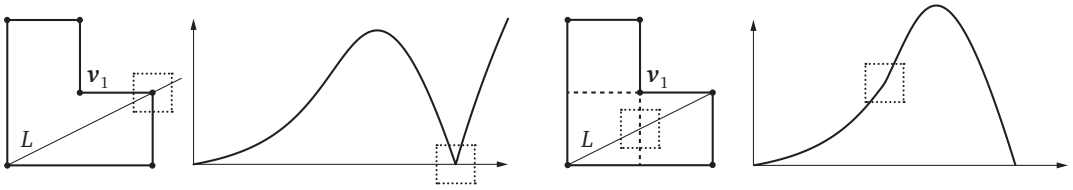


Figure 2.4. Mean value coordinates are only C^0 at the vertices of the polygon (left), while positive mean value coordinates are also C^0 along two dashed lines in the polygon's interior and are not defined outside the polygon (right). Both functions are plotted along the line L with respect to the vertex $\boldsymbol{v}_1$.

non-convex polygons, the denominator W of these coordinates vanishes at some points in the polygon's interior, so that they are not well-defined (see Figure 2.3).

2.1.3 Mean value coordinates

The derivation of discrete harmonic coordinates suggests that different properties of harmonic functions can be exploited to derive other generalized barycentric coordinates. Floater [2003] considers the circumferential mean value property of a harmonic function $u\colon \Omega \to \mathbb{R}$, which states that for any disc $B = B(\boldsymbol{p}, r) \subset \Omega$ of radius $r > 0$, centred at $\boldsymbol{p}$, with boundary ∂B,

$$u(\boldsymbol{p}) = \frac{1}{2\pi r} \int_{\boldsymbol{x} \in \partial B} u(\boldsymbol{x}) d\boldsymbol{x}. \tag{2.3}$$

He then shows that applying (2.3) to a piecewise linear function leads to *mean value coordinates*. These coordinates are given by the weight functions

$$w_i = \frac{\tan(\alpha_{i-1}/2) + \tan(\alpha_i/2)}{r_i}, \qquad i = 1, \ldots, n, \tag{2.4}$$

where α_{i-1} and α_i are the angles shown in Figure 2.2 and $r_i = \|\boldsymbol{v}_i - \boldsymbol{p}\|$.

Hormann and Floater [2006] show that mean value coordinates are well-defined even for sets of nested simple polygons and everywhere in the plane. They are positive inside the kernel of a star-shaped polygon and satisfy properties (1.10b) and (1.10c). Moreover, mean value coordinates are positive inside any quadrilateral, similarity invariant, and smooth, except at the vertices of P, where they are only C^0 (see Figure 2.4, left).

To derive *positive mean value coordinates* for non-convex polygons, Lipman et al. [2007] use the transfinite description of mean value coordinates [Ju, Schaefer and Warren, 2005] and restrict the integration to the part of ∂P that is visible

from $\boldsymbol{p}$. These coordinates can be efficiently evaluated by exploiting the GPU, but they are only C^0 along certain lines inside the polygon (see Figure 2.4, right).

2.1.4 Complete family of coordinates

Floater et al. [2006] show that for arbitrary real functions $c_i : \Omega \to \mathbb{R}$, the weights

$$w_i = \frac{c_{i-1}A_i - c_iB_i + c_{i+1}A_{i-1}}{A_{i-1}A_i}, \qquad i = 1, \dots, n, \tag{2.5}$$

with the signed areas A_i and B_i shown in Figure 2.2, satisfy the property (1.11). Therefore, the task of finding generalized barycentric coordinates simplifies to finding functions c_i such that the weights in (2.5) are positive or, if not, at least sum up to a non-vanishing denominator W, without having to worry about properties (1.8) and (1.9), which are satisfied by construction, and the resulting coordinates are as smooth as the functions c_i. Moreover, any set of generalized barycentric coordinates can be expressed in terms of the weights in (2.5) with the proper choice of c_i.

An interesting special case of this *complete family of coordinates* is given by $c_i = \|\boldsymbol{v}_i - \boldsymbol{p}\|^p$ for any $p \in \mathbb{R}$. In this case, the choices $p = 0$, $p = 1$, and $p = 2$ give Wachspress, mean value, and discrete harmonic coordinates, respectively. Surprisingly, the only coordinates for this choice of c_i, which are non-negative for any convex polygon, are Wachspress and mean value coordinates [Floater et al., 2006]. For any p, the coordinates are similarity invariant, and a simple geometric interpretation can be found in [Langer et al., 2006, Appendix A].

An alternative geometric construction for general weight functions w_i, which satisfy (1.8) and (1.9) and are non-negative by construction, is based on power diagrams [Budninskiy et al., 2016]. However, the resulting *power coordinates* are not necessarily smooth.

2.1.5 Metric coordinates

Malsch et al. [2005] and Sukumar and Malsch [2006] construct the so-called *metric coordinates*, which are given by the weight functions

$$w_i = \frac{A_{i-2}}{C_{i-1}q_{i-2}q_{i-1}} - \frac{B_i}{C_iq_{i-1}q_i} + \frac{A_{i+1}}{C_{i+1}q_iq_{i+1}}, \qquad i = 1, \dots, n,$$

where A_i, B_i, and C_i are the areas shown in Figure 2.2, and

$$q_i = r_i + r_{i+1} - e_i$$

with $r_i = \|\boldsymbol{v}_i - \boldsymbol{p}\|$ and $e_i = \|\boldsymbol{v}_{i+1} - \boldsymbol{v}_i\|$. The functions q_i are non-negative everywhere in the plane, which guarantees a non-vanishing denominator W for all simple polygons [Hormann and Floater, 2006]. Metric coordinates are not necessarily positive inside convex polygons, satisfy properties (1.10b) and (1.10c), and they are smooth, except at the vertices of P, where they are only C^1. These coordinates are not well-defined for polygons with three consecutive collinear vertices.

2.1.6 Poisson coordinates

Taking inspiration from the derivation of mean value coordinates, Li and Hu [2013] derive *Poisson coordinates* from the Poisson integral formula for a harmonic function $u\colon \Omega \to \mathbb{R}$, which states that for any disc $B = B(r) \subset \Omega$ of radius $r > 0$, centred at the origin, with boundary ∂B and any point $\boldsymbol{p} \in B$,

$$u(\boldsymbol{p}) = \frac{1}{2\pi r} \int_{\boldsymbol{x} \in \partial B} \frac{r^2 - \|\boldsymbol{p}\|^2}{\|\boldsymbol{p} - \boldsymbol{x}\|^2} u(\boldsymbol{x}) d\boldsymbol{x}. \tag{2.6}$$

Poisson coordinates extend mean value coordinates, because the circumferential mean value theorem (2.3) is a special case of the formula (2.6) when $\boldsymbol{p}$ is at the centre of B. They are well-defined for any simple polygon P and possess the same properties as mean value coordinates. Moreover, following a specific construction, Poisson coordinates are proved to be pseudo-harmonic, that is, they reproduce harmonic functions on n-dimensional balls (see [Chen and Gotsman, 2016] for more details on pseudo-harmonic coordinates). The closed-form expressions of the Poisson weights w_i are rather lengthy and can be found in [Li and Hu, 2013, Section 4.1].

2.1.7 Gordon–Wixom coordinates

Consider a line L through $\boldsymbol{p} \in \Omega$ along some direction at a given angle θ with respect to a fixed axis, which intersects ∂P at the points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ (see Fig-

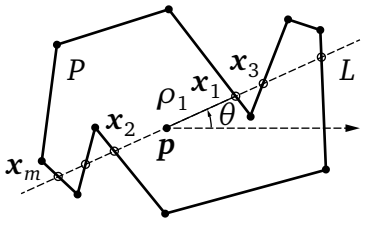


Figure 2.5. Notation used for the Gordon–Wixom interpolant.

ure 2.5). Summing over all m intersections, Belyaev [2006] and Belyaev and Fayolle [2015] extend *Gordon–Wixom coordinates* [Gordon and Wixom, 1974] from convex to arbitrary simple polygons and show that they can be obtained as basis functions from the transfinite interpolant

$$f(\boldsymbol{p}) = \frac{1}{2\pi} \int_0^{2\pi} \left(\sum_{i=1}^{m} \frac{\varepsilon_i f(\boldsymbol{x}_i)}{\rho_i} \Big/ \sum_{i=1}^{m} \frac{\varepsilon_i}{\rho_i} \right) d\theta \tag{2.7}$$

where $f(\boldsymbol{x}_i)$ are data sampled from a piecewise-linear function f given along the polygon's boundary, $\rho_i = \|\boldsymbol{p} - \boldsymbol{x}_i\|$, and $\varepsilon_i \in \{-1, 0, 1\}$ with $\varepsilon_i = -1$ if the ray from $\boldsymbol{p}$ to $\boldsymbol{x}_i$ approaches $\boldsymbol{x}_i$ from the outside of P, $\varepsilon_i = 0$ if it is tangent to ∂P at $\boldsymbol{x}_i$, or $\varepsilon_i = 1$ if it approaches $\boldsymbol{x}_i$ from the inside of P. These coordinates satisfy properties (1.10b) and (1.10c), but they can be negative inside non-convex polygons. Manson et al. [2011] extend (2.7) and present *positive Gordon–Wixom coordinates* inside any simple polygon, which are as smooth as the boundary. The analytic expressions of these coordinates are rather lengthy, and we refer the reader to [Manson et al., 2011, Section 3.2].

2.1.8 Harmonic coordinates

As observed in [Floater et al., 2006], one way of acquiring generalized barycentric coordinates with respect to any simple polygon is by solving the Laplace equation

$$\Delta \boldsymbol{b} = \boldsymbol{0} \tag{2.8}$$

subject to suitable Dirichlet boundary conditions. These boundary conditions are given by a set of piecewise linear functions $\boldsymbol{g} = [g_1, \ldots, g_n] \colon \partial P \to \mathbb{R}^n$ such that every g_i has the Lagrange property (1.10b). These so-called *harmonic coordinates* satisfy all properties in (1.10).

The classical way [Joshi et al., 2007] to approximate these coordinates is by discretizing (2.8) over the space of piecewise linear functions with respect to a triangulation of P. Alternatively, harmonic coordinates can also be approximated using the method of fundamental solutions [Martin et al., 2008], the boundary element method [Rustamov, 2008], and the complex variable boundary method [Weber and Gotsman, 2010].

2.1.9 Maximum entropy coordinates

Sukumar [2004] and Arroyo and Ortiz [2006] show independently that generalized barycentric coordinates can be obtained as the solution of a constrained optimization problem based on the principle of maximum entropy [Jaynes, 1957].

These *maximum entropy coordinates* are non-negative, but the Lagrange property (1.10b) holds only for strictly convex polygons. It was later realized in [Sukumar and Wright, 2007] that, using prior distributions [Kullback and Leibler, 1951; Jaynes, 1963; Shore and Johnson, 1980], the constrained optimization can be modified to be

$$\max_{\boldsymbol{b}(\boldsymbol{p})\in\mathbb{R}_+^n} -\sum_{i=1}^{n} b_i(\boldsymbol{p}) \ln \frac{b_i(\boldsymbol{p})}{\omega_i(\boldsymbol{p})}$$

subject to (1.8) and (1.9), where $\omega_i : \Omega \to \mathbb{R}_+$ is a prior estimate for b_i, which can also be viewed as a weight function.

Hormann and Sukumar [2008] show how to construct ω_i such that maximum entropy coordinates satisfy all properties in (1.10), except smoothness, for any simple polygon. Regarding the smoothness, these coordinates are proven to be C^0-continuous [Sukumar and Wets, 2007] for any set of C^k prior functions with $k \geq 0$ and assumed to be as smooth as the priors. Unlike harmonic coordinates, maximum entropy coordinates do not require solving a global optimization problem, but can rather be evaluated locally at any point $\boldsymbol{p} \in \Omega$, using Newton's method that converges quadratically.

2.1.10 Local coordinates

Zhang et al. [2014] propose to minimize the sum of total variations of the coordinate functions b_i,

$$\min_{\boldsymbol{b}} \sum_{i=1}^{n} \int_{\Omega} \|\nabla b_i\|,$$

subject to the constraints given by (1.8), (1.9), (1.10a), (1.10b), and (1.10c), where the total variation of b_i is the L_1-norm of $\|\nabla b_i\|$ in Ω. The solution of this convex minimization problem gives the so-called *local coordinates*, which are at least C^0. The name stems from the fact that the numerically computed function values are very close to zero in a large part of Ω. The exact support of these coordinates, however, is not known.

2.1.11 Affine coordinates

Among all generalized barycentric coordinates with respect to a scattered set of points Π (see Figure 2.6), Waldron [2011] suggests to consider those with minimal L_2-norm, which are uniquely defined as the *affine* functions

$$b_i(\boldsymbol{p}) = (\boldsymbol{p}-\boldsymbol{c}) \cdot ((MM^*)^{-1}(\boldsymbol{p}_i-\boldsymbol{c}))^T + \frac{1}{n}, \qquad i = 1,\ldots,n,$$

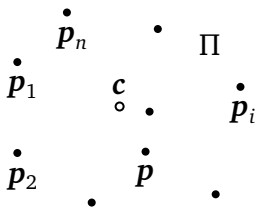


Figure 2.6. Notation used for affine coordinates.

where $\boldsymbol{c} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{p}_i$ is the barycentre of Π, $M = (\boldsymbol{p}_1 - \boldsymbol{c}, \ldots, \boldsymbol{p}_n - \boldsymbol{c})$ is a $2 \times n$ matrix, and M^* is the adjoint of M. These coordinates are non-negative inside a convex region that contains $\boldsymbol{c}$, and they are equal to $\frac{1}{n}$ at $\boldsymbol{c}$. They are smooth and well-defined everywhere in the plane, but, in general, the Lagrange property (1.10b) holds only in the case $n = 3$ (see Section 1.1).

2.1.12 Sibson coordinates

Let V be the Voronoi diagram [Voronoi, 1908] of the set Π and C_i be the Voronoi cell in V that contains $\boldsymbol{p}_i$. Further let $V_{\boldsymbol{p}}$ be the Voronoi diagram of $\Pi \cup \{\boldsymbol{p}\}$ and $C_{\boldsymbol{p}}$ be the Voronoi cell in $V_{\boldsymbol{p}}$ that contains $\boldsymbol{p}$ (see Figure 2.7). Intersecting the cells C_i with $C_{\boldsymbol{p}}$, we can define the weight functions

$$w_i = \text{Area}[C_i \cap C_{\boldsymbol{p}}], \qquad i = 1, \ldots, n. \tag{2.9}$$

These areas were originally proposed by Sibson in [Sibson, 1980, 1981] with the emphasis on natural neighbour interpolation, where the natural neighbours of some $\boldsymbol{p}$ are all points from Π for which $w_i(\boldsymbol{p}) \neq 0$.

Normalizing the weights in (2.9) as in (1.12) defines the *Sibson coordinates*. These coordinates are well-defined over the convex hull of Π, have local support, and satisfy the Lagrange property (1.10b). They are C^1, except at $\boldsymbol{p}_i$, where they

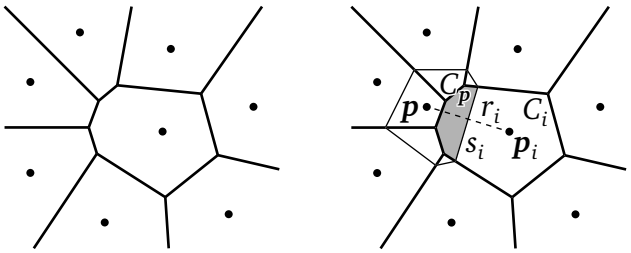


Figure 2.7. Voronoi diagram of a set of scattered points (left) and notation used for the construction of Sibson and Laplace coordinates (right).

are only C^0. A simple computational algorithm for Sibson coordinates can be found in [Sukumar, 2003] and further details in [Sukumar et al., 1998].

2.1.13 Laplace coordinates

The concept of natural neighbours (see Section 2.1.12) is also used for the definition of so-called *Laplace coordinates* [Christ et al., 1982; Belikov et al., 1997; Sugihara, 1999], which are given by the weight functions

$$w_i = \begin{cases} \frac{s_i}{r_i}, & \text{if } i \in I_p, \\ 0, & \text{otherwise,} \end{cases} \qquad i = 1, \dots, n,$$

where s_i is the length of the edge of C_p that is contained in C_i, $r_i = \|\boldsymbol{p}_i - \boldsymbol{p}\|$, and $I_p \subset \{1, \dots, n\}$ is the subset of indices of the natural neighbours of $\boldsymbol{p}$ (see Figure 2.7).

These coordinates have the same properties as Sibson coordinates, but they are only C^0 along the boundary of their support. Interestingly, for polygons, whose vertices lie on a circle, Sibson, Laplace, Wachspress, and discrete harmonic coordinates are all identical [Sukumar and Malsch, 2006]. A simple computational algorithm for Laplace coordinates can be found in [Sukumar, 2003] and further details in [Sukumar et al., 2001].

2.1.14 Hermite coordinates

To interpolate not only boundary data, but also derivative information along the boundary, we can complement the functions b_i with a second set of n functions d_i and substitute the Lagrange property (1.10b) with the *Hermite property*

$$\begin{aligned} b_i(\boldsymbol{v}_j) &= \delta_{ij}, & d_i(\boldsymbol{p}) &= 0 & \forall \boldsymbol{p} &\in \partial P, \\ \frac{\partial b_i}{\partial \boldsymbol{\eta}}(\boldsymbol{p}) &= 0 \qquad \forall \boldsymbol{p} \in \partial P, & \frac{\partial d_i}{\partial \boldsymbol{\eta}}(\boldsymbol{p}) &= \delta_{ij} & \forall \boldsymbol{p} &\in [\boldsymbol{v}_j, \boldsymbol{v}_{j+1}], \end{aligned} \tag{2.10}$$

where $\boldsymbol{\eta}$ is the unit vector normal to the boundary ∂P. In this setting, the key properties of generalized barycentric coordinates in Definition 1 take on the form

$$\sum_{i=1}^{n} b_i(\boldsymbol{p}) + \sum_{i=1}^{n} d_i(\boldsymbol{p}) = 1 \qquad \forall \boldsymbol{p} \in \bar{\Omega}, \tag{2.11}$$

$$\sum_{i=1}^{n} b_i(\boldsymbol{p}) \boldsymbol{v}_i + \sum_{i=1}^{n} d_i(\boldsymbol{p}) \boldsymbol{\eta}_i = \boldsymbol{p} \qquad \forall \boldsymbol{p} \in \bar{\Omega}, \tag{2.12}$$

where $\boldsymbol{\eta}_i$ is the unit normal of the edge $[\boldsymbol{v}_i, \boldsymbol{v}_{i+1}]$. Note that (2.11) is essentially the same as (1.8), because $d_i(\boldsymbol{p}) = 0$.

To the best of our knowledge, Lipman et al. [2008] were the first to derive coordinates b_i and d_i that satisfy (2.11) and (2.12) from Green's third identity. These *Green coordinates* have a closed form, but do not satisfy (2.10). By fitting a bivariate polynomial in a weighted least squares sense to values and derivatives given along the polygon's boundary, Manson and Schaefer [2010] construct *moving least squares coordinates*, which are defined for arbitrary, even disconnected polygons and have a closed form. Another closed-form solution for Hermite coordinates with property (2.10) for arbitrary simple polygons is presented by Li et al. [2013], who derive *cubic mean value coordinates* from the mean value property of biharmonic functions. Weber et al. [2012] extend harmonic coordinates to *biharmonic coordinates*, which satisfy (2.10), but do not posses a closed form.

2.1.15 Complex coordinates

If we interpret P as a subset of $\mathbb{C}$ and view its vertices $\boldsymbol{v}_i = (x_i, y_i)$ as complex numbers $z_i = x_i + \mathrm{i}\, y_i$, then we can reformulate generalized barycentric coordinates as complex functions [Weber et al., 2009, 2011]. These *complex coordinates* are very useful in the context of planar shape deformation, for example, for image deformation (see Section 2.3.2). The complex setting also reveals that Green coordinates (see Section 2.1.14) can alternatively be derived from Cauchy's integral formula [Weber et al., 2009].

2.1.16 Comparison

To compare the generalized barycentric coordinates from this section, except for Hermite and complex, we summarize some of their properties in Table 2.1 and show contour plots of some coordinate functions for a convex as well as a concave polygon in Figures 2.8–2.11.

In particular, Table 2.1 lists the domain for which the coordinates are well-defined and whether or not they have a closed-form definition, are non-negative (1.10a), and satisfy the Lagrange property (1.10b). In addition, the table shows how smooth the coordinates are over the interior of the respective domain. The partition of unity (1.8) and linear reproduction (1.9) properties are not included in the table, because all the coordinates comply with them. Moreover, we do not include the extra property (1.10c) since it is satisfied by all the coordinates defined for polygons and does not apply to coordinates for scattered points. Finally, 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 $\boldsymbol{p} \in \Omega$ is at least $O(n)$.

For a better comparison we group all coordinates by the first two properties. The first group includes Wachspress (WP), discrete harmonic (DH), and the complete family (CF) of coordinates. These coordinates are well-defined for convex polygons only and have a closed form. The second group consists of mean value (MV), positive mean value (PM), metric (MT), Poisson (PS), Gordon–Wixom (GW), and positive Gordon–Wixom (PG) coordinates that are well-defined for arbitrary simple polygons and have a closed form. In the third group, we list harmonic (HM), maximum entropy (ME), and local (LC) coordinates that are also well-defined for arbitrary simple polygons, but can be obtained only numerically. The last group includes affine (AF), Sibson (SB), and Laplace (LP) coordinates that are defined with respect to sets of scattered points.

Figures 2.8 and 2.9 provide a visual comparison of all coordinates by showing contour plots of two different basis functions for a convex polygon. A similar comparison of all coordinates that are well-defined for a concave polygon is given in Figures 2.10 and 2.11. In all plots, the thick curves represent contours for the function values $0.0, 0.1, \ldots, 1.0$ and allow for a comparison of the general shapes of the coordinate functions. In addition, the decay towards zero can be deduced from the thin curves, which represent contours for the values $0.01, 0.02, \ldots, 0.09$. The same contour spacing is used for negative function values, and the corresponding regions are shaded in light grey. Likewise, regions with function values greater than one are shaded in dark grey. As a representative of the complete family (CF) of coordinates, we choose the function

$$c_i = 2r_i \frac{\sin\alpha_{i-1} + \sin\alpha_i}{\sin\alpha_{i-1} + \sin\alpha_i + \sin(\alpha_{i-1} + \alpha_i)},$$

which was proposed in [Floater et al., 2006, Section 5]. For affine, Sibson, and Laplace coordinates we treat the vertices of the polygon as a set of scattered points and restrict the plot to the convex hull. The plots show that mean value, Poisson, and Gordon–Wixom coordinates can become negative inside nonconvex polygons. The same is true for metric coordinates, also for convex polygons, and for affine coordinates, which do not even necessarily satisfy the Lagrange property.

Table 2.1. Properties of 2D generalized barycentric coordinates.

Coordinates (Section)		Valid domain	Closed form	Non-negativity	Lagrange property	Smoothness
WP	(2.1.1)		✓	✓	✓	C^∞
DH	(2.1.2)		✓		✓	C^∞
CF	(2.1.4)		✓		✓	C^∞
MV	(2.1.3)		✓		✓	C^∞
PM	(2.1.3)		✓	✓	✓	C^0
MT	(2.1.5)		✓		✓	C^∞
PS	(2.1.6)		✓		✓	C^∞
GW	(2.1.7)		✓		✓	C^0
PG	(2.1.7)		✓	✓	✓	C^0
HM	(2.1.8)			✓	✓	C^∞
ME	(2.1.9)			✓	✓	C^k
LC	(2.1.10)			✓	✓	C^0
AF	(2.1.11)		✓			C^∞
SB	(2.1.12)		✓	✓	✓	C^1
LP	(2.1.13)		✓	✓	✓	C^0

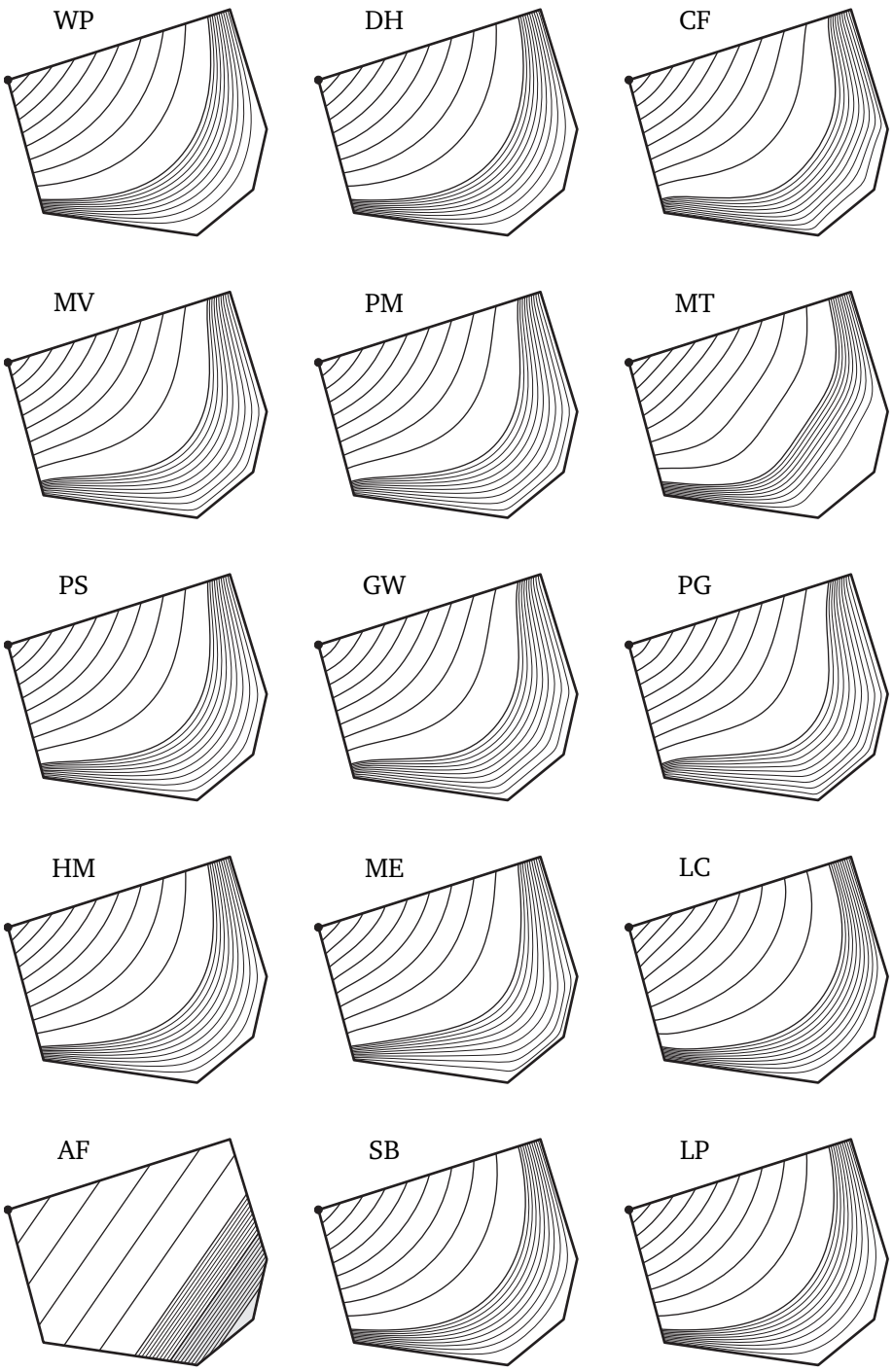


Figure 2.8. 2D generalized barycentric coordinates for a convex polygon with respect to the marked vertex.

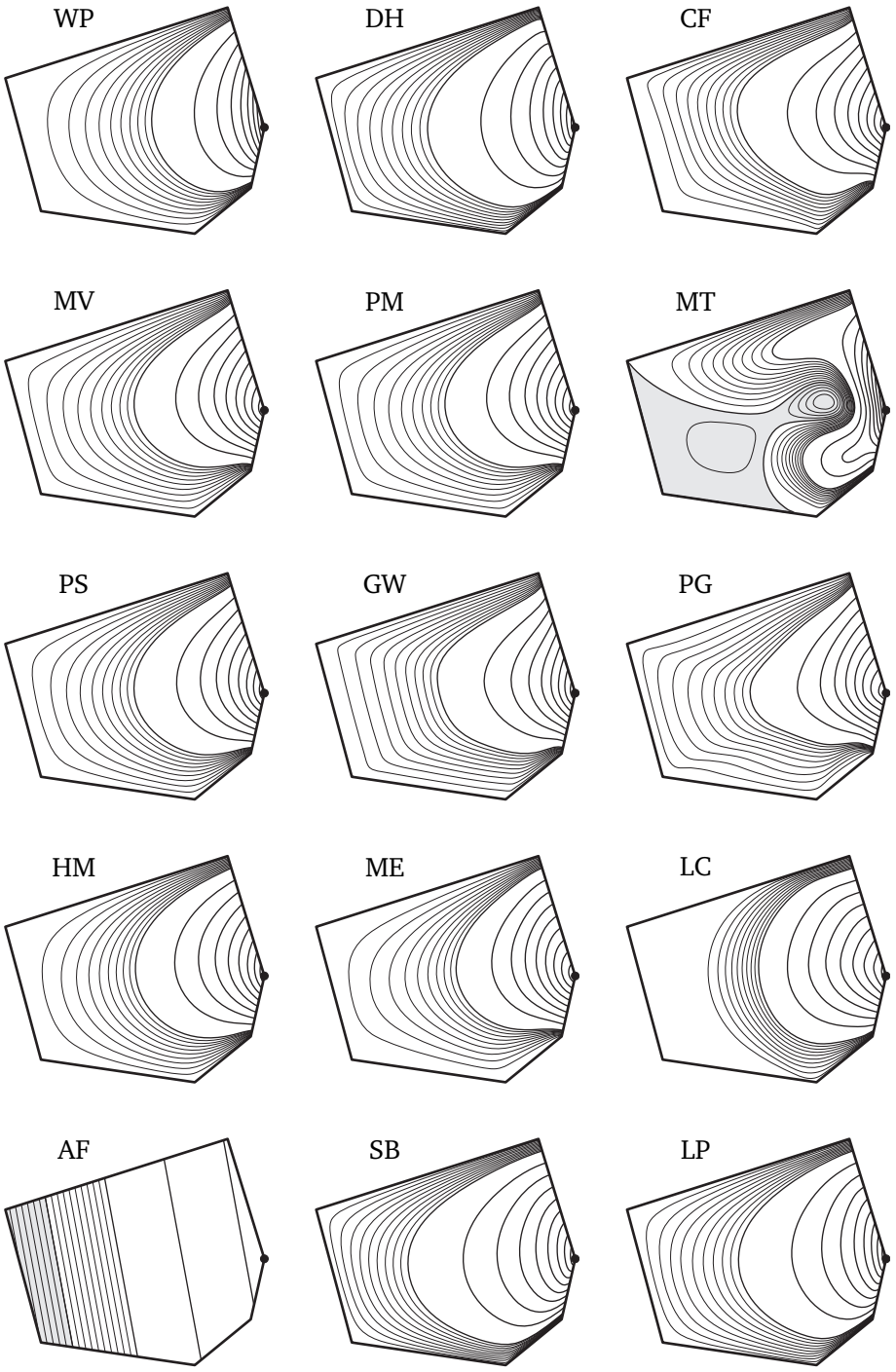


Figure 2.9. 2D generalized barycentric coordinates for a convex polygon with respect to the marked vertex.

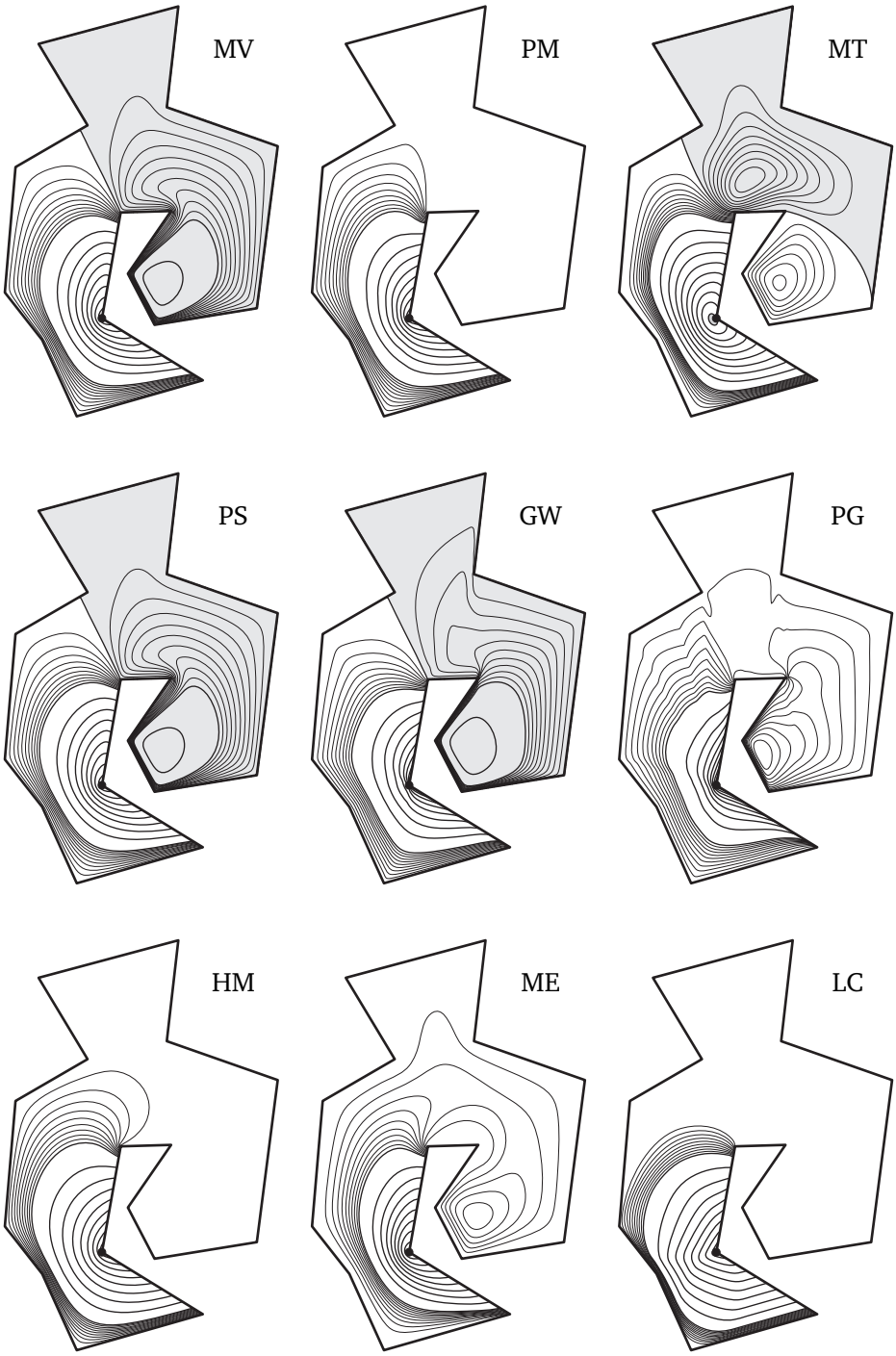


Figure 2.10. 2D generalized barycentric coordinates for a concave polygon with respect to the marked vertex.

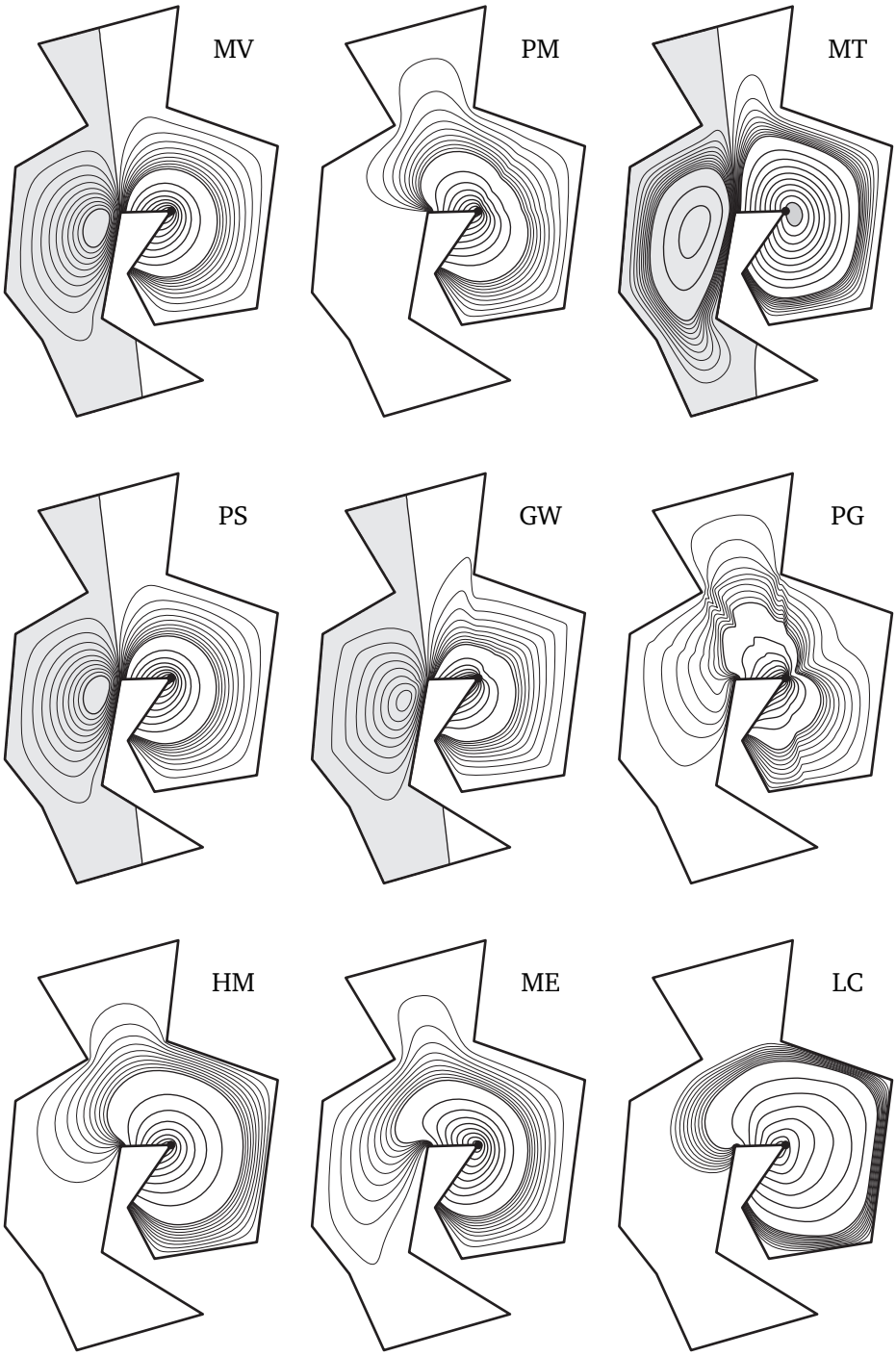


Figure 2.11. 2D generalized barycentric coordinates for a concave polygon with respect to the marked vertex.

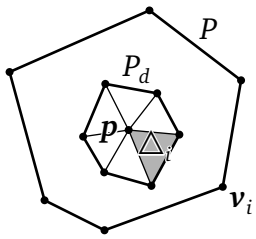


Figure 2.12. Polar dual P_d of a hexagon P centred at $\boldsymbol{p}$, where the triangle $\triangle_i$ is dual to the vertex $\boldsymbol{v}_i$.

2.2 3D coordinates

We now focus on the 3D case and consider a polyhedron P with n vertices $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ and m faces $F_1, \ldots, F_m$. We present explicit formulations for some generalized barycentric coordinates in this setting, which are direct extensions of the corresponding 2D constructions. Since the properties of these coordinates carry over from 2D to 3D, we do not mention them explicitly.

2.2.1 Wachspress coordinates

Wachspress coordinates can be generalized to convex polyhedra in which all vertices have exactly three incident faces [Wachspress, 1975; Warren, 1996; Warren et al., 2007], and to arbitrary convex polyhedra using the concept of polar duals [Ju, Schaefer, Warren and Desbrun, 2005]. The polar dual of a bounded convex polyhedron that contains the origin is itself a convex polyhedron of the form $P_d = \{\boldsymbol{x} \in \mathbb{R}^3 \mid \boldsymbol{x} \cdot \boldsymbol{y} \leq 1 \ \forall \boldsymbol{y} \in P\}$. Since each vertex $\boldsymbol{v}_i$ of P has a dual polygonal pyramid $\triangle_i$ in P_d (see Figure 2.12 for a 2D example), translating P such that $\boldsymbol{p}$ is at the origin and letting

$$w_i = \mathrm{Vol}[\triangle_i], \qquad i = 1, \ldots, n$$

leads to *3D Wachspress coordinates* after the normalization in (1.12). Note that the constructions in [Warren, 1996; Warren et al., 2007] also extend to higher dimensions.

2.2.2 Discrete harmonic coordinates

It was first mentioned in [Meyer et al., 2003] that *3D discrete harmonic coordinates* exist, but no exact formula was given. Later, Ju et al. [2007] show that for

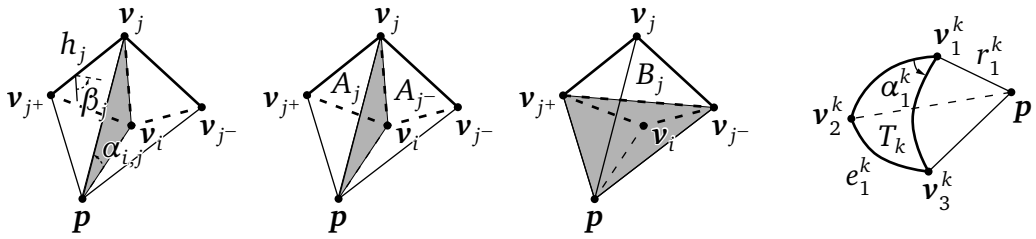


Figure 2.13. Notation used for signed angles, distances, and signed volumes in a polyhedron with triangular faces.

convex polyhedra with triangular faces these coordinates are given by the weight functions

$$w_i = \sum_{j \in N_i} h_j \cot \beta_j, \qquad i = 1, \dots, n,$$

where $N_i \subset \{1, \dots, n\}$ is the subset of the indices of the vertices in the one-ring neighbourhood of $\boldsymbol{v}_i$, β_j is the dihedral angle between the faces $[\boldsymbol{v}_i, \boldsymbol{v}_j, \boldsymbol{v}_{j^+}]$ and $[\boldsymbol{p}, \boldsymbol{v}_j, \boldsymbol{v}_{j^+}]$, and $h_j = \left\| \boldsymbol{v}_j - \boldsymbol{v}_{j^+} \right\|$ (see Figure 2.13).

Interestingly, unlike their 2D counterparts (see Section 2.1.2), discrete harmonic and Wachspress coordinates are not identical for polyhedra, whose vertices lie on a common sphere. However, [Ju et al., 2007, Section 3.2.2] introduce *Voronoi coordinates*, which are identical to Wachspress coordinates on such polyhedra.

2.2.3 Mean value coordinates

Floater et al. [2005] and Ju, Schaefer and Warren [2005] independently generalize mean value coordinates to arbitrary polyhedra with triangular faces. Consider a face $F_k = [\boldsymbol{v}_1^k, \boldsymbol{v}_2^k, \boldsymbol{v}_3^k]$ of P. By projecting F_k onto the unit sphere centred at $\boldsymbol{p}$ we obtain a triangular wedge T_k. The weight functions for *3D mean value coordinates* with respect to the vertices $\boldsymbol{v}_j^k$ of the face F_k are then defined as

$$w_j^k = \frac{e_j^k - e_{j-1}^k \cos \alpha_{j+1}^k - e_{j+1}^k \cos \alpha_{j-1}^k}{r_j^k \sin \alpha_{j+1}^k \sin \alpha_{j-1}^k}, \qquad j = 1, \dots, 3, \quad k = 1, \dots, m,$$

with the angles α_j^k and the spherical edge lengths e_j^k as shown in Figure 2.13 and $r_j^k = \left\| \boldsymbol{v}_j^k - \boldsymbol{p} \right\|$. Details regarding the generalization of this construction to polyhedra with arbitrary faces can be found in [Floater, 2015].

2.2.4 Complete family of coordinates

Ju et al. [2007] show that for convex polyhedra with triangular faces, Wachspress, discrete harmonic, and mean value coordinates can all be unified in a simple framework, which closely resembles the one in Section 2.1.4. This *complete family of 3D coordinates* is given by the weight functions

$$w_i = \sum_{j \in N_i} \frac{c_{j,j^+}}{A_j} + \sum_{j \in N_i} \frac{c_{i,j} B_j}{A_j A_{j^-}}, \qquad i = 1, \dots, n,$$

where $c_{i,j} : \Omega \to \mathbb{R}$ are arbitrary real functions, A_j and B_j are the signed volumes shown in Figure 2.13, and $N_i \subset \{1, \dots, n\}$ is the subset of the indices of the vertices in the one-ring neighbourhood of $\boldsymbol{v}_i$. The name stems from the fact that this framework includes all possible coordinates for polyhedra with triangular faces. For example, given the signed angle $\alpha_{i,j}$ formed by $\boldsymbol{v}_i - \boldsymbol{p}$ and $\boldsymbol{v}_j - \boldsymbol{p}$, the choices

$$\begin{aligned} c_{i,j} &= 2 - \left(\frac{\|\boldsymbol{v}_i - \boldsymbol{p}\|}{\|\boldsymbol{v}_j - \boldsymbol{p}\|} + \frac{\|\boldsymbol{v}_j - \boldsymbol{p}\|}{\|\boldsymbol{v}_i - \boldsymbol{p}\|} \right) \cdot \cos \alpha_{i,j}, \\ c_{i,j} &= \left\| (\boldsymbol{v}_i - \boldsymbol{p}) \times (\boldsymbol{v}_j - \boldsymbol{p}) \right\|^2, \\ c_{i,j} &= \left\| (\boldsymbol{v}_i - \boldsymbol{p}) \times (\boldsymbol{v}_j - \boldsymbol{p}) \right\| \cdot \alpha_{i,j} \end{aligned}$$

lead to Wachspress, discrete harmonic, and mean value coordinates, respectively.

Alternatively, all non-negative coordinates for convex polyhedra can be represented as *3D power coordinates* [Budninskiy et al., 2016], and both approaches can be extended to convex polytopes in higher dimensions.

2.2.5 Other coordinates

The extensions of positive mean value [Lipman et al., 2007], harmonic [Joshi et al., 2007], maximum entropy [Hormann and Sukumar, 2008] and local [Zhang et al., 2014] coordinates to 3D are straightforward and mentioned in the respective papers. Extensions of Sibson and Laplace coordinates to arbitrary higher dimensions are discussed in [Bobach et al., 2006] and to higher continuity in [Hiyoshi and Sugihara, 2000]. The other 2D coordinates in Section 2.1 can probably be extended to 3D, too, but to the best of our knowledge, this has not been done so far. Apart from Wachspress coordinates, the only other coordinates that have been explicitly extended to arbitrary higher dimension are affine coordinates [Waldron, 2011].

2.3 Applications

As we explained in Section 1.3, the main application of generalized barycentric coordinates is interpolation. We now discuss colour interpolation (see Section 2.3.1) and image deformation (see Section 2.3.2) as two example applications of the barycentric interpolant $\boldsymbol{f}$ in (1.13). Since, the majority of the coordinate functions in Section 2.1 are not usually applied to these applications, from now on we compare colour interpolation and image deformation based only on mean value, metric, harmonic, maximum entropy, and local coordinates.

2.3.1 Colour interpolation

Given a polygon P with RGB colour data $\boldsymbol{f}_i \in [0,1]^3$ specified at the vertices $\boldsymbol{v}_i$ of P, we can propagate these colours to the interior of P by applying (1.13). If the coordinates b_i are not bounded between 0 and 1, then the interpolated RGB values can happen to be outside the valid range [0, 1] and we truncate these values to 0 or 1, respectively. Due to this truncation, the interpolation function $\boldsymbol{f}$ is only C^0 along certain curves inside the polygon, and this can be seen in Figure 2.14, where mean value and metric coordinates posses the truncation artefacts, while harmonic, maximum entropy, and local coordinates do not.

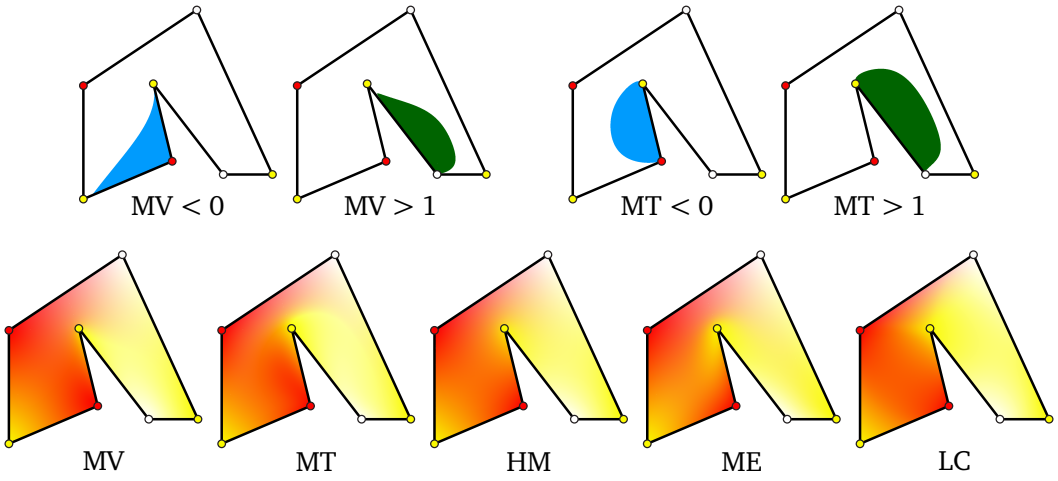


Figure 2.14. Colour interpolation with different generalized barycentric coordinates (bottom row) and regions where the interpolation of the blue and green colour channels has been truncated for values < 0 and > 1, respectively (top row). Note, that in this example, the mean value and metric interpolation of the red channel and the interpolations of all channels with harmonic, maximum entropy, and local coordinates are within the range $[0,1]$.

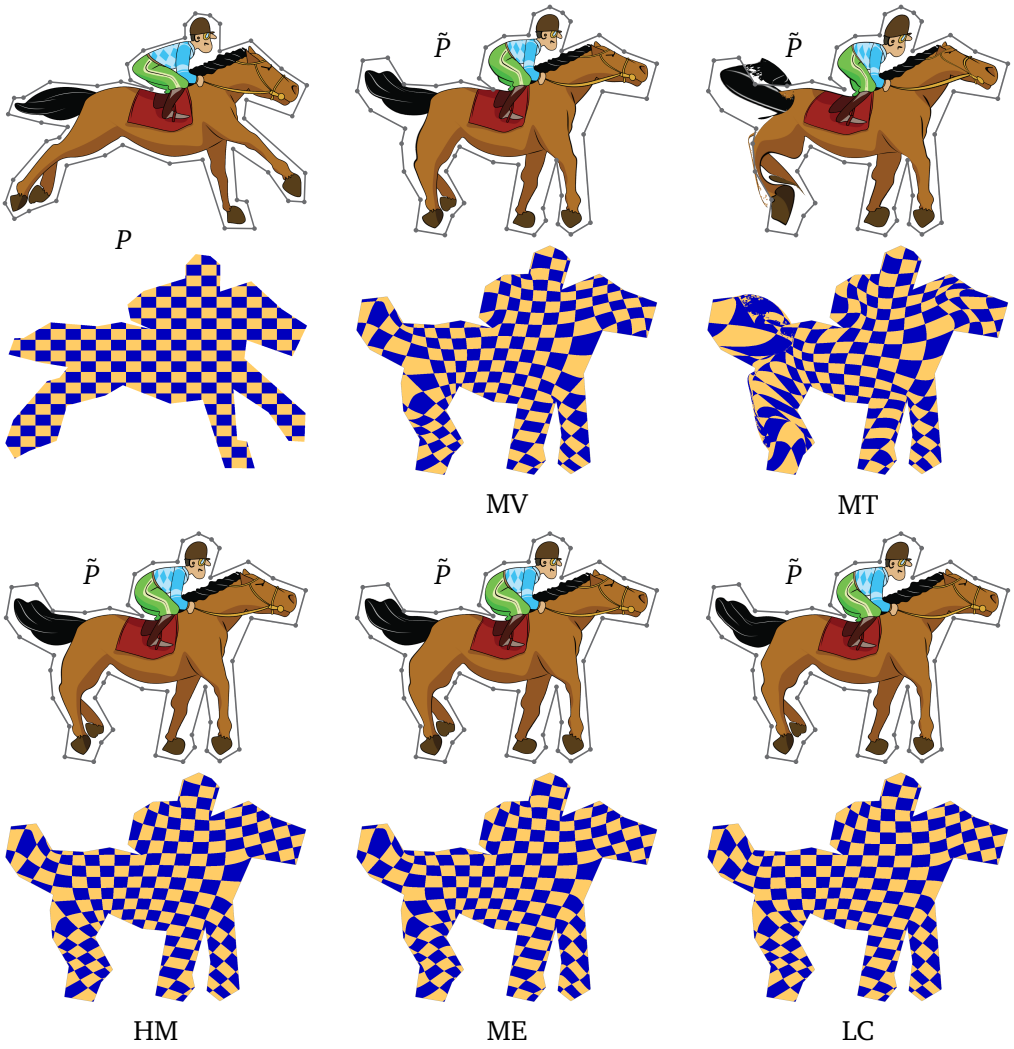


Figure 2.15. Comparison of image deformation with different generalized barycentric coordinates.

2.3.2 Image deformation

We can deform an image that is contained in the source polygon P by specifying a target polygon $\tilde{P}$ with the vertices $\tilde{\boldsymbol{v}}_1, \ldots, \tilde{\boldsymbol{v}}_n$ and using (1.13) with the data $\boldsymbol{f}_i = \tilde{\boldsymbol{v}}_i$, that is

$$\boldsymbol{f}(\boldsymbol{p}) = \sum_{i=1}^{n} b_i(\boldsymbol{p}) \tilde{\boldsymbol{v}}_i. \tag{2.13}$$

Given a source image $I : \Omega \to C$ that maps Ω to some colour space C and a target image $\tilde{I} : \tilde{\Omega} \to C$, the deformed image is then generated by simply setting

$$\tilde{I}(\boldsymbol{f}(\boldsymbol{p})) = I(\boldsymbol{p}).$$

Since it is often necessary in image deformation to preserve straight edges, image deformation based on generalized barycentric coordinates is particularly useful due to property (1.10c). We also want to remark that the deformed image depends smoothly on the target vertex positions, because the coordinate functions b_i are defined with respect to the source polygon, which remains unchanged.

Figure 2.15 gives a visual comparison of the deformed images using different generalized barycentric coordinates. While mean value, harmonic, maximum entropy, and local coordinates lead to similar results, the deformation with metric coordinates suffers from high distortions and is visually inappropriate for this example. We also notice that the locality of local coordinates guarantees the mapping to be isometric and leads to a less distorted shape of the horse.

A 3D analogue of image deformation is character articulation. By computing 3D generalized barycentric coordinates $b_i(\boldsymbol{p})$ for all points $\boldsymbol{p}$ of a 3D object with respect to the vertices $\boldsymbol{v}_i$ of the cage $P \subset \mathbb{R}^3$ around this object, the object is deformed with respect to the modified cage $\tilde{P} \subset \mathbb{R}^3$ with the vertices $\tilde{\boldsymbol{v}}_i$ using the interpolant (2.13).

Chapter 3

Exponential three-point coordinates

As discussed in Section 2.1.4, Floater et al. [2006] present a complete family of generalized barycentric coordinates for strictly convex polygons. In particular, for any given set of functions $c_1, \dots, c_n : \Omega \to \mathbb{R}$, let

$$w_i(\boldsymbol{p}) = \frac{c_{i+1}(\boldsymbol{p})A_{i-1}(\boldsymbol{p}) - c_i(\boldsymbol{p})B_i(\boldsymbol{p}) + c_{i-1}(\boldsymbol{p})A_i(\boldsymbol{p})}{A_{i-1}(\boldsymbol{p})A_i(\boldsymbol{p})}, \qquad i = 1, \dots, n, \tag{3.1}$$

where $A_i(\boldsymbol{p})$ and $B_i(\boldsymbol{p})$ are the triangle areas shown in Figure 3.1. The functions b_i are then computed as in (1.12),

$$b_i(\boldsymbol{p}) = \frac{w_i(\boldsymbol{p})}{W(\boldsymbol{p})}, \qquad i = 1, \dots, n \tag{3.2}$$

with

$$W(\boldsymbol{p}) = \sum_{i=1}^{n} w_i(\boldsymbol{p}), \tag{3.3}$$

and they are well-defined and satisfy conditions (1.8) and (1.9) for any $\boldsymbol{p} \in \Omega$, as long as the denominator $W(\boldsymbol{p})$ does not vanish. Moreover, if the w_i in (3.1)

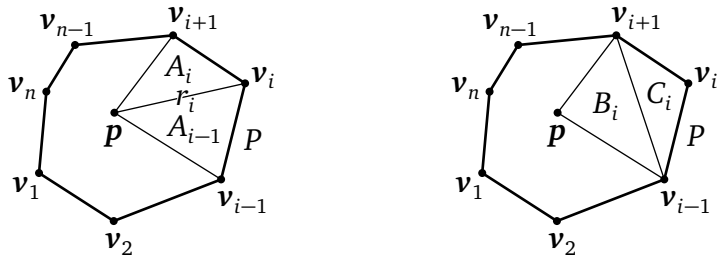


Figure 3.1. Notation used for the definition of exponential three-point coordinates with respect to a strictly convex polygon P.

are non-negative on Ω, then the b_i extend continuously to $\bar{\Omega}$ and satisfy condition (1.10b). However, the non-negativity of the w_i is only a sufficient condition and the construction above usually leads to proper generalized barycentric coordinates even if it is not satisfied.

Floater et al. [2006] further study a particular one-parameter family of coordinates that we call *exponential three-point coordinates*. These coordinates are defined by setting $c_i(\boldsymbol{p}) = r_i(\boldsymbol{p})^p$ in (3.1) for some $p \in \mathbb{R}$ and $r_i(\boldsymbol{p}) = \|\boldsymbol{v}_i - \boldsymbol{p}\|$ (see Figure 3.1). It also turns out that Wachspress (Section 2.1.1), mean value (Section 2.1.3), and discrete harmonic coordinates (Section 2.1.2) are special members of this family for $p = 0$, $p = 1$, and $p = 2$, respectively, and that $p = 0$ and $p = 1$ are the only choices of p for which the w_i in (3.1) are positive. According to the sufficient condition mentioned above, this implies that Wachspress and mean value coordinates satisfy all three conditions (1.8), (1.9), and (1.10b), but is that true for other values of p, too? In this chapter, we confirm that all exponential three-point coordinates do satisfy these conditions (Section 3.1), analyse some numerical issues that arise when implementing these coordinates (Section 3.2), and present both numerically stable and efficient implementations of Wachspress, mean value, and discrete harmonic coordinates for the Computational Geometry Algorithm Library [CGAL, 2016] (Section 3.3).

3.1 Theoretical aspects

To show that all exponential three-point coordinates satisfy conditions (1.8), (1.9), and (1.10b), let us first observe that the denominator $W(\boldsymbol{p})$ in (3.3) does not vanish for any $\boldsymbol{p} \in \Omega$.

Proposition 1. *Exponential three-point coordinates are well-defined over Ω for any $p \in \mathbb{R}$.*

Proof. Omitting the argument $\boldsymbol{p}$ and noticing that $A_{i-1} + A_i = B_i + C_i$ with $C_i = \text{Area}(\boldsymbol{v}_{i-1}, \boldsymbol{v}_i, \boldsymbol{v}_{i+1})$, as shown in Figure 3.1, we can write W as

$$\begin{aligned} W &= \sum_{i=1}^{n} \frac{r_{i+1}^p A_{i-1} - r_i^p (A_{i-1} + A_i - C_i) + r_{i-1}^p A_i}{A_{i-1} A_i} \\ &= \sum_{i=1}^{n} \frac{r_{i+1}^p - r_i^p}{A_i} + \sum_{i=1}^{n} \frac{r_i^p C_i}{A_{i-1} A_i} - \sum_{i=1}^{n} \frac{r_i^p - r_{i-1}^p}{A_{i-1}} \\ &= \sum_{i=1}^{n} \frac{r_i^p C_i}{A_{i-1} A_i}, \end{aligned} \tag{3.4}$$

which is clearly positive for $\boldsymbol{p} \in \Omega$. Therefore, the b_i in (3.2) do not have any singularities in Ω. □

Next, let us analyse the behaviour of the functions b_i as $\boldsymbol{p}$ approaches any of the open edges $E_i = (\boldsymbol{v}_i, \boldsymbol{v}_{i+1})$, $i = 1, \dots, n$ of P. In this case, the area A_i converges to zero, so that w_i and w_{i+1} diverge to infinity. We can fix this problem by introducing the products

$$\mathcal{A} = \prod_{j=1}^{n} A_j, \qquad \mathcal{A}_i = \prod_{\substack{j=1 \\ j\neq i}}^{n} A_j, \qquad \mathcal{A}_{i-1,i} = \prod_{\substack{j=1 \\ j\neq i-1,i}}^{n} A_j, \qquad i = 1, \dots, n, \tag{3.5}$$

of all areas A_j and those with one or two terms missing, respectively, and considering the functions

$$\tilde{w}_i = w_i \mathcal{A} = r_{i+1}^p \mathcal{A}_i - r_i^p B_i \mathcal{A}_{i-1,i} + r_{i-1}^p \mathcal{A}_{i-1}, \qquad i = 1, \dots, n, \tag{3.6}$$

and

$$\tilde{W} = W\mathcal{A} = \sum_{i=1}^{n} \tilde{w}_i. \tag{3.7}$$

Since $\mathcal{A}$ is well-defined and does not vanish over Ω, it is clear that the functions

$$\tilde{b}_i = \frac{\tilde{w}_i}{\tilde{W}}, \qquad i = 1, \dots, n, \tag{3.8}$$

coincide with the b_i on Ω, but they have the advantage of being well-defined over the open edges of P.

Proposition 2. *Exponential three-point coordinates extend continuously to* $\Omega \cup E_1 \cup \cdots \cup E_n$ *and are linear along* $E_1, \dots, E_n$ *for any* $p \in \mathbb{R}$.

Proof. Let us write $\boldsymbol{p} \in E_j$ as $\boldsymbol{p} = (1-t)\boldsymbol{v}_j + t\boldsymbol{v}_{j+1}$ for some $t \in (0,1)$ and note that $A_j(\boldsymbol{p}) = 0$ and

$$-\frac{B_j(\boldsymbol{p})}{A_{j-1}(\boldsymbol{p})} = \frac{r_{j+1}(\boldsymbol{p})}{r_j(\boldsymbol{p})} = \frac{1-t}{t}.$$

Therefore, by (3.6) and omitting the argument $\boldsymbol{p}$,

$$\tilde{w}_j = r_{j+1}^p \mathcal{A}_j + r_j^{p-1} r_{j+1} \mathcal{A}_j = \left(r_{j+1}^{p-1} + r_j^{p-1}\right) r_{j+1} \mathcal{A}_j$$

and similarly

$$\tilde{w}_{j+1} = \left(r_{j+1}^{p-1} + r_j^{p-1}\right) r_j \mathcal{A}_j.$$

Moreover,

$$\tilde{W} = \tilde{w}_j + \tilde{w}_{j+1} = \big(r_{j+1}^{p-1} + r_j^{p-1}\big)\big(r_j + r_{j+1}\big)\mathcal{A}_j > 0,$$

because $\tilde{w}_k = 0$ for $k \neq j, j+1$, and so

$$\tilde{b}_j = \frac{r_{j+1}}{r_j + r_{j+1}} = 1 - t, \qquad \tilde{b}_{j+1} = \frac{r_j}{r_j + r_{j+1}} = t, \tag{3.9}$$

and $\tilde{b}_k = 0$ for $k \neq j, j+1$. □

The functions $\tilde{b}_i$ are not well-defined at the polygon's vertices $\boldsymbol{v}_j$, except for $p = 0$, but the linear behaviour along the edges E_j in (3.9) implies that $\tilde{b}_i(\boldsymbol{p})$ converges to δ_{ij} as $\boldsymbol{p}$ approaches $\boldsymbol{v}_j$ along ∂P. It turns out that this behaviour also holds for $\boldsymbol{p}$ approaching $\boldsymbol{v}_j$ arbitrarily inside P (see Section 3.1.1), so that a continuous extension of exponential three-point coordinates to $\bar{\Omega}$ is obtained by enforcing the Lagrange property (1.10b). For $p \leq 1$, the coordinates can further be extended to some region around P, but they have unremovable singularities arbitrarily close to the vertices for $p > 1$ (see Section 3.1.2).

3.1.1 Convergence from inside

Let us first consider the case $p < 0$ and analyse the behaviour of the functions $\tilde{b}_i$ as $\boldsymbol{p}$ approaches some vertex $\boldsymbol{v}_j$ of P. In this case, the distance r_j converges to zero, so that r_j^p and at least $\tilde{w}_j$ diverge to infinity. Similar to above, we can fix this problem by introducing the products

$$\mathcal{R} = \prod_{j=1}^{n} r_j^{-p}, \qquad \mathcal{R}_i = \prod_{\substack{j=1 \\ j\neq i}}^{n} r_j^{-p}, \qquad i = 1, \dots, n,$$

and considering the functions

$$\hat{w}_i = \tilde{w}_i \mathcal{R} = \mathcal{R}_{i+1}\mathcal{A}_i - \mathcal{R}_i B_i \mathcal{A}_{i-1,i} + \mathcal{R}_{i-1}\mathcal{A}_{i-1}, \qquad i = 1, \dots, n, \tag{3.10}$$

and

$$\hat{W} = \tilde{W}\mathcal{R} = \sum_{i=1}^{n} \hat{w}_i.$$

Since $\mathcal{R}$ is well-defined and does not vanish over $\Omega \cup E_1 \cup \cdots \cup E_n$, it is clear that the functions

$$\hat{b}_i = \frac{\hat{w}_i}{\hat{W}}, \qquad i = 1, \dots, n,$$

coincide with the $\tilde{b}_i$ on $\Omega \cup E_1 \cup \cdots \cup E_n$, but they have the advantage of being well-defined at the vertices of P.

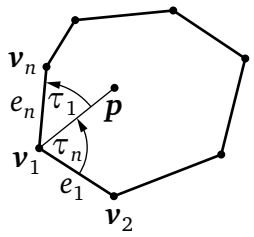


Figure 3.2. Notation for angles and edge lengths used in the proofs of Lemmas 2 and 3.

Lemma 1. *Exponential three-point coordinates extend continuously to $\bar{\Omega}$ for $p < 0$.*

Proof. First observe that $A_{j-1}(\boldsymbol{p})$, $A_j(\boldsymbol{p})$, and $r_j(\boldsymbol{p})$ vanish at $\boldsymbol{p} = \boldsymbol{v}_j$. Therefore, $\mathcal{A}_i = 0$ for all i, $\mathcal{A}_{i-1,i} = \mathcal{R}_i = 0$ for $i \neq j$, and $\mathcal{A}_{j-1,j}, \mathcal{R}_j > 0$, so that all terms of the $\hat{w}_i$ in (3.10) vanish, except for the second term of $\hat{w}_j$. Consequently, $\hat{w}_i = 0$ for $i \neq j$, $\hat{w}_j = -\mathcal{R}_j B_j \mathcal{A}_{j-1,j} > 0$, $\hat{W} = \hat{w}_j > 0$, and finally $\hat{b}_i(\boldsymbol{v}_j) = \delta_{ij}$. □

The reasoning in the proof of Lemma 1 does not carry over to the case $p > 0$, because $\mathcal{R}$ and $\mathcal{R}_i$ for $i \neq j$ diverge to infinity as $\boldsymbol{p}$ approaches $\boldsymbol{v}_j$. However, for $0 < p < 1$, this divergence is counterbalanced by the zero-convergence of the areas A_{j-1} and A_j, so that the $\hat{w}_i$ converge to finite values at $\boldsymbol{p} = \boldsymbol{v}_j$.

Lemma 2. *Exponential three-point coordinates extend continuously to $\bar{\Omega}$ for $0 < p < 1$.*

Proof. Without loss of generality, we consider the case where $\boldsymbol{p}$ approaches $\boldsymbol{v}_1$, so that A_1, A_n, and r_1 converge to zero, while all other A_i and r_i remain strictly positive. The key idea now is to show that the two quotients A_1/r_1^p and A_n/r_1^p converge to zero, too. Denoting the length of E_1 by $e_1 = \|\boldsymbol{v}_2 - \boldsymbol{v}_1\|$ and the signed angle between the vectors $\boldsymbol{v}_2 - \boldsymbol{v}_1$ and $\boldsymbol{p} - \boldsymbol{v}_1$ by τ_1 (see Figure 3.2), we can bound the first quotient as

$$0 \leq \frac{A_1}{r_1^p} = \frac{r_1 e_1 \sin \tau_1}{2 r_1^p} \leq \frac{r_1^{1-p} e_1}{2} \tag{3.11}$$

for any $\boldsymbol{p} \in \Omega$. Since the upper bound is zero at $\boldsymbol{p} = \boldsymbol{v}_1$, we conclude

$$\lim_{\boldsymbol{p} \to \boldsymbol{v}_1} \frac{A_1(\boldsymbol{p})}{r_1(\boldsymbol{p})^p} = 0 \tag{3.12}$$

and similarly

$$\lim_{\boldsymbol{p} \to \boldsymbol{v}_1} \frac{A_n(\boldsymbol{p})}{r_1(\boldsymbol{p})^p} = 0. \tag{3.13}$$

It follows that all terms of the $\hat{w}_i$ in (3.10) with a diverging factor $\mathcal{R}_i$, $i \neq 1$, converge to zero, because they contain one of these two quotients. Among the other three terms with factor $\mathcal{R}_1$, which is finite at $\boldsymbol{p} = \boldsymbol{v}_1$, the terms in $\hat{w}_2$ and $\hat{w}_n$ are zero, because $\mathcal{A}_1$ and $\mathcal{A}_n$ vanish, so that only the second term in $\hat{w}_1$ is non-zero. Consequently, $\lim_{\boldsymbol{p}\to\boldsymbol{v}_1} \hat{w}_i(\boldsymbol{p}) = 0$ for $i \neq 1$, $\lim_{\boldsymbol{p}\to\boldsymbol{v}_1} \hat{w}_1(\boldsymbol{p}) = -\mathcal{R}_1(\boldsymbol{v}_1)B_1(\boldsymbol{v}_1)\mathcal{A}_{n,1}(\boldsymbol{v}_1) > 0$, $\lim_{\boldsymbol{p}\to\boldsymbol{v}_1} \hat{W}(\boldsymbol{v}_1) > 0$, and finally $\lim_{\boldsymbol{p}\to\boldsymbol{v}_1} \hat{b}_i(\boldsymbol{p}) = \delta_{i,1}$. □

The proof of Lemma 2 does not extend to the case $p > 1$, because the upper bound in (3.11) diverges. Going back to the functions $\tilde{b}_i$ in (3.8), we see that they are not well-defined at the vertices of P, because all the $\tilde{w}_i$ and thus also $\tilde{W}$ are zero at $\boldsymbol{p} = \boldsymbol{v}_j$. However, for $p > 1$, this problem can be fixed by considering the functions $\tilde{w}_i/r_j$, $i = 1, \dots, n$, and $\tilde{W}/r_j$.

Lemma 3. *Exponential three-point coordinates extend continuously to $\bar{\Omega}$ for $p > 1$.*

Proof. As in the proof of Lemma 2, we consider only the case where $\boldsymbol{p}$ approaches $\boldsymbol{v}_1$. Like in (3.11), we can bound the quotients A_1/r_1 and A_n/r_1 for any $\boldsymbol{p} \in \Omega$ as

$$0 \leq \frac{A_1}{r_1} \leq \frac{e_1}{2}, \qquad 0 \leq \frac{A_n}{r_1} \leq \frac{e_n}{2}, \tag{3.14}$$

where $e_n = \|\boldsymbol{v}_n - \boldsymbol{v}_1\|$ is the length of E_n (see Figure 3.2). Since these bounds are constants, they also hold in the limit. For $i \neq 1$, we then observe that all terms of $\tilde{w}_i$ in (3.6) contain either A_1 or A_n plus one other factor (A_1, A_n, B_2, B_n, or r_1^p) that vanishes at $\boldsymbol{v}_1$, so that $\lim_{\boldsymbol{p}\to\boldsymbol{v}_1} \tilde{w}_i(\boldsymbol{p})/r_1(\boldsymbol{p}) = 0$. It remains to show that

$$\frac{\tilde{w}_1}{r_1} = \left(\frac{r_2^p A_n}{r_1} - r_1^{p-1} B_1 + \frac{r_n^p A_1}{r_1} \right) \mathcal{A}_{n,1},$$

and thus also $\tilde{W}/r_1$, converges to a non-zero, finite value. By (3.14),

$$\frac{r_2^p A_n}{r_1} + \frac{r_n^p A_1}{r_1} \leq \frac{r_2^p e_n + r_n^p e_1}{2}$$

for any $\boldsymbol{p} \in \Omega$ and this upper bound converges to the positive constant $c^\star = (e_1^p e_n + e_n^p e_1)/2$. Moreover,

$$\begin{aligned}
\frac{r_2^p A_n}{r_1} + \frac{r_n^p A_1}{r_1} &= \frac{r_2^p e_n \sin\tau_n + r_n^p e_1 \sin\tau_1}{2} \\
&\geq \min(r_2, r_n)^p \min(e_1, e_n)(\sin\tau_1 + \sin\tau_n)/2 \\
&\geq \min(r_2, r_n)^p \min(e_1, e_n)(\sin\tau_1 \cos\tau_n + \sin\tau_n \cos\tau_1)/2 \\
&= \min(r_2, r_n)^p \min(e_1, e_n) \sin(\tau_1 + \tau_n)/2,
\end{aligned}$$

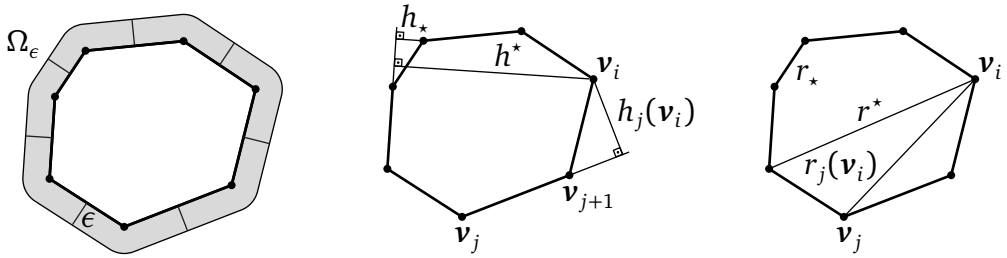


Figure 3.3. Notation used in Section 3.1.2.

where τ_n is the signed angle between $\boldsymbol{p}-\boldsymbol{v}_1$ and $\boldsymbol{v}_n-\boldsymbol{v}_1$ (see Figure 3.2), and this lower bound converges to the positive constant $c_\star = \min(e_1, e_n)^{p+1} \sin(\tau_1+\tau_n)/2$. It follows that $(r_2^p A_n + r_n^p A_1)/r_1$ converges to a positive, finite value $c \in [c_\star, c^\star]$ and since r_1^{p-1} vanishes at $\boldsymbol{p} = \boldsymbol{v}_1$ and $\mathcal{A}_{n,1}$ does not, the proof is complete. Note that the limit $c\mathcal{A}_{n-1}$ of $\tilde{w}_1/r_1$ may not be the same for two different sequences of $\boldsymbol{p}$, which both converge to $\boldsymbol{v}_1$, but this does not affect the proof, because the ratio $(\tilde{w}_1/r_1)/(\tilde{W}/r_1)$ always converges to 1. □

We are now ready to summarize our observations.

Theorem 1. *Exponential three-point coordinates are continuous generalized barycentric coordinates over $\bar{\Omega}$ for any $p \in \mathbb{R}$.*

Proof. It follows from Proposition 1 and Proposition 5 in [Floater et al., 2006] that exponential three-point coordinates are continuous and satisfy conditions (1.8) and (1.9) over Ω for any $p \in \mathbb{R}$. Proposition 2 and Lemmas 1, 2, and 3 further show that they can be extended continuously to $\bar{\Omega}$ for $p \neq 0, 1$ and that this extension satisfies condition (1.10b) and is piecewise linear along the boundary ∂P. For $p = 0$ and $p = 1$, the same boundary behaviour follows from Corollary 2 in [Floater et al., 2006], and it implies that conditions (1.8) and (1.9) hold for any point on ∂P and thus for any $\boldsymbol{p} \in \bar{\Omega}$. □

3.1.2 Convergence from outside

Let us now enlarge the domain from $\bar{\Omega}$ to the open set Ω_ϵ by adding all points $\boldsymbol{p} \in \mathbb{R}^2$, which are ϵ-close to Ω (see Figure 3.3), and analyse the continuity of exponential three-point coordinates over Ω_ϵ.

To this end, let $h_i(\boldsymbol{p})$ be the (shortest) distance between a point $\boldsymbol{p}$ and the line through $\boldsymbol{v}_i$ and $\boldsymbol{v}_{i+1}$, and let

$$h_\star = \min_{\substack{i,j=1,\dots,n \\ j\neq i-1,i}} h_j(\boldsymbol{v}_i), \qquad h^\star = \max_{i,j=1,\dots,n} h_j(\boldsymbol{v}_i)$$

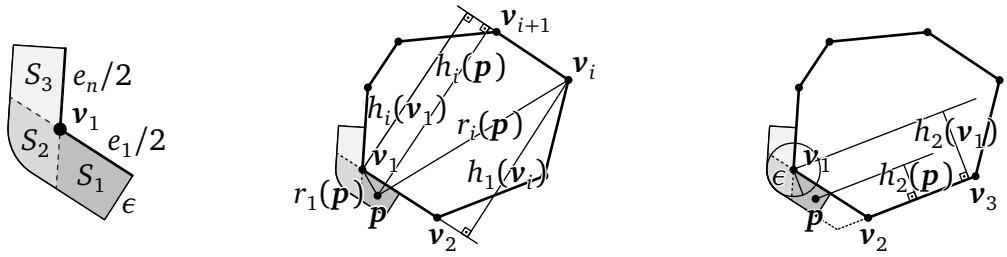


Figure 3.4. Notation used in the proof of Lemma 4.

be the minimum and maximum distance between the vertices and the supporting lines of P. We further denote the minima and maxima of distances between vertices of P, of edge lengths, and of areas C_i by

$$r_\star = \min_{\substack{i,j=1,\dots,n \\ j\neq i}} r_j(\boldsymbol{v}_i), \qquad r^\star = \max_{i,j=1,\dots,n} r_j(\boldsymbol{v}_i),$$
$$e_\star = \min_{i=1,\dots,n} \|\boldsymbol{v}_i - \boldsymbol{v}_{i+1}\|, \qquad e^\star = \max_{i=1,\dots,n} \|\boldsymbol{v}_i - \boldsymbol{v}_{i+1}\|,$$
$$C_\star = \min_{i=1,\dots,n} C_i, \qquad C^\star = \max_{i=1,\dots,n} C_i,$$

respectively and finally introduce the positive constants

$$c_\star = \min(h_\star, r_\star, e_\star, C_\star), \qquad c^\star = \min(h^\star, r^\star, e^\star, C^\star), \tag{3.15}$$

which we use for defining upper bounds on ϵ that guarantee $\tilde{W}$ to be positive over $\Omega_\epsilon \setminus \bar{\Omega}$ for $p < 1$.

Lemma 4. *If $p < 0$ and*

$$\epsilon < \frac{c_\star}{n8^n}\Big(\frac{c_\star}{c^\star}\Big)^{2n-p}, \tag{3.16}$$

then $\tilde{W}(\boldsymbol{p}) > 0$ for any $\boldsymbol{p} \in \Omega_\epsilon \setminus \bar{\Omega}$.

Proof. Since $n \geq 3$, $p < 0$, and $c_\star \leq c^\star$, we conclude from (3.16) that

$$\epsilon < c_\star/4. \tag{3.17}$$

Without loss of generality, we now focus on the situation around $\boldsymbol{v}_1$ and consider the three regions (see Figure 3.4)

$$\begin{aligned} S_1 &= \{\boldsymbol{p} \in \Omega_\epsilon : A_1(\boldsymbol{p}) < 0, A_n(\boldsymbol{p}) \geq 0, r_1(\boldsymbol{p}) \leq r_2(\boldsymbol{p})\}, \\ S_2 &= \{\boldsymbol{p} \in \Omega_\epsilon : A_1(\boldsymbol{p}) < 0, A_n(\boldsymbol{p}) < 0\}, \\ S_3 &= \{\boldsymbol{p} \in \Omega_\epsilon : A_1(\boldsymbol{p}) \geq 0, A_n(\boldsymbol{p}) < 0, r_1(\boldsymbol{p}) \leq r_n(\boldsymbol{p})\}, \end{aligned} \tag{3.18}$$

because all other cases follow by symmetry.

Let us start with the case $\boldsymbol{p} \in S_1$ and establish some bounds for $r_i(\boldsymbol{p})$ and $A_i(\boldsymbol{p})$. Since $\boldsymbol{p}$ is closer to $\boldsymbol{v}_1$ than to $\boldsymbol{v}_2$, we can use the triangle inequality and (3.17) to get

$$r_1(\boldsymbol{p}) \le e_1/2 + \epsilon < e^\star/2 + c_\star/4 < c^\star$$

and thus

$$r_1^p > (c^\star)^p, \tag{3.19}$$

because $p < 0$. Moreover, since $\boldsymbol{p}$ and $\boldsymbol{v}_i$ for $i \ge 3$ lie on opposite sides of the line through $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$, we have

$$r_i(\boldsymbol{p}) > h_1(\boldsymbol{v}_i) \ge h_\star \ge c_\star, \qquad i = 3, \dots, n. \tag{3.20}$$

We next derive some bounds for $h_i(\boldsymbol{p})$, which then turn into bounds for $A_i(\boldsymbol{p})$ because $|A_i(\boldsymbol{p})| = e_i h_i(\boldsymbol{p})/2$. We first note that $h_1(\boldsymbol{p}) < \epsilon$, hence

$$|A_1(\boldsymbol{p})| = e_1 h_1(\boldsymbol{p})/2 < e^\star \epsilon < c^\star \epsilon. \tag{3.21}$$

In general, we can get an upper bound for all $h_i(\boldsymbol{p})$ by triangle inequality,

$$h_i(\boldsymbol{p}) \le h_i(\boldsymbol{v}_1) + r_1(\boldsymbol{p}) \le h^\star + r^\star < 2c^\star.$$

For $i = 2$, a lower bound can be obtained by recalling that $\boldsymbol{p}$ is closer to $\boldsymbol{v}_1$ than to $\boldsymbol{v}_2$, so that

$$h_2(\boldsymbol{p}) > h_2(\boldsymbol{v}_1)/2 - \epsilon > h_\star/2 - c_\star/4 \ge c_\star/4.$$

For $i \ge 3$, the minimum distance from any point on the edge $[\boldsymbol{v}_1, \boldsymbol{v}_2]$ to the line through $\boldsymbol{v}_i$ and $\boldsymbol{v}_{i+1}$ is either $h_i(\boldsymbol{v}_1)$ or $h_i(\boldsymbol{v}_2)$, and so, since $\boldsymbol{p}$ is ϵ-close to $[\boldsymbol{v}_1, \boldsymbol{v}_2]$,

$$h_i(\boldsymbol{p}) > \min(h_i(\boldsymbol{v}_1), h_i(\boldsymbol{v}_2)) - \epsilon > h_\star - c_\star/4 > c_\star/4.$$

Overall, we conclude that

$$\frac{(c_\star)^2}{8} < A_i(\boldsymbol{p}) < (c^\star)^2, \qquad i = 2, \dots, n. \tag{3.22}$$

The idea now is to use (3.4) to rewrite $\tilde{W}$ in (3.7) as

$$\tilde{W} = r_1^p C_1 \mathcal{A}_{n,1} + r_2^p C_2 \mathcal{A}_{1,2} - |A_1| \sum_{i=3}^{n} r_i^p C_i \mathcal{A}_{1,i-1,i} \tag{3.23}$$

with

$$\mathcal{A}_{1,i-1,i} = \prod_{\substack{j=2 \\ j \ne i-1,i}}^{n} A_j,$$

and to show that the first term in (3.23) dominates the last term. To this end, we observe that

$$\begin{aligned}\frac{(c^\star)^p C_1 \mathcal{A}_{n,1}}{c^\star \sum_{i=3}^n r_i^p C_i \mathcal{A}_{1,i-1,i}} &\overset{(3.15)}{>} \frac{(c^\star)^p c_\star \mathcal{A}_{n,1}}{c^\star \sum_{i=3}^n r_i^p c^\star \mathcal{A}_{1,i-1,i}} \\ &\overset{(3.20)}{>} \frac{(c^\star)^p c_\star \mathcal{A}_{n,1}}{c^\star \sum_{i=3}^n (c_\star)^p c^\star \mathcal{A}_{1,i-1,i}} \\ &\overset{(3.22)}{>} \frac{(c^\star)^p c_\star (c_\star)^{2(n-2)}/8^{n-2}}{c^\star \sum_{i=3}^n (c_\star)^p c^\star (c^\star)^{2(n-3)}} = \frac{(c_\star)^{2n-3-p}}{(n-2)8^{n-2}(c^\star)^{2n-4-p}} \\ &\overset{(3.16)}{>} 2\epsilon,\end{aligned}$$

where we obtain the second inequality by recalling that $p < 0$, and so

$$\frac{1}{2}(c^\star)^p C_1 \mathcal{A}_{n,1} > c^\star \epsilon \sum_{i=3}^n r_i^p C_i \mathcal{A}_{1,i-1,i}.$$

Using (3.19) and (3.21), we then conclude

$$\frac{1}{2} r_1^p C_1 \mathcal{A}_{n,1} > |A_1| \sum_{i=3}^n r_i^p C_i \mathcal{A}_{1,i-1,i}, \tag{3.24}$$

which implies $\tilde{W} > 0$, and similar arguments lead to

$$\frac{1}{2} r_1^p C_1 \mathcal{A}_{n,1} > |A_n| \sum_{i=2}^{n-1} r_i^p C_i \mathcal{A}_{n,i-1,i} \tag{3.25}$$

for the case $\boldsymbol{p} \in S_3$. If $\boldsymbol{p} \in S_2$, then we rewrite $\tilde{W}$ as

$$\tilde{W} = r_1^p C_1 \mathcal{A}_{n,1} - |A_1| r_n^p C_n \mathcal{A}_{1,n-1,n} - |A_n| r_2^p C_2 \mathcal{A}_{n,1,2} + \sum_{i=3}^{n-1} r_i^p C_i \mathcal{A}_{i-1,i},$$

which is positive because of (3.24) and (3.25), which are also valid in this case. □

Lemma 5. *If* $0 \le p < 1$ *and*

$$\epsilon < c_\star \left(\frac{1}{n 8^n} \left(\frac{c_\star}{c^\star} \right)^{2n} \right)^{\frac{1}{1-p}}, \tag{3.26}$$

then $\tilde{W}(\boldsymbol{p}) > 0$ *for any* $\boldsymbol{p} \in \Omega_\epsilon \setminus \bar{\Omega}$.

Proof. As in the proof of Lemma 4, it follows from (3.26) that $\epsilon < c_\star/4$, and we proceed by considering the first of the three regions in (3.18). For any $\boldsymbol{p} \in S_1$, the bounds in (3.21) and (3.22) still hold, and we further observe that

$$|A_1(\boldsymbol{p})| = e_1 h_1(\boldsymbol{p})/2 < e^\star r_1(\boldsymbol{p})/2 < c^\star r_1(\boldsymbol{p})$$

and therefore

$$r_1^p \geq (|A_1|/c^\star)^p. \tag{3.27}$$

Moreover, by triangle inequality we get the upper bound

$$r_i(\boldsymbol{p}) \leq r_1(\boldsymbol{p}) + r_i(\boldsymbol{v}_1) < e_1/2 + \epsilon + r_i(\boldsymbol{v}_1) < c^\star/2 + c^\star/4 + c^\star < 2c^\star \tag{3.28}$$

for any i. With these bounds at hand we conclude that

$$\begin{aligned}
&\frac{C_1\mathcal{A}_{n,1}}{(c^\star)^p \sum_{i=3}^n r_i^p C_i \mathcal{A}_{1,i-1,i}} \\
&\overset{(3.15)}{>} \frac{c_\star \mathcal{A}_{n,1}}{(c^\star)^p \sum_{i=3}^n r_i^p c^\star \mathcal{A}_{1,i-1,i}} \\
&\overset{(3.28)}{\geq} \frac{c_\star \mathcal{A}_{n,1}}{(c^\star)^p \sum_{i=3}^n (2c^\star)^p c^\star \mathcal{A}_{1,i-1,i}} \\
&\overset{(3.22)}{>} \frac{c_\star (c_\star)^{2(n-2)}/8^{n-2}}{(c^\star)^p \sum_{i=3}^n (2c^\star)^p c^\star (c^\star)^{2(n-3)}} = \frac{(c^\star)^{2(1-p)}}{2^p (n-2) 8^{n-2}} \Big(\frac{c_\star}{c^\star}\Big)^{2n-3} \\
&> 2(c^\star)^{1-p}(c_\star)^{1-p} \frac{1}{n8^n} \Big(\frac{c_\star}{c^\star}\Big)^{2n} \\
&\overset{(3.26)}{>} 2(c^\star \epsilon)^{1-p} \\
&\overset{(3.21)}{>} 2|A_1|^{1-p},
\end{aligned}$$

so that

$$\frac{1}{2}(|A_1|/c^\star)^p C_1 \mathcal{A}_{n,1} > |A_1| \sum_{i=3}^n r_i^p C_i \mathcal{A}_{1,i-1,i}.$$

Using (3.27), we then get

$$\frac{1}{2} r_1^p C_1 \mathcal{A}_{n,1} > |A_1| \sum_{i=3}^n r_i^p C_i \mathcal{A}_{1,i-1,i},$$

which implies $\tilde{W} > 0$, exactly as in the proof of Lemma 4, and also the other cases $\boldsymbol{p} \in S_3$ and $\boldsymbol{p} \in S_2$ follow analogously. □

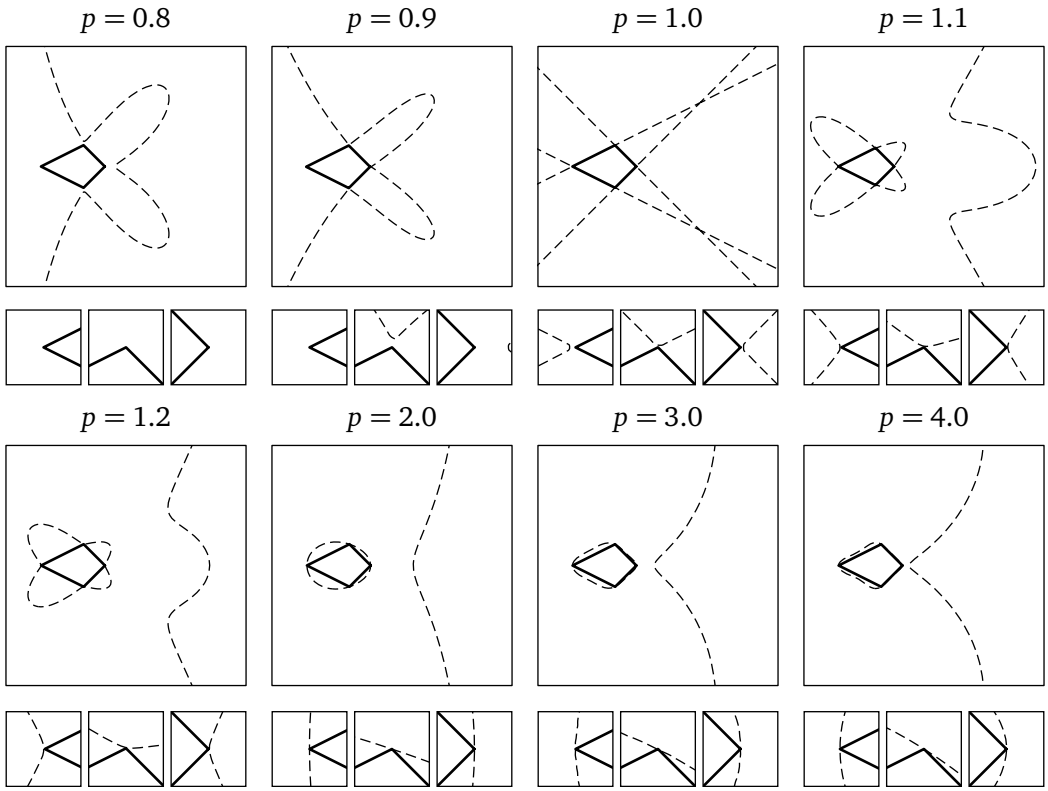


Figure 3.5. Zero level set $\{\boldsymbol{p} \in \mathbb{R}^2 : \tilde{W}(\boldsymbol{p}) = 0\}$ (dashed) for different values of p and close-ups, which zoom to the regions next to the left, top, and right corners of the polygon.

The reasoning in Lemma 5 does not extend to the case $p = 1$, because the upper bound in (3.26) converges to zero as p approaches 1. This suggests that $\tilde{W}$ vanishes at the vertices of P for $p \geq 1$, and Figure 3.5 confirms that the zero level curve $\{\boldsymbol{p} \in \mathbb{R}^2 : \tilde{W}(\boldsymbol{p}) = 0\}$ passes through the vertices of P for $p \geq 1$. For $p = 1$, this is not a problem, because Hormann and Floater [2006] prove that the corresponding mean value coordinates are continuous over $\mathbb{R}^2$. But the following two examples show that exponential three-point coordinates for $p > 1$ can have non-removable singularities in $\mathbb{R}^2 \setminus \bar{\Omega}$ arbitrarily close to the vertices of P, and so they are, in general, not continuous over Ω_ϵ for any $\epsilon > 0$. Note that the polygons in both examples were chosen to keep the calculations as simple as possible, but that we observed the same phenomena for all other polygons that we tested.

Example 1. Let us examine the exponential three-point coordinates for $1 < p < 2$ over the unit square with vertices $\boldsymbol{v}_1 = (0,0)$, $\boldsymbol{v}_2 = (0,1)$, $\boldsymbol{v}_3 = (-1,1)$, $\boldsymbol{v}_4 =$

$(-1,0)$. For $x \geq 0$, it turns out that

$$\tilde{W}(x,0) = (x^p(1+x) - (1+x)^p x)/8,$$

hence $\tilde{W}(0,0) = 0$ and $(\partial \tilde{W}/\partial x)(0,0) = -1/8$, because $p > 1$. Consequently, there exists some $\epsilon \in (0,1)$, such that $\tilde{W}$ is negative over the open edge between $\boldsymbol{v}_1 = (0,0)$ and $\boldsymbol{p}_\epsilon^x = (\epsilon, 0)$, and from Proposition 2 we know that $\tilde{W}$ is positive over the open edge between $\boldsymbol{v}_1$ and $\boldsymbol{p}_\epsilon^y = (0,\epsilon)$. It follows that for any $\delta \in (0,\epsilon)$ there exists some point $\boldsymbol{p}_\delta$ on the open edge $(\boldsymbol{p}_\delta^x, \boldsymbol{p}_\delta^y)$, such that $\tilde{W}(\boldsymbol{p}_\delta) = 0$.

At least for the coordinate function $b_3 = \tilde{w}_3/\tilde{W}$ it is easy to see that these singularities are non-removable close to $\boldsymbol{v}_1$, because $\tilde{w}_3$ is negative over the open triangle $\triangle_\delta = (\boldsymbol{v}_1, \boldsymbol{p}_\delta^x, \boldsymbol{p}_\delta^y)$ for δ sufficiently small. To see this, we recall from (3.6) that $\tilde{w}_3 = \bar{w}_3 A_1 A_4$ with

$$\bar{w}_3(\boldsymbol{p}) = r_4(\boldsymbol{p})^p A_2(\boldsymbol{p}) - r_3(\boldsymbol{p})^p B_3(\boldsymbol{p}) + r_2(\boldsymbol{p})^p A_3(\boldsymbol{p}).$$

Since $\bar{w}_3(\boldsymbol{v}_1) = 1 - 2^{\frac{p}{2}-1} > 0$ for $p < 2$, there exists some $\delta > 0$ such that $\bar{w}_3$ is positive over $\triangle_\delta$. Therefore, $\tilde{w}_3$ is negative over $\triangle_\delta$, because A_1 is negative and A_4 is positive over this region.

Despite the existence of these non-removable singularities, it seems hard to find an example of a sequence $(u_k)_{k\in\mathbb{N}}$ with $\lim_{k\to\infty} u_k = \boldsymbol{v}_j$, such that the $\lim_{k\to\infty} b_i(u_k) \neq b_i(\boldsymbol{v}_j)$ in the case $1 < p < 2$. In particular, our numerical experiments suggest that b_i always converges to the correct value at $\boldsymbol{v}_j$, if $\boldsymbol{v}_j$ is approached along any line through $\boldsymbol{v}_j$. This is not the case for $p \geq 2$, though.

Example 2. For the case $p \geq 2$, we consider the quadrilateral with vertices $\boldsymbol{v}_1 = (1,0)$, $\boldsymbol{v}_2 = (0,1)$, $\boldsymbol{v}_3 = (-2,0)$, $\boldsymbol{v}_4 = (0,-1)$ and study the behaviour of b_1 along the vertical ray $R = \{(1,y) : y > 0\}$. For $p = 2$, we find that $b_1(1,y) = (9-4y^2)/15$ for $y > 0$, hence

$$\lim_{\substack{\boldsymbol{p}\to\boldsymbol{v}_1 \\ \boldsymbol{p}\in R}} b_1(\boldsymbol{p}) = \lim_{y\to 0^+} b_1(1,y) = 3/5 < 1 = b_1(\boldsymbol{v}_1),$$

which shows that b_1 is not continuous over Ω_ϵ for any $\epsilon > 0$. For $p > 2$, we get

$$w_1(1,y) = 2\frac{(1+(1+y)^2)^{\frac{p}{2}} - (1+(1-y)^2)^{\frac{p}{2}}}{y} - 4y^{p-2}$$

for $y > 0$, and using L'Hôpital's rule, we obtain

$$\lim_{y\to 0} w_1(1,y) = 2^{\frac{p}{2}+1} p.$$

Similar reasoning shows that

$$\lim_{y\to 0} W(1,y) = 2^{\frac{p}{2}+1}p + 8(3^{p-1} - 2^{\frac{p}{2}})/3,$$

hence

$$\lim_{\substack{\boldsymbol{p}\to\boldsymbol{v}_1\\ \boldsymbol{p}\in R}} b_1(\boldsymbol{p}) = \lim_{y\to 0^+} \frac{w_1(1,y)}{W(1,y)} = \frac{9p}{9p + 3^p 2^{2-\frac{p}{2}} - 12} < 1 = b_1(\boldsymbol{v}_1),$$

which again demonstrates that b_1 is not continuous over Ω_ϵ for any $\epsilon > 0$.

We should point out that the direction of the ray R does not by chance happen to be tangent to the zero level curve $\{\boldsymbol{p} \in \mathbb{R}^2 : \tilde{W}(\boldsymbol{p}) = 0\}$ at $\boldsymbol{v}_1$ in this example. In fact, our numerical experiments suggest that b_i converges to the correct value at $\boldsymbol{v}_j$ along any other line through $\boldsymbol{v}_j$. Overall, we conclude this section by summarizing our observations.

Theorem 2. *For any $p \leq 1$, there exists an $\epsilon > 0$, such that the exponential three-point coordinates are continuous barycentric coordinates that satisfy* (1.8)*,* (1.9)*, and* (1.10b) *over Ω_ϵ.*

Proof. For $p = 1$, the statement is proven in [Hormann and Floater, 2006], and for $p < 1$, it follows from Theorem 1, Lemmas 4 and 5, and by noting that Lemmas 1 and 2 carry over from $\bar{\Omega}$ to Ω_ϵ. The proof of Lemma 1 extends because $\hat{W}(\boldsymbol{p}) > 0$ for any $\boldsymbol{p} \in \Omega_\epsilon$, and the only change in the proof of Lemma 2 is that the lower bound for the ratio A_1/r_1^p in (3.11) must be replaced by $-r_1^{1-p}e_1/2$, because A_1 can now be negative, but this does not affect the limit in (3.12) and similarly for the limit in (3.13). □

3.2 Practical aspects

A naive implementation of exponential three-point coordinates is quite straightforward, however to make this implementation numerically stable for a general set of polygons, a few problems have to be solved. We first explain these problems and show how to solve them for a general value p in Section 3.2.1. We then further consider three special cases, $p = 0$ for Wachspress coordinates (Section 3.2.2), $p = 2$ for discrete harmonic coordinates (Section 3.2.3), and $p = 1$ for mean value coordinates (Section 3.2.4).

3.2.1 Computing exponential three-point coordinates

Given a strictly convex polygon, for any $\boldsymbol{p} \in \Omega$ and $p \in \mathbb{R}$, the exponential three point coordinates $b_i(\boldsymbol{p})$ can be computed through the normalization (3.2) with the weights

$$w_i(\boldsymbol{p}) = \frac{r_{i+1}^p(\boldsymbol{p})A_{i-1}(\boldsymbol{p}) - r_i^p(\boldsymbol{p})B_i(\boldsymbol{p}) + r_{i-1}^p(\boldsymbol{p})A_i(\boldsymbol{p})}{A_{i-1}(\boldsymbol{p})A_i(\boldsymbol{p})}, \qquad i = 1, \dots, n, \quad (3.29)$$

as long as the denominator (3.3) does not vanish. As shown in Proposition 1, this denominator never vanishes for any value of the exponent p. However, when $\boldsymbol{p}$ is approaching an open edge E_i, the area $A_i(\boldsymbol{p})$ converges to zero, and our numerical experiments reveal that $w_i(\boldsymbol{p})$ and $w_{i+1}(\boldsymbol{p})$ are not well-defined not only along E_i (see Section 3.1) but also at some $\boldsymbol{p} \in \Omega$ in the vicinity of E_i (in general, approximately 10^{-10} away from E_i and closer). The latter problem is caused by the numerical division by zero that happens because of the insufficient precision of the standard real data types used in computers. Since exponential three-point coordinates comply with the property (1.10c), they can be computed separately for all $\boldsymbol{p} \in \partial P$ by linearly interpolating between the vertices of P (see Equations (3.32) below). For other $\boldsymbol{p}$, there are three different ways how to avoid the numerical instabilities.

The first option is to use a multiple precision library, for example MPFR [MPFR, 2016], however, in general, these libraries are not easy to use and they do not provide a full solution to the problem. For any defined precision, there always exists some $\boldsymbol{p} \in \Omega$ in the vicinity of the polygon's boundary, where this precision would not be enough to avoid all numerical instabilities. In addition, the computation becomes slower.

Another way to fix the problem is to treat all $\boldsymbol{p} \in \Omega$, such that for some chosen threshold $\epsilon > 0$ the (shortest) distance

$$h_i(\boldsymbol{p}) = \left\| \boldsymbol{p}' - \boldsymbol{p} \right\| < \epsilon, \quad (3.30)$$

where $\boldsymbol{p}' \in \partial P$ is the orthogonal projection of $\boldsymbol{p}$ onto the segment $[\boldsymbol{v}_i, \boldsymbol{v}_{i+1}]$ (see

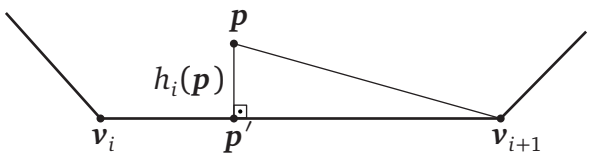


Figure 3.6. The orthogonal projection $\boldsymbol{p}'$ of the point $\boldsymbol{p}$ onto the segment $[\boldsymbol{v}_i, \boldsymbol{v}_{i+1}]$ and the shortest distance $h_i(\boldsymbol{p})$ between these points.

Figure 3.6), as boundary points. Alternatively, condition (3.30) can be substituted by condition

$$|A_i(\boldsymbol{p})| = |(\boldsymbol{v}_i - \boldsymbol{p}) \times (\boldsymbol{v}_{i+1} - \boldsymbol{p})| < \epsilon \qquad \text{and} \qquad (\boldsymbol{v}_i - \boldsymbol{p}) \cdot (\boldsymbol{v}_{i+1} - \boldsymbol{p}) < 0, \quad (3.31)$$

where "×" and "·" denote the cross and dot product, respectively. If condition (3.31) is true, we set the coordinates $b_j(\boldsymbol{p}), j \neq i, i+1$ to zero, and the coordinates $b_i(\boldsymbol{p})$ and $b_{i+1}(\boldsymbol{p})$ are computed with respect to the segment $[\boldsymbol{v}_i, \boldsymbol{v}_{i+1}]$ as (see also Chapter 1)

$$b_i(\boldsymbol{p}) = \frac{\|\boldsymbol{p}' - \boldsymbol{v}_{i+1}\|}{e_i} = \frac{(\boldsymbol{p} - \boldsymbol{v}_{i+1}) \cdot (\boldsymbol{v}_i - \boldsymbol{v}_{i+1})}{e_i^2}, \qquad b_{i+1}(\boldsymbol{p}) = 1 - b_i(\boldsymbol{p}), \quad (3.32)$$

where the edge length $e_i = \|\boldsymbol{v}_i - \boldsymbol{v}_{i+1}\|$ (see Figure 3.2). This solution works well in practice, however the threshold ϵ must be carefully chosen for each new polygon by the user, and snapping to the boundary only approximates the correct result at the given $\boldsymbol{p}$.

The last option to solve the numerical problems above is to use the modified weights (3.6) and to compute the coordinates as in (3.8). As shown in Proposition 2, these modified weights lead to the well-defined coordinates $\tilde{b}_i(\boldsymbol{p})$ for all $\boldsymbol{p} \in \Omega \cup E_1 \cup \ldots \cup E_n$. Although we demonstrate in Section 3.1.1 that this definition does not extend to the whole $\bar{\Omega}$, due to Theorem 1, the coordinates $\tilde{b}_i$ at the vertices $\boldsymbol{v}_j$ can be obtained separately according to the Lagrange property $\tilde{b}_i(\boldsymbol{v}_j) = \delta_{ij}$. To detect the situation when $\boldsymbol{p}$ is at the vertex $\boldsymbol{v}_i$ of the polygon, it is enough to check if

$$r_i^2(\boldsymbol{p}) = (\boldsymbol{v}_i - \boldsymbol{p}) \cdot (\boldsymbol{v}_i - \boldsymbol{p}) = 0. \quad (3.33)$$

The third solution is more general than the first two, but it comes at the price of computational speed. Since, for each weight (3.6), the products $\mathcal{A}_i$, $\mathcal{A}_{i-1,i}$, and $\mathcal{A}_{i-1}$ with $n-2$ elements have to be computed, the time to evaluate the modified coordinate functions $\tilde{b}_i$ at one point $\boldsymbol{p}$ with respect to n vertices of the polygon is $O(n^2)$. The time to compute the original coordinates $b_i(\boldsymbol{p})$ for n vertices of the polygon is only $O(n)$.

We know that for any $\boldsymbol{p} \in \Omega_\epsilon \setminus \bar{\Omega}$, Theorem 2 states that exponential three-point coordinates with $p \leq 1$ satisfy the partition of unity (1.8), linear reproduction (1.9), and the Lagrange (1.10b) property. However, in practice, only the coordinates with $p = 1$ (or mean value coordinates) can be computed safely for such exterior points, because they are well-defined everywhere in $\mathbb{R}^2$ [Hormann and Floater, 2006]. For other values of p, this ϵ-region, where the coordinates are valid, may happen to be too small for any practical applications. At some

other points, the denominator W may vanish (see Figure 3.5 for $p \leq 1$) and unavoidable division by zero occurs.

To conclude this section, we want to remark that all exponential three-point coordinates with $p = \{2k,\ k \in \mathbb{Z}\}$ can be computed symbolically or in other words exactly, because, for any $\boldsymbol{p} \in \mathbb{R}^2$, the signed areas $A_i(\boldsymbol{p}) = \text{Area}(\boldsymbol{p}, \boldsymbol{v}_i, \boldsymbol{v}_{i+1})$ and $B_i(\boldsymbol{p}) = \text{Area}(\boldsymbol{p}, \boldsymbol{v}_{i-1}, \boldsymbol{v}_{i+1})$ can be computed symbolically by using the determinant definition of area as in Section 1.1 and the distance $r_i^{2k}(\boldsymbol{p})$ can be represented as

$$r_i^{2k}(\boldsymbol{p}) = ((\boldsymbol{v}_i - \boldsymbol{p}) \cdot (\boldsymbol{v}_i - \boldsymbol{p}))^k, \qquad i = 1, \ldots, n.$$

All exponential coordinates with $p = \{2k+1,\ k \in \mathbb{Z}\}$ involve a square root operation and can be only approximated.

3.2.2 Computing Wachspress coordinates

To compute Wachspress coordinates, one approach would be to use (3.29) with $p = 0$, however this formula can be simplified if we recall that $C_i = A_{i-1} + A_i - B_i$, where $C_i = \text{Area}(\boldsymbol{v}_{i-1}, \boldsymbol{v}_i, \boldsymbol{v}_{i+1})$ is a constant area (see Figure 3.1), and all C_i have to be computed only once. In this case, the weights for computing Wachspress coordinates become

$$w_i(\boldsymbol{p}) = \frac{C_i}{A_{i-1}(\boldsymbol{p}) A_i(\boldsymbol{p})}, \qquad i = 1, \ldots, n. \tag{3.34}$$

These weights are not numerically stable for some $\boldsymbol{p} \in \Omega$ in the vicinity of the polygon's boundary, as discussed in Section 3.2.1, and can be modified as

$$\tilde{w}_i(\boldsymbol{p}) = C_i \mathcal{A}_{i-1,i}(\boldsymbol{p}), \qquad i = 1, \ldots, n, \tag{3.35}$$

where the product $\mathcal{A}_{i-1,i}$ is defined in (3.5). The coordinate functions $\tilde{b}_i(\boldsymbol{p})$ in (3.8) with the modified weights $\tilde{w}_i(\boldsymbol{p})$ are well-defined for all $\boldsymbol{p} \in \Omega \cup E_1 \cup \ldots, \cup E_n$ (see Section 3.1), and at the vertex $\boldsymbol{v}_j$ they are computed according to the Lagrange property $\tilde{b}_i(\boldsymbol{v}_j) = \delta_{ij}$.

An alternative way to compute Wachspress coordinates is to use the original definition (2.1). Since the direct computation of the cotangents $\cot\beta_i$ and $\cot\gamma_i$ (see Figure 2.2 for angle notation) is not error-resistant, Meyer et al. [2002] propose to compute them for each index i as

$$\cot\beta_i = \frac{(\boldsymbol{v}_{i+1} - \boldsymbol{v}_i) \cdot (\boldsymbol{p} - \boldsymbol{v}_i)}{|(\boldsymbol{v}_{i+1} - \boldsymbol{v}_i) \times (\boldsymbol{p} - \boldsymbol{v}_i)|}, \qquad \cot\gamma_i = \frac{(\boldsymbol{v}_i - \boldsymbol{v}_{i+1}) \cdot (\boldsymbol{p} - \boldsymbol{v}_{i+1})}{|(\boldsymbol{v}_i - \boldsymbol{v}_{i+1}) \times (\boldsymbol{p} - \boldsymbol{v}_{i+1})|}. \tag{3.36}$$

However, this formulation is not numerically stable for some $\boldsymbol{p} \in \Omega$ in the vicinity of the polygon's boundary as well as the one in (3.34).

According to our numerical experiments, we suggest to use the weights (3.34) with the $O(n)$ time complexity to implement Wachspress coordinates for all applications, where a query point is known to be at least 10^{-10} distance away from the polygon's boundary (see Pseudocode 2 in Appendix A), and the weights (3.35) with the $O(n^2)$ time complexity for all other applications (see Pseudocode 3 in Appendix A).

3.2.3 Computing discrete harmonic coordinates

Analogously to Section 3.2.2, discrete harmonic coordinates can be computed by using (3.29) with $p = 2$,

$$w_i(\boldsymbol{p}) = \frac{r_{i+1}^2(\boldsymbol{p})A_{i-1}(\boldsymbol{p}) - r_i^2(\boldsymbol{p})B_i(\boldsymbol{p}) + r_{i-1}^2(\boldsymbol{p})A_i(\boldsymbol{p})}{A_{i-1}(\boldsymbol{p})A_i(\boldsymbol{p})}, \qquad i = 1, \ldots, n \quad (3.37)$$

or its numerically stable version (see Section 3.2.1 for more details)

$$\tilde{w}_i(\boldsymbol{p}) = r_{i+1}^2(\boldsymbol{p})\mathcal{A}_i(\boldsymbol{p}) - r_i^2(\boldsymbol{p})B_i(\boldsymbol{p})\mathcal{A}_{i-1,i}(\boldsymbol{p}) + r_{i-1}^2(\boldsymbol{p})\mathcal{A}_{i-1}(\boldsymbol{p}), \qquad i = 1, \ldots, n. \quad (3.38)$$

In addition, the original formulation (2.2) can also be used, where the cotangents $\cot\beta_i$ and $\cot\gamma_i$ (see Figure 2.2 for angle notation) are computed as in (3.36), but it exhibits the same numerical issues as the weights in (3.37).

According to our numerical experiments, we suggest to use the weights (3.37) with the $O(n)$ time complexity to implement discrete harmonic coordinates for all applications, where a query point is known to be at least 10^{-10} distance away from the polygon's boundary (see Pseudocode 4 in Appendix A), and the weights (3.38) with the $O(n^2)$ time complexity for all other applications (see Pseudocode 5 in Appendix A).

3.2.4 Computing mean value coordinates

Mean value coordinates are the most interesting member of the exponential three-point coordinates, because they are well-defined for all $\boldsymbol{p} \in \mathbb{R}^2$ and for all simple polygons [Hormann and Floater, 2006]. Analogously to Sections 3.2.2 and 3.2.3, the first way to compute these coordinates is to use the weights (3.29) with $p = 1$ or its numerically stable version (see Section 3.2.1 for more details)

$$\tilde{w}_i(\boldsymbol{p}) = r_{i+1}(\boldsymbol{p})\mathcal{A}_i(\boldsymbol{p}) - r_i(\boldsymbol{p})B_i(\boldsymbol{p})\mathcal{A}_{i-1,i}(\boldsymbol{p}) + r_{i-1}(\boldsymbol{p})\mathcal{A}_{i-1}(\boldsymbol{p}), \qquad i = 1, \ldots, n.$$

It then follows that the coordinates $\tilde{b}_i(\boldsymbol{p})$ in (3.8) can be computed for all $\boldsymbol{p} \in \Omega \cup E_1 \cup \ldots, \cup E_n$, but they are not defined for any $\boldsymbol{p} = \boldsymbol{v}_j$ (see Section 3.1). While for other exponential three-point coordinates, we suggest to compute the $\tilde{b}_i(\boldsymbol{v}_j)$ separately according to the Lagrange property (1.10b), for mean value coordinates, there is a way to overcome this constraint and obtain a numerically stable formulation for all points in the plane.

Given a strictly convex polygon, let us consider the original definition (2.4) of mean value coordinates. Floater [2014] suggests to use the trigonometric identity (see Figure 2.2 for angle notation)

$$\tan\left(\frac{\alpha_i}{2}\right) = \sqrt{\frac{1-\cos\alpha_i}{1+\cos\alpha_i}}$$

in this definition, to obtain the weights

$$\hat{w}_i = (r_{i-1}r_{i+1} - \boldsymbol{s}_{i-1}\cdot\boldsymbol{s}_{i+1})^{1/2} \prod_{j\neq i-1,i} (r_j r_{j+1} + \boldsymbol{s}_j \cdot \boldsymbol{s}_{j+1})^{1/2}, \qquad i = 1,\ldots,n, \quad (3.39)$$

where $\boldsymbol{s}_i = \boldsymbol{v}_i - \boldsymbol{p}$ and $r_i = \|\boldsymbol{s}_i\|$, which after the normalization

$$\hat{b}_i = \frac{\hat{w}_i}{\hat{W}}, \qquad i = 1,\ldots,n$$

are identical to the b_i over Ω, but have the advantage of being well-defined over $\bar{\Omega}$. Since, by definition, the weights $\hat{w}_i$ are always non-negative, they cannot be used for computing the coordinates with respect to concave polygons and for points outside a polygon. We solve this problem by introducing the weights

$$\bar{w}_i = \sigma_i \hat{w}_i, \qquad i = 1,\ldots,n, \quad (3.40)$$

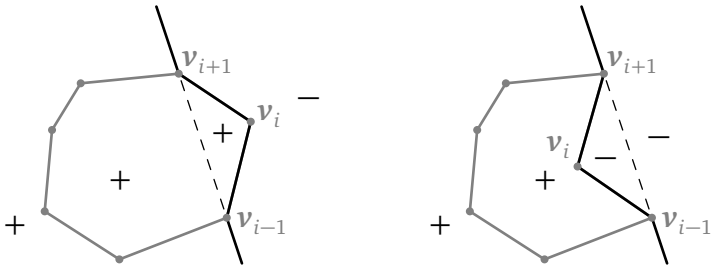


Figure 3.7. Signs of the mean value weight $\hat{w}_i(\boldsymbol{p})$ for a $\boldsymbol{p} \in \mathbb{R}^2$ with respect to a convex and a concave polygon. The black piecewise linear curve separates the positive and negative regions.

Pseudocode 1 Computing the sign $\sigma_i(\boldsymbol{p})$ of the mean value weight $\hat{w}_i(\boldsymbol{p})$ (see Equation (3.39)) for a point $\boldsymbol{p} \in \mathbb{R}^2$.

```
1: function σ_i(p)
2:     if A_{i-1}(p) > 0 and A_i(p) > 0 and B_i(p) ≤ 0 then return 1
3:     if A_{i-1}(p) < 0 and A_i(p) < 0 and B_i(p) ≥ 0 then return −1
4:     if B_i(p) > 0 then return 1
5:     if B_i(p) < 0 then return −1
6:     return 0
```

where σ_i is the sign function with the values shown in Figure 3.7. This sign function defines the correct sign of the weight $\hat{w}(\boldsymbol{p})$ with respect to any simple polygon and for any $\boldsymbol{p} \in \mathbb{R}^2$, and it can be computed as in Pseudocode 1.

Except for the formulas above, there are a few other ways how to obtain mean value coordinates. These formulations are not as precise as (3.40), but they offer a nice trade-off between computational speed and precision. As described in [Hormann and Floater, 2006], the first option is to use the trigonometric identity

$$\tan\left(\frac{\alpha_i}{2}\right) = \frac{1-\cos\alpha_i}{\sin\alpha_i} = \frac{r_i r_{i+1} - \boldsymbol{s}_i \cdot \boldsymbol{s}_{i+1}}{2A_i}, \qquad i = 1, \dots, n$$

in the original formulation (2.4). However, analogously to the weights (3.29) with $p = 1$, the $w_i(\boldsymbol{p})$ based on this identity involve division by the area $A_i(\boldsymbol{p})$ and are not numerically stable for $\boldsymbol{p} \in \mathbb{R}^2$ in the vicinity of the line through the vertices $\boldsymbol{v}_i$ and $\boldsymbol{v}_{i+1}$ (see Figure 3.5 for $p = 1$), where the numerical division by zero can happen. The problem can be partially fixed without sacrificing computational speed by considering the trigonometric identity

$$\tan\left(\frac{\alpha_i}{2}\right) = \frac{\sin\alpha_i}{1+\cos\alpha_i} = \frac{2A_i}{r_i r_{i+1} + \boldsymbol{s}_i \cdot \boldsymbol{s}_{i+1}}, \qquad i = 1, \dots, n. \tag{3.41}$$

The mean value weights based on this identity exhibit numerical instabilities only in the vicinity of the polygon's boundary when the angle α_i approaches 180° and so $\boldsymbol{s}_i \cdot \boldsymbol{s}_{i+1} = r_i r_{i+1} \cos\alpha_i \approx -r_i r_{i+1}$.

According to our numerical experiments, we suggest to use the weights (2.4) based on the trigonometric identity (3.41) with the $O(n)$ time complexity to implement mean value coordinates for all applications, where a query point is known to be at least 10^{-10} distance away from the polygon's boundary (see Pseudocode 6 in Appendix A), and the weights (3.40) with the $O(n^2)$ time complexity for all other applications (see Pseudocode 7 in Appendix A).

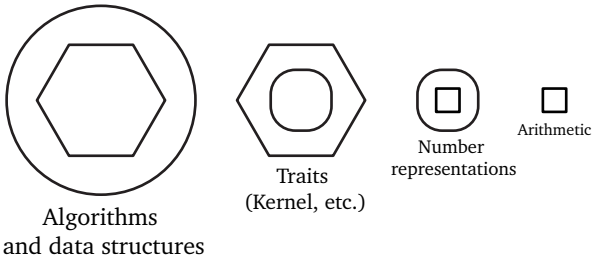


Figure 3.8. Generic design adopted in CGAL.

3.3 Computational Geometry Algorithm Library

The *Computational Geometry Algorithm Library* [CGAL, 2016] (CGAL) is an open source C++ library that provides efficient and reliable implementations of different geometric algorithms. It is used in geographic information systems, computer graphics, molecular biology, robotics, and many other areas. The key features of this library are the high quality of the provided code and accuracy of the algorithms behind this code.

Several algorithms in CGAL require an efficient and reliable implementation of some closed-form generalized barycentric coordinates. The current implementation of these coordinates is based on the old and error-prone formulas, and the code itself is not flexible enough for introducing new constructions of barycentric coordinates in the library. The latter suggests to create a new package with the state of the art constructions of some commonly used closed-form generalized barycentric coordinates.

We shortly explain the main design concepts that are adopted in CGAL in Section 3.3.1. We then show how these concepts are applied to implement the package with different generalized barycentric coordinates in Section 3.3.2 and discuss the package performance in Section 3.3.3.

3.3.1 Design concepts

The main goal of CGAL is to provide the users with correct, robust, and flexible implementations of different geometric algorithms. It means that each package needs to behave according to its specification, contain as precise as possible implementations of the geometric algorithms, be modular, extendable, and compatible with other C++ libraries.

To fulfil the flexibility requirement, this library follows a *generic design* and consists of three major layers (see Figure 3.8). The highest layer is the layer with

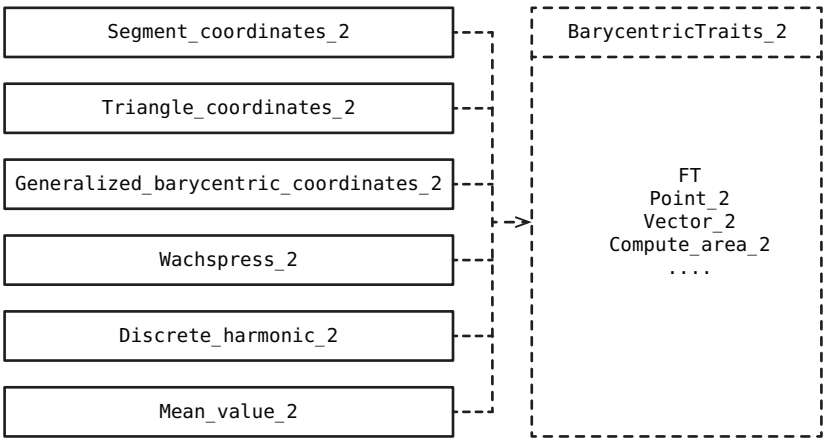


Figure 3.9. The traits class used in the package 2D Generalized Barycentric Coordinates.

geometric algorithms and data structures that are parameterized by the types of geometric objects and operations, which they use. These objects and operations are collected in special classes that are called the *traits classes*. A simple example of the traits class is a *geometric kernel* that defines different geometric objects such as points, lines, circles and operations on these objects such as intersections, orientations, and so on. The traits classes are in turn parameterized by different *number representations* with the corresponding *arithmetic*. For example, the geometric kernel above can be used either with floating-point number representations such as IEEE 754 [Standards Committee et al., 2008] or exact number representations such as LEDA rational [LEDA, 2016].

To describe a set of requirements for each class in CGAL, it is common to call this set of requirements a *concept*, where a concrete implementation of this concept that fulfils all the corresponding requirements is called a *model*. For example, `Cartesian<FT>` is the model of the concept `Kernel` that is parameterized by the field number type `FT`, contains different geometric primitives from the geometric kernel, and is used to parameterize different geometric algorithms.

All three layers are implemented using C++ templates, object-oriented solutions and generic design patterns. Many algorithms in CGAL are purely sequential, however the library permits and even encourages parallelization.

3.3.2 Package

Our package is called *2D Generalized Barycentric Coordinates* and is the part of the library's layer with geometric algorithms and data structures. This package

follows the generic programming paradigm employed in CGAL and contains different implementations of Wachspress (Section 3.2.2), discrete harmonic (Section 3.2.3), and mean value (Section 3.2.4) coordinates. For completeness, we also implemented barycentric coordinates with respect to a line segment (see Equation 3.32) and a triangle (see Section 1.1). The package consists of six classes, which are parameterized by the traits class that has to be a model of the concept `BarycentricTraits_2` (see Figure 3.9). This concept includes a field number type `FT` and the following

- 2D geometric objects:
  - `Point_2` : a 2D point,
  - `Vector_2` : a 2D vector,
- 2D constructions:
  - `Compute_area_2` : returns the signed area of a triangle in the plane,
  - `Compute_squared_distance_2` : returns the squared Euclidean distance between two points,
  - `Compute_squared_length_2` : returns the squared length of a vector,
  - `Compute_scalar_product_2` : returns the dot product of two vectors,
- 2D predicates:
  - `Equal_2` : checks if a point equals to another point,
  - `Collinear_2` : checks if three points are collinear,
  - `Collinear_are_ordered_along_line_2` : checks if a point is in between two other points and all three points are collinear.

All geometric kernels implemented in CGAL satisfy the criteria of the concept `BarycentricTraits_2` and can be used to parameterize the barycentric coordinate classes, where the three main classes are

- `Segment_coordinates_2`,
- `Triangle_coordinates_2`,
- `Generalized_barycentric_coordinates_2`.

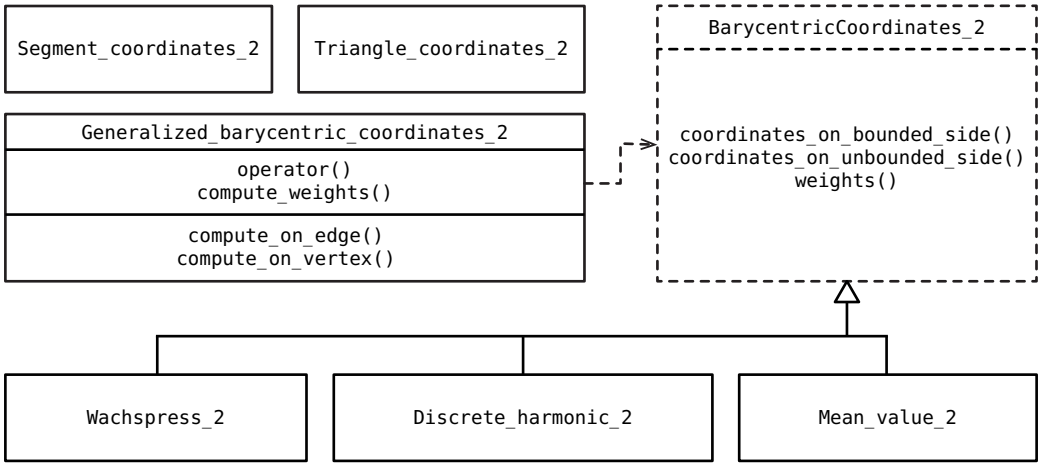


Figure 3.10. Generic design adopted in the package 2D Generalized Barycentric Coordinates.

While the first two classes contain direct implementations of barycentric coordinates with respect to a line segment and a triangle, the third class is the base class for all generalized barycentric coordinates and implemented according to the behavioural software design pattern called the *strategy pattern* [Gamma et al., 1995]. This class provides a common interface for a majority of closed-form generalized barycentric coordinates (see Figure 3.10) and is parameterized by the concrete coordinate class that has to be a model of the concept `BarycentricCoordinates_2`. This concept includes the class constructor and the three methods

- `weights` : returns generalized barycentric weights,
- `coordinates_on_bounded_side` : returns coordinates computed over the polygon's interior,
- `coordinates_on_unbounded_side` : returns coordinates computed outside the polygon,

which implement the main behaviour of the base class that can be accessed by the operator

- `operator()` : computes generalized barycentric coordinates for a query point in the plane, and by the method
- `compute_weights` : computes generalized barycentric weights for a query point in the plane.

The base class also contains the implementation of the correct behaviour (3.32) of the coordinates for a query point on the polygon's boundary that is provided by the methods

- `compute_on_edge` : computes generalized barycentric coordinates for a query point on a polygon's edge,
- `compute_on_vertex` : computes generalized barycentric coordinates for a query point at a polygon's vertex.

The concept `BarycentricCoordinates_2` has currently implemented three different models

- `Wachspress_2`,
- `Discrete_harmonic_2`,
- `Mean_value_2`,

and all three classes provide an option to choose between the fast $O(n)$ and the precise $O(n^2)$ implementation (see Section 3.2.1) of the corresponding coordinates.

All implementations are optimized for different computing platforms (for example, Windows, Linux, and Mac OS) and contain different data-checking mechanisms (for example, C++ and CGAL assertions), for situations such as wrong input data, division by zero, and many others. We also prepared and passed a test suite on a variety of computing platforms that contains 24 different tests controlling the correct behaviour of the package according to its specification. We finally remark that our package does not depend on other CGAL packages, and all classes are encapsulated in the namespace `Barycentric_coordinates` and so separated from the rest of the CGAL code.

Mean value coordinates are the most generic coordinates in this package, because they allow arbitrary simple polygons and 2D query points as input. Wachspress and discrete harmonic coordinates are, by definition, limited to strictly convex polygons only. Segment coordinates take as input any non-degenerate segment, and triangle coordinates allow an arbitrary non-degenerate triangle.

The main entry point to the component is an input iterator over the vertices of a polygon. The polygon's vertices must follow clockwise or anticlockwise ordering and can be of any type. However, internally the classes use the type `Point_2`, that is why an appropriate traits class that converts the user's point type to the type `Point_2` must be provided, and the same argument holds for query points.

Table 3.1. Some values related to the package 2D Generalized Barycentric Coordinates, where STL stands for the Standard Template Library in C++.

Property	**Value**
Version	1.0
Classes	6
Extendable design	yes
Supported CGAL kernels	any
Supported iterators	any STL like iterator
Supported query points	`Point_2` like points
Supported output containers	any
Number of tests	24
Number of examples	7
Number of performance experiments/trials	18/100

Once instantiated for some polygon, the coordinates can be computed multiple times for different query points with respect to all the vertices of the provided polygon. The output of the computation is a set of coordinate values that can be stored in an arbitrary container providing an appropriate output iterator. In addition, all coordinate classes return a pointer to the last stored element and a status of the computation (Boolean true or false). Some summarized statistics about the package can be found in Table 3.1, and a typical usage example can be found in Listing 3.1.

Listing 3.1. Computing mean value coordinates in CGAL.

```
// Example with mean value coordinates.

// Some CGAL includes.
#include <CGAL/Barycentric_coordinates_2/Mean_value_2.h>
#include <CGAL/Exact_predicates_inexact_constructions_kernel.h>
#include <CGAL/Barycentric_coordinates_2/Generalized_barycentric_coordinates_2.h>

// Some convenient typedefs.
typedef CGAL::Exact_predicates_inexact_constructions_kernel Kernel;

typedef Kernel::FT      Scalar;
typedef Kernel::Point_2 Point;

typedef std::vector<Scalar> Scalar_vector;
typedef std::vector<Point>  Point_vector;

typedef std::back_insert_iterator<Scalar_vector> Vector_insert_iterator;
typedef boost::optional<Vector_insert_iterator>  Output_type;
```

```
typedef CGAL::Barycentric_coordinates::
Mean_value_2<Kernel> Mean_value;

typedef CGAL::Barycentric_coordinates::
Generalized_barycentric_coordinates_2<Mean_value, Kernel> Mean_value_coordinates;

using std::cout; using std::endl; using std::string;

int main()
{
    // Construct a polygon.
    const int number_of_vertices = 10;
    Point_vector vertices(number_of_vertices);

    vertices[0] = Point(0, 0); vertices[1] = ...;

    // Instantiate some interior points in the polygon.
    const int number_of_interior_points = 8;
    const Point interior_points[] = { Point(1, 1), ... };

    // Instantiate the class with mean value coordinates for the polygon above.
    Mean_value_coordinates mean_value_coordinates(vertices.begin(), vertices.end());

    // We use the fast O(n) algorithm.
    const auto type_of_algorithm = CGAL::Barycentric_coordinates::FAST;

    // We compute coordinates in the polygon's interior.
    const auto query_point_location = CGAL::Barycentric_coordinates::ON_BOUNDED_SIDE;

    // Print some information about the polygon and coordinates.
    mean_value_coordinates.print_information();

    // Create an std::vector to store coordinates.
    Scalar_vector coordinates;

    // Compute mean value coordinates for all the interior points above.
    for (int i = 0; i < number_of_interior_points; ++i) {
        const Output_type result = mean_value_coordinates(interior_points[i],
                                                          std::back_inserter(coordinates),
                                                          query_point_location,
                                                          type_of_algorithm);

        // Output the coordinates b for each point p.
        const string computation_status = (result ? "success." : "failure.");
        cout << "\nFor p_" << i + 1 << " status: " << computation_status << endl;

        for (int j = 0; j < number_of_vertices; ++j)
            cout << "b_" << j + 1 << " = " << coordinates[i * number_of_vertices + j] << endl;
    }

    return EXIT_SUCCESS;
}
```

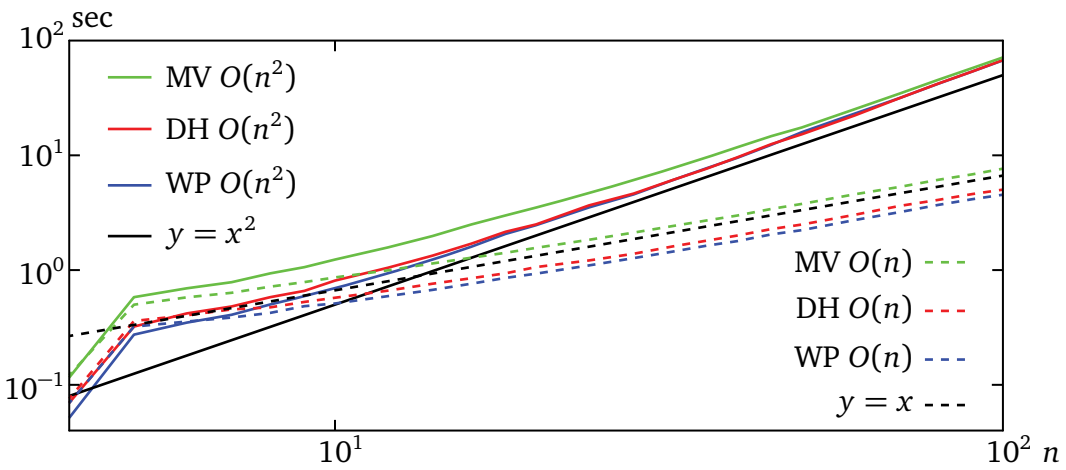


Figure 3.11. Log-log plot: Time in seconds to evaluate n coordinate functions for a polygon with n vertices at 1 million points with the dashed $O(n)$ and the solid $O(n^2)$ algorithms for Wachspress (blue), discrete harmonic (red), and mean value (green) coordinates. The black dashed and solid lines show the linear $y = x$ and the squared $y = x^2$ trend, respectively.

3.3.3 Performance

Except for the most important requirement on barycentric coordinates to be as precise as possible, it is very important for them to be as fast as possible to compute. These coordinates are used in many applications (see Section 1.3), where they must be computed for millions of points and so their real time usage is crucial. When writing the code, we tried to fulfil this important requirement, and in this section we present a few results about the computation times of Wachspress, discrete harmonic, and mean value coordinates.

The structure of the performance experiment that we run for all coordinate functions consists of computing coordinate values at ≥ 1 million strictly interior points with respect to a given polygon. At each iteration of the loop, we create a query point, pass it to the function, and compute n coordinates using the fast $O(n)$ or the precise $O(n^2)$ algorithm. We perform 100 trials of this experiment, and the time presented in the log-log plot in Figure 3.11 is the arithmetic mean of all trials.

The time to compute coordinates depends on many factors such as memory allocation, input kernel, output container, number of points, etc. In our experiments, we use the most standard C++ and CGAL features with minimum memory allocation. Therefore, the final time is the average time that can be ex-

pected without deep optimization but still with efficient memory allocation. It also means that it may vary depending on the usage of the package.

For all tests we use a MacBook Pro 2011 with 2 GHz Intel Core i7 processor (2 cores) and 8 GB 1333 MHz DDR3 memory. The installed operating system is Mac OS X 10.9 Maverick. In order to compile the suite with all performance experiments, we use the Clang 5.0 64 bit compiler.

From Figure 3.11 it is easy to see that for a small n, the precise $O(n^2)$ algorithm is as fast as the more efficient $O(n)$ algorithm but, when we increase n, the linear algorithm outperforms the squared one, as expected. One of the reasons for this to happen is because for a small n the operation of multiplication over $n-2$ elements inside the $O(n^2)$ algorithm, when computing generalized barycentric weights, takes almost the same time as the corresponding operation of division in the $O(n)$ algorithm. For a polygon with many vertices, this multiplication is much slower.

Chapter 4

A convex combination of Wachspress and mean value coordinates

Since Wachspress coordinates (see Sections 2.1.1 and 3.2.2) are rational polynomials that are well-defined over the domain Ω_ϵ (see the proof of Theorem 2) for a strictly convex polygon, they are C^∞ at the vertices of this polygon. In fact, they are C^∞ at the vertices of any simple polygon as long as the denominator W in (3.3) does not vanish at these vertices. However, since this denominator may vanish at some points inside concave polygons, Wachspress coordinates are not well-defined for such polygons. On the contrary, mean value coordinates (see Section 2.1.3) are only C^0 at the polygon's vertices, but they are well-defined for all $\boldsymbol{p} \in \mathbb{R}^2$ and for all simple polygons. These particular properties suggest that new closed-form generalized barycentric coordinates for any simple polygon, which are C^∞ at the polygon's vertices and well-defined for all $\boldsymbol{p} \in \mathbb{R}^2$, can be obtained as a convex combination of Wachspress and mean value coordinates. Indeed, it turns out that such coordinates have a rather simple formulation and we discuss it in Section 4.1. We then compare these coordinates with some other generalized barycentric coordinates in Section 4.2 and conclude with a few remarks on the proposed approach in Section 4.3.

4.1 Construction

Let P be an arbitrary simple polygon in the plane with n vertices $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ (see Figure 4.1). We denote by

$$\boldsymbol{b}_{\text{WP}} = [b_1^{\text{WP}}, \ldots, b_n^{\text{WP}}] \colon \mathbb{R}^2 \to \mathbb{R}^n,$$
$$\boldsymbol{b}_{\text{MV}} = [b_1^{\text{MV}}, \ldots, b_n^{\text{MV}}] \colon \mathbb{R}^2 \to \mathbb{R}^n$$

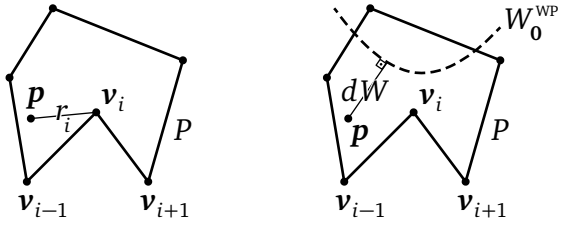


Figure 4.1. Notation used for generalized barycentric coordinates in Section 4.1.

the Wachspress and the mean value coordinates with respect to P. In particular, we use the closed-form formulation (1.12) for both coordinate functions,

$$b_i^{\text{WP}} = \frac{w_i^{\text{WP}}}{W^{\text{WP}}}, \qquad b_i^{\text{MV}} = \frac{w_i^{\text{MV}}}{W^{\text{MV}}}, \qquad i = 1, \dots, n.$$

We also consider the blending function

$$\mu \colon \mathbb{R}^2 \to [0, 1]$$

such that

$$\mu(\boldsymbol{p}) = \begin{cases} 0, & \forall\ \boldsymbol{p} \in \mathbb{R}^2 : W^{\text{WP}}(\boldsymbol{p}) = 0, \\ 1, & \forall\ \boldsymbol{p} \in \mathbb{R}^2 : r_i(\boldsymbol{p}) = 0 \text{ and } W^{\text{WP}}(\boldsymbol{p}) \neq 0, \qquad i = 1, \dots, n, \end{cases} \tag{4.1}$$

where $r_i(\boldsymbol{p}) = \|\boldsymbol{p} - \boldsymbol{v}_i\|$ and the zero level set $W_{\mathbf{0}}^{\text{WP}} = \{\boldsymbol{p} \in \mathbb{R}^2 : W^{\text{WP}}(\boldsymbol{p}) = 0\}$ of the Wachspress denominator W^{WP} is shown in Figure 4.1. Using the Wachspress and the mean value coordinates $\boldsymbol{b}_{\text{WP}}$ and $\boldsymbol{b}_{\text{MV}}$, we finally define new coordinate functions $\boldsymbol{b} = [b_1, \dots, b_n] \colon \mathbb{R}^2 \to \mathbb{R}^n$ (see Figure 4.2) as a convex combination

$$b_i = \mu b_i^{\text{WP}} + (1 - \mu) b_i^{\text{MV}}, \qquad i = 1, \dots, n, \tag{4.2}$$

where the blending function μ is such that the coordinates b_i are the mean value coordinates for all the points from the set $W_{\mathbf{0}}^{\text{WP}}$ and are the Wachspress coordinates at the polygon's vertices if the zero level curve $W_{\mathbf{0}}^{\text{WP}}$ does not pass through these vertices. The coordinates b_i satisfy the partition of unity property (1.8), because for a point $\boldsymbol{p} \in \mathbb{R}^2$ we have

$$\begin{aligned} \sum_{i=1}^{n} b_i(\boldsymbol{p}) &= \mu(\boldsymbol{p}) \sum_{i=1}^{n} b_i^{\text{WP}}(\boldsymbol{p}) + (1 - \mu(\boldsymbol{p})) \sum_{i=1}^{n} b_i^{\text{MV}}(\boldsymbol{p}) \\ &= \mu(\boldsymbol{p}) + 1 - \mu(\boldsymbol{p}) = 1. \end{aligned}$$

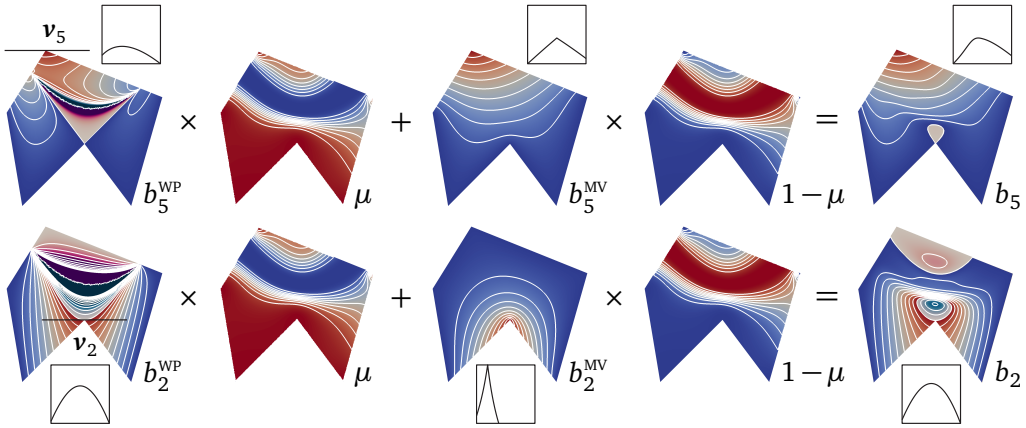


Figure 4.2. The coordinate functions b_2 and b_5 for the concave $\boldsymbol{v}_2$ (bottom row) and the convex $\boldsymbol{v}_5$ (top row) vertices are defined by multiplying the Wachspress coordinates b_2^{WP} and b_5^{WP} with the blending function μ and then adding to the result of the multiplication of the mean value coordinates b_2^{MV} and b_5^{MV} with the blending function $1-\mu$. The magenta, blue/red, and cyan colour ranges show where the coordinates are < 0, belong to the interval $[0,1]$, and > 1, respectively. The white curves are the contour lines with the 0.1 interval. The inset shows the cross sections of all coordinates along the horizontal lines through $\boldsymbol{v}_2$ and $\boldsymbol{v}_5$ (see left column).

and the linear reproduction property (1.9), because

$$\begin{aligned}\sum_{i=1}^{n} b_i(\boldsymbol{p})\boldsymbol{v}_i &= \mu(\boldsymbol{p})\sum_{i=1}^{n} b_i^{\text{WP}}(\boldsymbol{p})\boldsymbol{v}_i + (1-\mu(\boldsymbol{p}))\sum_{i=1}^{n} b_i^{\text{MV}}(\boldsymbol{p})\boldsymbol{v}_i \\ &= \mu(\boldsymbol{p})\boldsymbol{p} + \boldsymbol{p} - \mu(\boldsymbol{p})\boldsymbol{p} = \boldsymbol{p}.\end{aligned}$$

Since both Wachspress and mean value coordinates can be negative inside concave polygons and $\mu \in [0,1]$, the coordinate functions b_i can also be negative inside these polygons. It further follows directly from the definition that the b_i satisfy the Lagrange property (1.10b) and are linear along the polygon's edges. Finally, due to the construction of the blending function μ, the coordinates b_i are C^∞ at the vertex $\boldsymbol{v}_j$ as long as the zero level curve $W_{\boldsymbol{0}}^{\text{WP}}$ does not pass through this vertex, which can happen only for polygons with three or more collinear vertices (see Figure 4.3). In this case, the b_i at $\boldsymbol{v}_j$ are C^0 as mean value coordinates. For all other $\boldsymbol{p} \in \mathbb{R}^2 \setminus \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\}$, since Wachspress and mean value coordinates are both C^∞, the b_i are as smooth as the blending function μ.

To this end, we still have to show how to construct a blending function μ that satisfies the conditions (4.1). Taubin [1994] proposes an efficient approach for

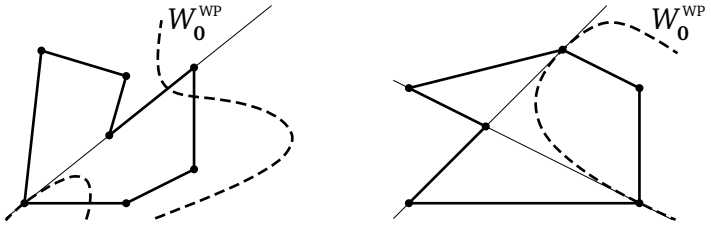


Figure 4.3. For polygons with three or more collinear vertices the zero level curve W_0^{WP} passes through some vertices of the polygon.

approximating the Euclidean distance dW from $\boldsymbol{p}$ to the implicit curve W_0^{WP} (see Figure 4.1) as

$$dW(\boldsymbol{p}) = \frac{|W^{\text{WP}}(\boldsymbol{p})|}{\|\nabla W^{\text{WP}}(\boldsymbol{p})\|}.$$

We further define the position of $\boldsymbol{p}$ with respect to the polygon's vertices as

$$dV(\boldsymbol{p}) = \prod_{i=1}^{n} r_i(\boldsymbol{p}),$$

where $r_i(\boldsymbol{p}) = \|\boldsymbol{p} - \boldsymbol{v}_i\|$ (see Figure 4.1). The blending function μ can then be constructed as

$$\mu = \begin{cases} \frac{dW^4}{dV^2+dW^4}, & dV^2 + dW^4 \neq 0, \\ 0, & dV^2 + dW^4 = 0. \end{cases} \tag{4.3}$$

On the one hand, if $W^{\text{WP}}(\boldsymbol{p}) = 0$ for some $\boldsymbol{p} \in \mathbb{R}^2$, then it follows that $dW(\boldsymbol{p}) = 0$ and hence $\mu(\boldsymbol{p}) = 0$. On the other hand, if $r_i(\boldsymbol{p}) = 0$ for any index $i = 1, \ldots, n$, then it follows that $dV(\boldsymbol{p}) = 0$ and hence $\mu(\boldsymbol{p}) = 1$. We also note that the denominator $dV^2(\boldsymbol{p}) + dW^4(\boldsymbol{p}) = 0$ if and only if both $dV(\boldsymbol{p}) = 0$ and $dW(\boldsymbol{p}) = 0$, which can happen only for polygons with three and more collinear vertices (see Figure 4.3). It is also clear from the definition (4.3) that the function μ is a smooth function and satisfies conditions (4.1).

4.2 Comparison

Figures 4.4 and 4.5 show a comparison of our (WM) coordinates with Wachspress (WP) and mean value (MV) coordinates for two different polygons and coordinate functions associated to convex and concave vertices. As expected, for the convex polygon all coordinates are well-defined and non-negative, but this is different for the concave polygon, where Wachspress coordinates have poles. Instead, our coordinates are well-defined for such polygons as well as mean value

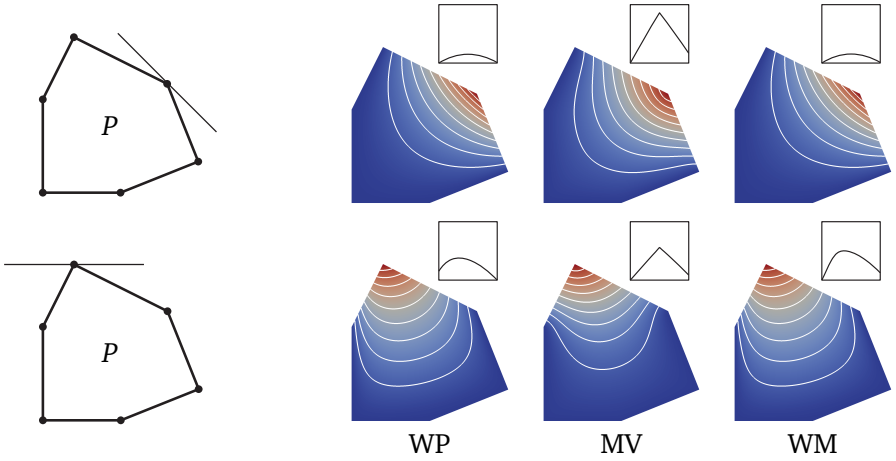


Figure 4.4. Comparison of different barycentric coordinate functions for two different vertices with respect to a convex polygon. The colour range shows function values between 0 (blue) and 1 (red), and the white curves are the contour lines at $0.1, 0.2, \ldots, 0.9$. The inset shows the cross sections along the centre part of the lines depicted in the left column.

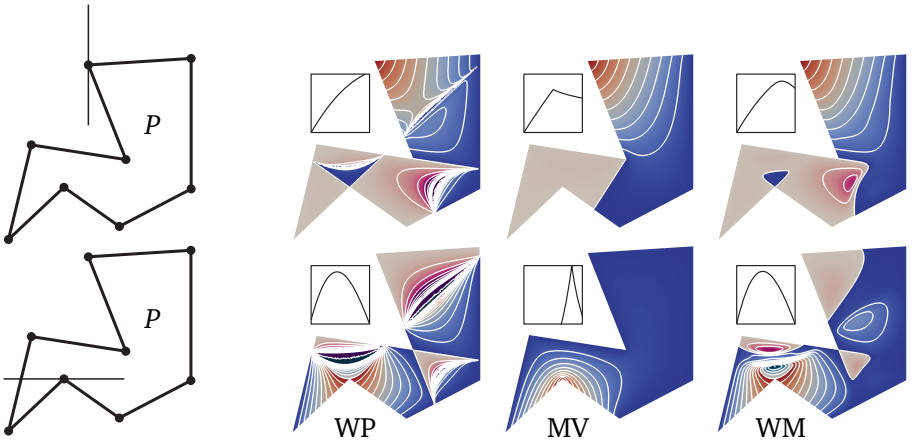


Figure 4.5. Comparison of different barycentric coordinate functions for a convex and a concave vertex with respect to a concave polygon. The magenta, blue/red, and cyan colour ranges show where the coordinates are < 0, belong to the interval $[0, 1]$, and > 1, respectively. The white curves are the contour lines with the 0.1 interval. The inset shows the cross sections along the centre part of the lines depicted in the left column.

coordinates and are C^∞ at the polygon's vertices, while mean value coordinates are only C^0 at these vertices (shown in the corresponding inset). Our coordinates are also C^∞ at the vertices of the convex polygon and are very similar to Wachspress coordinates in this case.

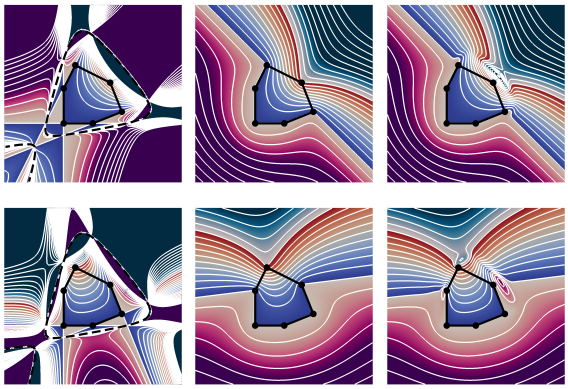


Figure 4.6. Extension of the Figure 4.4 to the polygon's exterior. The magenta, blue/red, and cyan colour ranges show where the coordinates are < 0, belong to the interval $[0,1]$, and > 1, respectively. The white curves are the contour lines with the 0.1 interval.

Even though Wachspress coordinates are well-defined and C^∞ at the vertices of a convex polygon, they may have poles outside this polygon. Instead, our coordinates are C^∞ at the polygon's vertices and are well-defined everywhere in the plane (see Figure 4.6).

We implemented all coordinates above in C++ on a MacBook Pro 2013 with 2.4 GHz Intel Core i7 processor and 8 GB 1600 MHz DDR3 memory. Since it is important for our coordinates to obtain the correct position dW of $\boldsymbol{p}$ with respect to the implicit curve $W_{\mathbf{0}}^{\text{WP}}$ and any numerical instabilities may have a negative impact on the final result, we suggest to implement these coordinates with the slower but precise $O(n^2)$ algorithm. In particular, in our implementation, we use the precise weights (3.35) for computing Wachspress coordinates b_i^{WP} and the distance dW, the precise weights (3.40) for computing mean value coordinates b_i^{MV}, and compute the function $dV = \prod_{i=1}^{n} r_i^2$ with the squared distance r_i^2 defined as in (3.33) (see Pseudocode 8 in Appendix A). For a fair comparison, we also use the $O(n^2)$ algorithms (see Pseudocodes 3 and 7 from the same appendix) to evaluate Wachspress and mean value coordinate functions.

4.3 Discussion

While convex combinations of barycentric coordinates keep their defining properties (1.8) and (1.9), the right choice of the weighting coefficients makes it possible to calibrate some other important properties of these coordinates. We showed that taking a convex combination (4.2) of Wachspress and mean value

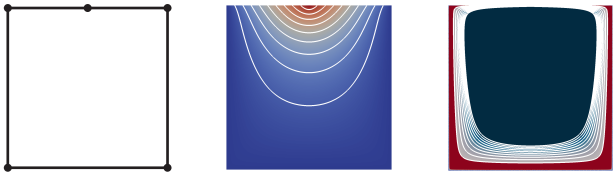


Figure 4.7. For a simple unit square P with three collinear vertices, while mean value coordinates with respect to the centre vertex are bounded between 0 (blue) and 1 (red), which is indicated by the contour lines at $0.1, \ldots, 0.9$, metric coordinates are not well-defined and it can be seen from the plot where the red colour indicates all values between 1 and 100 and the cyan range with the contour lines at $100, \ldots, 1000$ shows the function values > 100.

coordinates is a good start to obtain new coordinates with the improved properties. In particular, we construct new coordinate functions that are both smooth at the polygon's vertices as Wachspress and are well-defined everywhere in the plane and for all simple polygons as mean value coordinates.

In principle, instead of mean value coordinates one could use any other barycentric coordinates as a building block for our construction, as long as they are well-defined everywhere in $\mathbb{R}^2$ and have a simple closed form. For example, an alternative choice could be metric coordinates (see Section 2.1.5), which do satisfy these constraints. However, these coordinates are not well-defined for polygons with three consecutive collinear vertices (see Figure 4.7) and this is the reason for us to choose mean value coordinates instead.

The main limitation of our approach is that for polygons with three and more collinear vertices, not all coordinates b_i are guaranteed to be C^∞ at the polygon's vertices, because the zero level curve $W_{\mathbf{0}}^{\text{WP}}$ passes through some of these vertices (see Figure 4.3) that makes the construction only C^0 at these vertices. Moreover, our coordinates do not satisfy the non-negativity property (1.10a) (see Figures 4.2 and 4.5), because both Wachspress and mean value coordinates can be negative inside concave polygons. Therefore it still remains future work to build smooth barycentric coordinate functions that are non-negative inside an arbitrary simple polygon and have a closed form, and we do the first steps towards this goal in Chapters 5 and 6.

Chapter 5

Blended mean value coordinates

Let us recall from Section 2.1 that generalized barycentric coordinates, which satisfy the non-negativity property (1.10a) for non-convex polygons, cannot be more than C^0 at concave corners of such polygons. By sacrificing the non-negativity property, we show in Chapter 4 that smooth coordinate functions at the vertices of a simple polygon can be obtained as a convex combination or blend of Wachspress and mean value coordinates. We now show that blending mean value coordinates alone over the triangles of the *constrained Delaunay triangulation* of the input polygon with appropriate blending functions can be used to construct non-negative coordinates for any simple polygon. In addition, these coordinates are also C^k-continuous in Ω with $k > 0$ and local, where the locality is defined by the size of the *support* (the coordinate function is local if the support is small)

$$\text{Supp}(b_i) = \overline{\{\boldsymbol{p} \in \Omega \mid b_i(\boldsymbol{p}) \neq 0\}}. \tag{5.1}$$

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. We describe the blending construction in Section 5.1 and compare our coordinates with some other generalized barycentric coordinates in Section 5.2. We further present several examples of applying our coordinates to colour interpolation and image deformation in Section 5.3 and conclude with a few remarks on the proposed approach in Section 5.4.

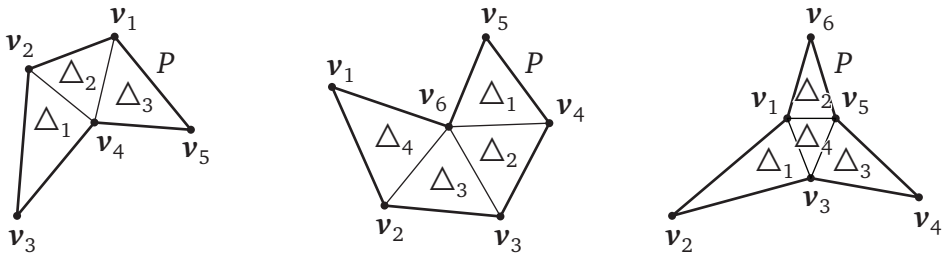


Figure 5.1. Pentagon can be split into three triangles (left), and hexagons can be split into four triangles (centre and right).

5.1 Blended mean value coordinates

It is known [Hormann and Tarini, 2004] 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 5.1.1) before extending this construction to hexagons (Section 5.1.2) and to arbitrary polygons (Section 5.1.3).

5.1.1 Coordinates for pentagons

Let $P = [\boldsymbol{v}_1, \dots, \boldsymbol{v}_5] \subset \mathbb{R}^2$ be a pentagon in the plane. Without loss of generality, we assume that it can be split into three triangles $\triangle_1 = [\boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4]$, $\triangle_2 = [\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_4]$, and $\triangle_3 = [\boldsymbol{v}_1, \boldsymbol{v}_4, \boldsymbol{v}_5]$ (see Figure 5.1, left). Alternatively, this triangulation can be seen as two overlapping quadrilaterals

$$\square_1 = \triangle_1 \cup \triangle_2 = [\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4],$$
$$\square_2 = \triangle_2 \cup \triangle_3 = [\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_4, \boldsymbol{v}_5]$$

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

$$\mu_1, \mu_2 \colon \triangle_2 \setminus \{\boldsymbol{v}_4\} \to [0, 1]$$

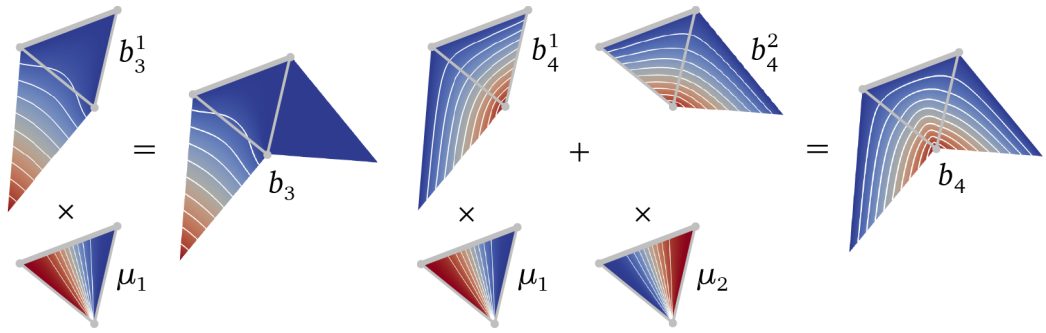


Figure 5.2. Construction of blended coordinates for the pentagon in Figure 5.1 (left). The coordinate function b_3 (left) is defined by multiplying the mean value coordinate function b_3^1 for $\square_1$ with the blending function μ_1 over $\triangle_2$ (outlined in grey). To construct b_4 (right), 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 $0.1, 0.2, \ldots, 0.9$.

such that

$$\mu_1(\boldsymbol{p}) = \begin{cases} 1, & \boldsymbol{p} \in [\boldsymbol{v}_2, \boldsymbol{v}_4), \\ 0, & \boldsymbol{p} \in (\boldsymbol{v}_4, \boldsymbol{v}_1], \end{cases} \qquad \mu_2(\boldsymbol{p}) = \begin{cases} 0, & \boldsymbol{p} \in [\boldsymbol{v}_2, \boldsymbol{v}_4), \\ 1, & \boldsymbol{p} \in (\boldsymbol{v}_4, \boldsymbol{v}_1], \end{cases}$$

and $\mu_1(\boldsymbol{p}) + \mu_2(\boldsymbol{p}) = 1$ for any $\boldsymbol{p} \in \triangle_2 \backslash \{\boldsymbol{v}_4\}$. Since μ_1 and μ_2 are not defined at $\boldsymbol{v}_4$ and actually diverge as $\boldsymbol{p} \to \boldsymbol{v}_4$, 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 unique barycentric coordinates $\lambda_{1,2} \colon \triangle_2 \to [0,1]$ (see Section 1.1) corresponding to $\boldsymbol{v}_2$ and $\boldsymbol{v}_1$ as

$$\lambda_1(\boldsymbol{p}) = \frac{\text{Area}[\boldsymbol{v}_1, \boldsymbol{p}, \boldsymbol{v}_4]}{\text{Area}[\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_4]}, \qquad \lambda_2(\boldsymbol{p}) = \frac{\text{Area}[\boldsymbol{p}, \boldsymbol{v}_2, \boldsymbol{v}_4]}{\text{Area}[\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_4]} \qquad \forall\, \boldsymbol{p} \in \triangle_2.$$

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

$$\mu_1 = \frac{\sigma_1}{\sigma_1 + \sigma_2}, \qquad \mu_2 = \frac{\sigma_2}{\sigma_1 + \sigma_2}, \qquad \sigma_1 = q \circ \lambda_1, \qquad \sigma_2 = q \circ \lambda_2. \tag{5.2}$$

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* $b_i \colon \bar{\Omega} \setminus$

$\{\boldsymbol{v}_4\} \to \mathbb{R}$, $i = 1, \dots, 5$ (see Figure 5.2) as

$$b_i(\boldsymbol{p}) = \begin{cases} b_i^1(\boldsymbol{p}), & \boldsymbol{p} \in \triangle_1, \\ b_i^1(\boldsymbol{p})\mu_1(\boldsymbol{p}) + b_i^2(\boldsymbol{p})\mu_2(\boldsymbol{p}), & \boldsymbol{p} \in \triangle_2 \setminus \{\boldsymbol{v}_4\}, \\ b_i^2(\boldsymbol{p}), & \boldsymbol{p} \in \triangle_3, \end{cases} \quad i = 1, 2, 4, \tag{5.3a}$$

$$b_3(\boldsymbol{p}) = \begin{cases} b_3^1(\boldsymbol{p}), & \boldsymbol{p} \in \triangle_1, \\ b_3^1(\boldsymbol{p})\mu_1(\boldsymbol{p}), & \boldsymbol{p} \in \triangle_2 \setminus \{\boldsymbol{v}_4\}, \\ 0, & \boldsymbol{p} \in \triangle_3, \end{cases} \tag{5.3b}$$

$$b_5(\boldsymbol{p}) = \begin{cases} 0, & \boldsymbol{p} \in \triangle_1, \\ b_5^2(\boldsymbol{p})\mu_2(\boldsymbol{p}), & \boldsymbol{p} \in \triangle_2 \setminus \{\boldsymbol{v}_4\}, \\ b_5^2(\boldsymbol{p}), & \boldsymbol{p} \in \triangle_3. \end{cases} \tag{5.3c}$$

These functions satisfy the partition of unity property (1.8), because

$$\begin{aligned} \boldsymbol{p} \in \triangle_1 \quad &\Rightarrow \quad \sum_{i=1}^{5} b_i(\boldsymbol{p}) = \sum_{i=1,2,3,4} b_i^1(\boldsymbol{p}) = 1, \\ \boldsymbol{p} \in \triangle_2 \setminus \{\boldsymbol{v}_4\} \quad &\Rightarrow \quad \sum_{i=1}^{5} b_i(\boldsymbol{p}) = \sum_{i=1,2,3,4} b_i^1(\boldsymbol{p})\mu_1(\boldsymbol{p}) + \sum_{i=1,2,4,5} b_i^2(\boldsymbol{p})\mu_2(\boldsymbol{p}) \\ &\qquad = \mu_1(\boldsymbol{p}) + \mu_2(\boldsymbol{p}) = 1, \\ \boldsymbol{p} \in \triangle_3 \quad &\Rightarrow \quad \sum_{i=1}^{5} b_i(\boldsymbol{p}) = \sum_{i=1,2,4,5} b_i^2(\boldsymbol{p}) = 1 \end{aligned}$$

and the linear reproduction property (1.9), because

$$\begin{aligned} \boldsymbol{p} \in \triangle_1 \quad &\Rightarrow \quad \sum_{i=1}^{5} b_i(\boldsymbol{p})\boldsymbol{v}_i = \sum_{i=1,2,3,4} b_i^1(\boldsymbol{p})\boldsymbol{v}_i = \boldsymbol{p}, \\ \boldsymbol{p} \in \triangle_2 \setminus \{\boldsymbol{v}_4\} \quad &\Rightarrow \quad \sum_{i=1}^{5} b_i(\boldsymbol{p})\boldsymbol{v}_i = \sum_{i=1,2,3,4} b_i^1(\boldsymbol{p})\boldsymbol{v}_i\mu_1(\boldsymbol{p}) + \sum_{i=1,2,4,5} b_i^2(\boldsymbol{p})\boldsymbol{v}_i\mu_2(\boldsymbol{p}) \\ &\qquad = \boldsymbol{p}(\mu_1(\boldsymbol{p}) + \mu_2(\boldsymbol{p})) = \boldsymbol{p}, \\ \boldsymbol{p} \in \triangle_3 \quad &\Rightarrow \quad \sum_{i=1}^{5} b_i(\boldsymbol{p})\boldsymbol{v}_i = \sum_{i=1,2,4,5} b_i^2(\boldsymbol{p})\boldsymbol{v}_i = \boldsymbol{p}. \end{aligned}$$

It further follows directly from the definition that the functions b_i satisfy the non-negativity property (1.10a) and have the Lagrange property (1.10b) at all

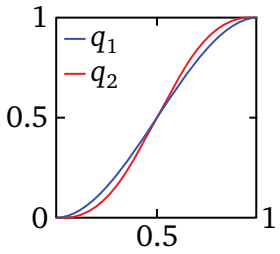


Figure 5.3. Examples of polynomial functions q_1 and q_2 used for the construction of C^1 and C^2 blended coordinates, respectively.

$\boldsymbol{v}_j$ for $j \neq 4$. To prove the Lagrange property at $\boldsymbol{v}_4$, we first observe that

$$\begin{aligned} b_i^1(\boldsymbol{p})\mu_1(\boldsymbol{p}) + b_i^2(\boldsymbol{p})\mu_2(\boldsymbol{p}) &\geq \min\{b_i^1(\boldsymbol{p}), b_i^2(\boldsymbol{p})\} \\ b_i^1(\boldsymbol{p})\mu_1(\boldsymbol{p}) + b_i^2(\boldsymbol{p})\mu_2(\boldsymbol{p}) &\leq \max\{b_i^1(\boldsymbol{p}), b_i^2(\boldsymbol{p})\} \qquad \forall\, \boldsymbol{p} \in \triangle_2 \setminus \{\boldsymbol{v}_4\} \end{aligned}$$

for $i = 1, 2, 4$, because $\mu_1(\boldsymbol{p}) + \mu_2(\boldsymbol{p}) = 1$, and

$$0 \leq b_3^1(\boldsymbol{p})\mu_1(\boldsymbol{p}) \leq b_3^1(\boldsymbol{p}), \qquad 0 \leq b_5^2(\boldsymbol{p})\mu_2(\boldsymbol{p}) \leq b_5^2(\boldsymbol{p}) \qquad \forall\, \boldsymbol{p} \in \triangle_2 \setminus \{\boldsymbol{v}_4\}$$

since $\mu_1(\boldsymbol{p}) \in [0,1]$ and $\mu_2(\boldsymbol{p}) \in [0,1]$. As all lower and upper bounds converge to $\delta_{i,4}$ as $\boldsymbol{p}$ approaches $\boldsymbol{v}_4$, we conclude that

$$\lim_{\boldsymbol{p} \to \boldsymbol{v}_4} b_i(\boldsymbol{p}) = \delta_{i,4}$$

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

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 $[\boldsymbol{v}_2, \boldsymbol{v}_4)$ and $(\boldsymbol{v}_4, \boldsymbol{v}_1]$, 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 5.3) are the polynomials

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

The function q_1 was used for the examples in Figures 5.2 and 5.4, and a comparison between C^1 and C^2 continuous coordinates, constructed with q_1 and q_2, respectively, can be found in Figures 5.6, 5.7, and 5.8.

5.1.2 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

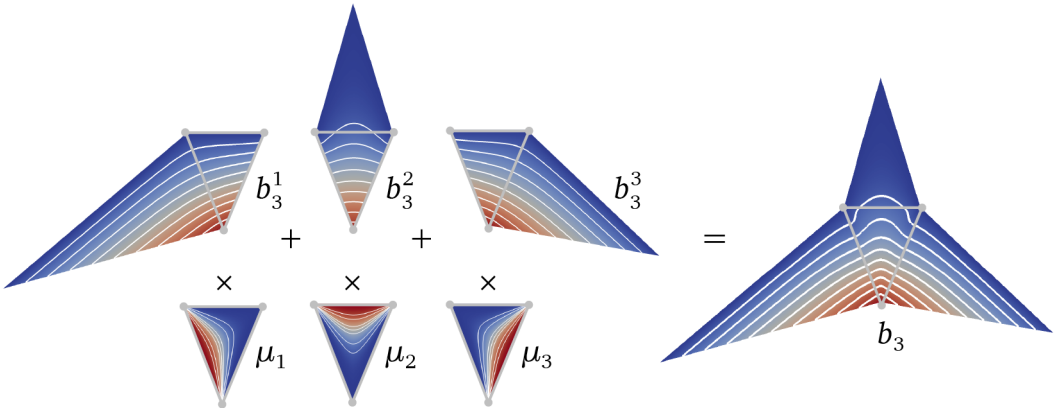


Figure 5.4. The construction of the blended coordinate function b_3 for the hexagon in Figure 5.1 (right) involves three blending functions over $\triangle_4$ (outlined in grey) for combining the quadrilateral mean value coordinates b_3^1, b_3^2, and b_3^3.

have only one or two neighbouring triangles (see Figure 5.1, centre), then the blended coordinates are constructed as described in Section 5.1.1. However, it can also happen that one of the triangles has three neighbours (see Figure 5.1, right). Given such a hexagon $P = [\boldsymbol{v}_1, \dots, \boldsymbol{v}_6] \subset \mathbb{R}^2$ in the plane, let

$$\begin{aligned}
\square_1 &= \triangle_1 \cup \triangle_4 = [\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_5], \\
\square_2 &= \triangle_2 \cup \triangle_4 = [\boldsymbol{v}_1, \boldsymbol{v}_3, \boldsymbol{v}_5, \boldsymbol{v}_6], \\
\square_3 &= \triangle_3 \cup \triangle_4 = [\boldsymbol{v}_1, \boldsymbol{v}_3, \boldsymbol{v}_4, \boldsymbol{v}_5]
\end{aligned}$$

be the quadrilaterals that overlap over the triangle $\triangle_4 = [\boldsymbol{v}_1, \boldsymbol{v}_3, \boldsymbol{v}_5]$. As in Section 5.1.1, 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

$$\mu_1, \mu_2, \mu_3 \colon \triangle_4 \setminus \{\boldsymbol{v}_1, \boldsymbol{v}_3, \boldsymbol{v}_5\} \to [0, 1],$$

such that

$$\begin{aligned}
\mu_1(\boldsymbol{p}) &= \begin{cases} 1, & \boldsymbol{p} \in (\boldsymbol{v}_1, \boldsymbol{v}_3), \\ 0, & \boldsymbol{p} \in (\boldsymbol{v}_3, \boldsymbol{v}_5) \cup (\boldsymbol{v}_5, \boldsymbol{v}_1), \end{cases} \\
\mu_2(\boldsymbol{p}) &= \begin{cases} 1, & \boldsymbol{p} \in (\boldsymbol{v}_5, \boldsymbol{v}_1), \\ 0, & \boldsymbol{p} \in (\boldsymbol{v}_1, \boldsymbol{v}_3) \cup (\boldsymbol{v}_3, \boldsymbol{v}_5), \end{cases} \\
\mu_3(\boldsymbol{p}) &= \begin{cases} 1, & \boldsymbol{p} \in (\boldsymbol{v}_3, \boldsymbol{v}_5), \\ 0, & \boldsymbol{p} \in (\boldsymbol{v}_5, \boldsymbol{v}_1) \cup (\boldsymbol{v}_1, \boldsymbol{v}_3), \end{cases}
\end{aligned}$$

and $\mu_1(\boldsymbol{p})+\mu_2(\boldsymbol{p})+\mu_3(\boldsymbol{p}) = 1$ for any $\boldsymbol{p} \in \triangle_4 \setminus \{\boldsymbol{v}_1, \boldsymbol{v}_3, \boldsymbol{v}_5\}$. 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 unique barycentric coordinates $\lambda_{1,2,3} \colon \triangle_4 \to [0,1]$ corresponding to $\boldsymbol{v}_5$, $\boldsymbol{v}_3$, and $\boldsymbol{v}_1$ as

$$\begin{aligned}
\lambda_1(\boldsymbol{p}) &= \frac{\text{Area}[\boldsymbol{v}_1, \boldsymbol{v}_3, \boldsymbol{p}]}{\text{Area}[\boldsymbol{v}_1, \boldsymbol{v}_3, \boldsymbol{v}_5]}, \\
\lambda_2(\boldsymbol{p}) &= \frac{\text{Area}[\boldsymbol{v}_1, \boldsymbol{p}, \boldsymbol{v}_5]}{\text{Area}[\boldsymbol{v}_1, \boldsymbol{v}_3, \boldsymbol{v}_5]}, \\
\lambda_3(\boldsymbol{p}) &= \frac{\text{Area}[\boldsymbol{p}, \boldsymbol{v}_3, \boldsymbol{v}_5]}{\text{Area}[\boldsymbol{v}_1, \boldsymbol{v}_3, \boldsymbol{v}_5]} \qquad \forall\, \boldsymbol{p} \in \triangle_4,
\end{aligned}$$

then we can construct the blending functions as

$$\mu_1 = \frac{\sigma_1}{\sigma_1+\sigma_2+\sigma_3}, \qquad \mu_2 = \frac{\sigma_2}{\sigma_1+\sigma_2+\sigma_3}, \qquad \mu_3 = \frac{\sigma_3}{\sigma_1+\sigma_2+\sigma_3}, \tag{5.4}$$

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 5.1.1. The construction of the blended coordinate functions $b_i \colon \bar{\Omega} \setminus \{\boldsymbol{v}_1, \boldsymbol{v}_3, \boldsymbol{v}_5\} \to \mathbb{R}$, $i = 1, \ldots, 6$ (see Figure 5.4) is then analogous to the construction (5.3), and with the same arguments as above it can be shown that these functions satisfy the key properties (1.8) and (1.9), are non-negative, have the Lagrange property, even at the vertices $\boldsymbol{v}_1$, $\boldsymbol{v}_3$, and $\boldsymbol{v}_5$, and are smooth.

5.1.3 Coordinates for arbitrary polygons

We are now ready to present the construction of blended barycentric coordinates for arbitrary simple polygons. Given the constrained Delaunay triangula-

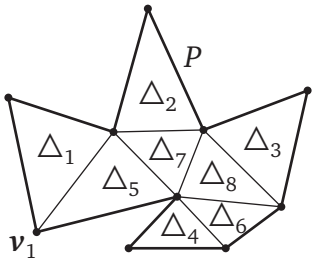


Figure 5.5. Example of a general polygon and its constrained Delaunay triangulation.

tion $\Delta = \{\triangle_1, \ldots, \triangle_m\}$ of a planar polygon $P = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n]$ 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 5.5, the quadrilateral mean value coordinates are blended over the triangles $\triangle_1, \ldots, \triangle_6$ as in Section 5.1.1 and over the triangles $\triangle_7$ and $\triangle_8$ as in Section 5.1.2. 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 (5.3), we do not present these formulas, but rather discuss how to evaluate them efficiently at any point $\boldsymbol{p}$ inside the polygon. Suppose we know the triangle $\triangle_j \in \Delta$ that contains $\boldsymbol{p}$, 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 Pseudocodes 9, 10, and 11 from Appendix 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 $\boldsymbol{p}$ is $O(1)$. On the other hand, it shows the locality of blended coordinates, because the support $\mathrm{Supp}(b_i)$

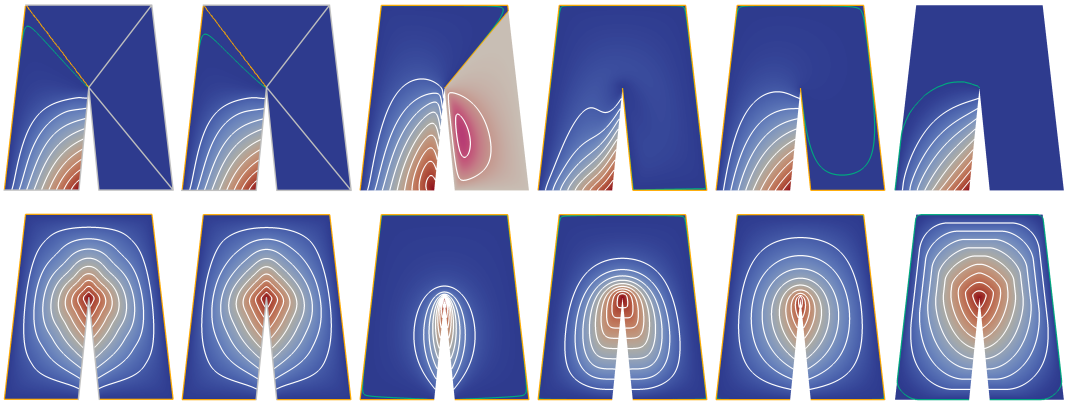


Figure 5.6. Comparison of different coordinate functions for a convex (top) and a concave (bottom) vertex. The white curves are the contour lines at $0.1, 0.2, \ldots, 0.9$ (and at -0.2 and -0.1 for mean value coordinates), the contour line at 10^{-4} is shown in green, and the orange line marks the support. The constrained Delaunay triangulation of the polygon for blended coordinates is outlined in grey, and the magenta colour range shows the negative function values for mean value coordinates.

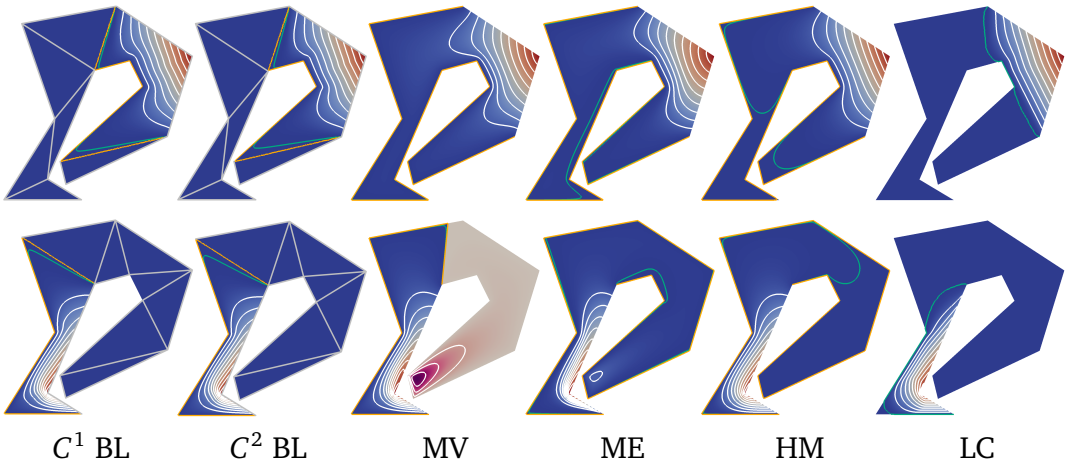


Figure 5.7. Comparison of different coordinate functions for a convex (top) and a concave (bottom) vertex. The white curves are the contour lines at $0.1, 0.2, \ldots, 0.9$ (and at $-0.3, -0.2, -0.1$ for mean value coordinates), the contour line at 10^{-4} is shown in green, and the orange line marks the support. The constrained Delaunay triangulation of the polygon for blended coordinates is outlined in grey, and the magenta colour range shows the negative function values for mean value coordinates.

of the coordinate function b_i is just the union of the triangles adjacent to $\boldsymbol{v}_i$ and their neighbouring triangles. For example, the support of b_1 in Figure 5.5 is $\mathrm{Supp}(b_1) = \triangle_1 \cup \triangle_5 \cup \triangle_7$.

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 $\boldsymbol{p}$. We shall briefly discuss three possible scenarios. To generate the images in Figures 5.6, 5.7, and 5.8, 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 bookkeeping during the subdivision process provides $\triangle_j$ for free, and the same holds in any application that allows to choose the evaluation points $\boldsymbol{p}$ per triangle of Δ. Another scenario is image deformation (see Section 2.3.2), where the coordinates need to be evaluated for each pixel of the deformed 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

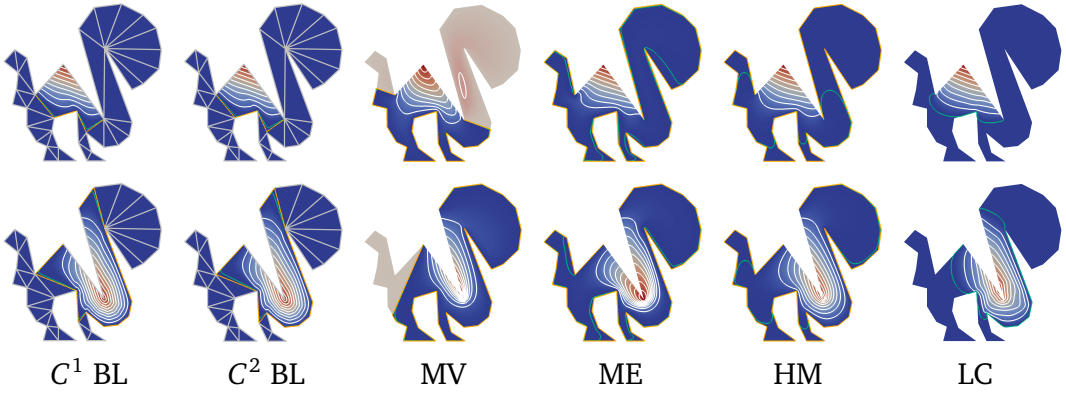


Figure 5.8. Comparison of different coordinate functions for a convex (top) and a concave (bottom) vertex. The white curves are the contour lines at $0.1, 0.2, \ldots, 0.9$ (and at -0.1 for mean value coordinates), the contour line at 10^{-4} is shown in green, and the orange line marks the support. The constrained Delaunay triangulation of the polygon for blended coordinates is outlined in grey, and the magenta colour range shows the negative function values for mean value coordinates.

breadth-first traversal. In the worst case, if the application requires to compute coordinates at truly random points $\boldsymbol{p}$, then $\triangle_j$ can be found in optimal $O(\log n)$ time [Devillers, 1998] by using a hierarchy of Delaunay triangulations, and we report some timings for this situation below.

5.2 Comparison

Figures 5.6, 5.7 and 5.8 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) coordinates for three different polygons (with 7, 13, and 39 vertices, respectively) and coordinate functions associated to convex and con-

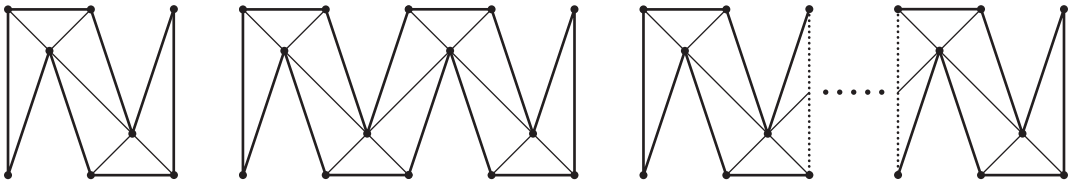


Figure 5.9. Concave test polygons, composed of 1, 2, and l pieces with 8, 14, and $n = 6l + 2$ vertices, respectively.

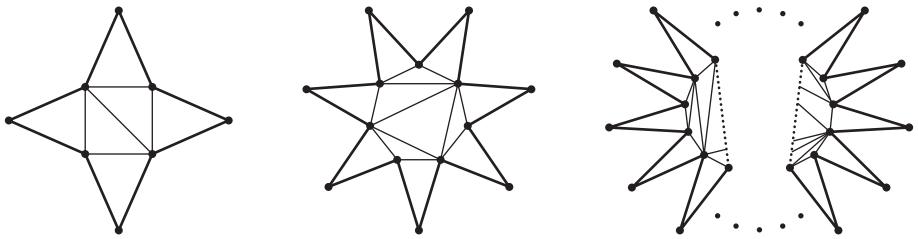


Figure 5.10. Isotoxal star polygons with 8, 14, and n vertices, respectively.

cave 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 coordinates also fulfil 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, and Zhang et al. [2014] 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 2013 with 2.4 GHz Intel Core i7 processor and 8 GB 1600 MHz DDR3 memory. For blended coordinates, we first build the constrained Delaunay triangulation of the polygon with TRIANGLE code [Shewchuk, 1996] and then evaluate the coordinate functions as explained above, using the corresponding pseudocodes from Appendix A. For mean value and maximum entropy coordinates, we implemented the pseudocodes 6 from Appendix A and [Hormann and Sukumar, 2008, Section 5], respectively. For harmonic coordinates, we use the sparse Cholesky decomposition in EIGEN [Guennebaud et al., 2010] to solve the linear system arising from the standard finite element discretization of the Laplace equation with Dirichlet boundary conditions, and local coordinates are computed with the code provided by Deng and Liu [2014].

To compare the performance, we created two sets of test polygons. The first set is shown in Figure 5.9 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 $p = 0, 1, \ldots, 4$. 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 5.10 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 Tables 5.11, 5.13 and Figures 5.12, 5.14. 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 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

	$n = 8$	$n = 14$	$n = 26$	$n = 50$	$n = 98$
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 5.11. Table with timings for polygons in Figure 5.9.

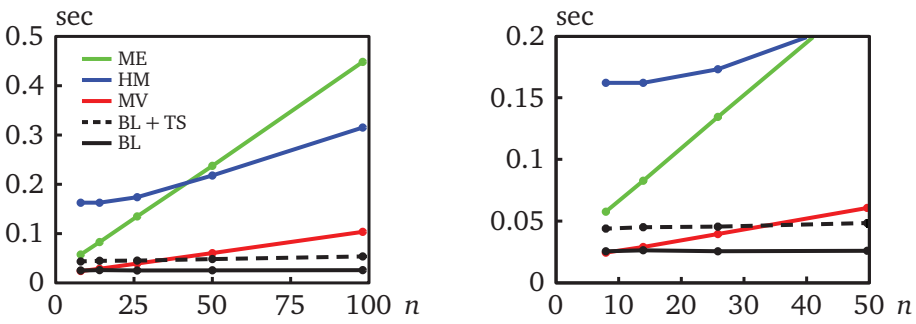


Figure 5.12. Plots with timings for polygons in Figure 5.9.

	$n=8$	$n=14$	$n=26$	$n=50$	$n=98$
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 5.13. Table with timings for polygons in Figure 5.10.

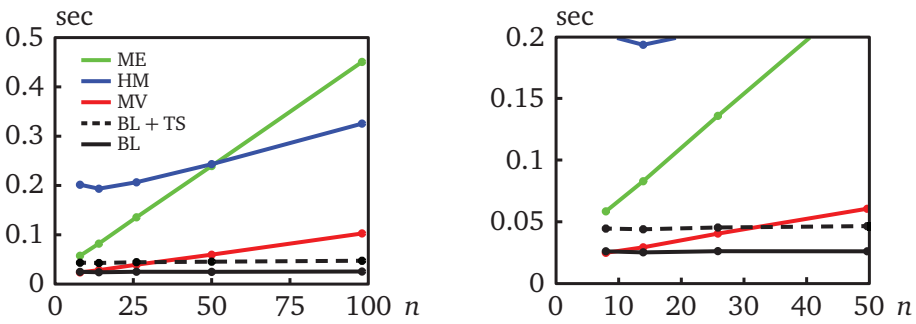


Figure 5.14. Plots with timings for polygons in Figure 5.10.

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 [Yvinec, 2016], 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 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 5.9, most of the query points lie in triangles with two neighbours, while for the polygons in Figure 5.10, 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 10 are about as expensive as executing Pseudocode 9 and Pseudocode 11 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 9 and 11 are always called similarly often for random evaluation points inside an arbitrary polygon, and the average evaluation time is therefore close to that of Pseudocode 10. Comparing the number of instructions in this routine with those for mean value coordinates further explains the break-even point at around $n = 10$.

5.3 Applications

As we discussed in Section 1.3, the main application of generalized barycentric coordinates is interpolation, and we use colour interpolation (see Section 2.3.1) and image deformation (see Section 2.3.2) as two example applications for demonstrating the behaviour of different generalized barycentric coordinates. In this section, we present several examples of applying blended coordinates to these applications and compare them to the examples based on mean value, maximum entropy, harmonic, and local coordinates.

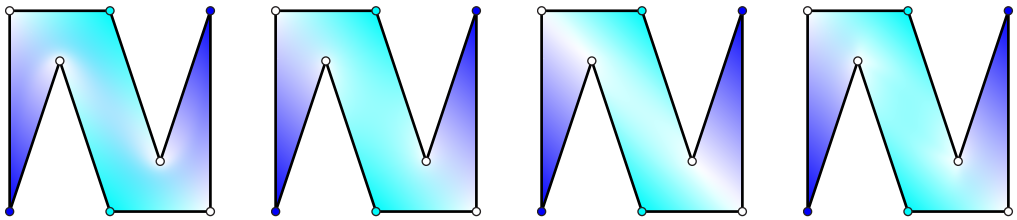


Figure 5.15. Comparison of colour interpolation with different generalized barycentric coordinates.

5.3.1 Colour interpolation

In Figure 5.15, we propagate eight RGB colours specified at the vertices of the polygon from Figure 5.9 to its interior. Since all coordinates are bounded between 0 and 1, the interpolated colour values are always in the convex hull of the given data at the polygon's vertices. Due to the global support of maximum entropy coordinates, we note that the violet colour propagates well into the centre part of the polygon. On the contrary, the locality of local coordinates leads to better separated colours. Instead, blended coordinates are less local than local coordinates for this example and result in colour interpolation similar to the one with harmonic coordinates, but they better separate the white colour and the obtained colour interpolant is C^2 continuous, while the one with harmonic coordinates is only piecewise linear. The interpolation result with C^1 blended coordinates is visually indistinguishable from the one with C^2 blended coordinates, and we omit it here.

5.3.2 Image deformation

We can also use blended coordinates to deform an image that is contained in the source polygon P with respect to a target polygon $\tilde{P}$ (see Figure 5.16). For this example, the deformation with maximum entropy, harmonic, local, and C^2 blended coordinates are quite similar, but blended coordinates are much faster to evaluate and can be used in real-time deformations even for very high-resolution images. The deformation with mean value coordinates suffers high distortions and is visually inappropriate for this example. We also note that the deformation result with C^1 blended coordinates is visually indistinguishable from the one with C^2 blended coordinates, and we omit it here.

5.4 Discussion

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 [Catmull and Rom, 1974], 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

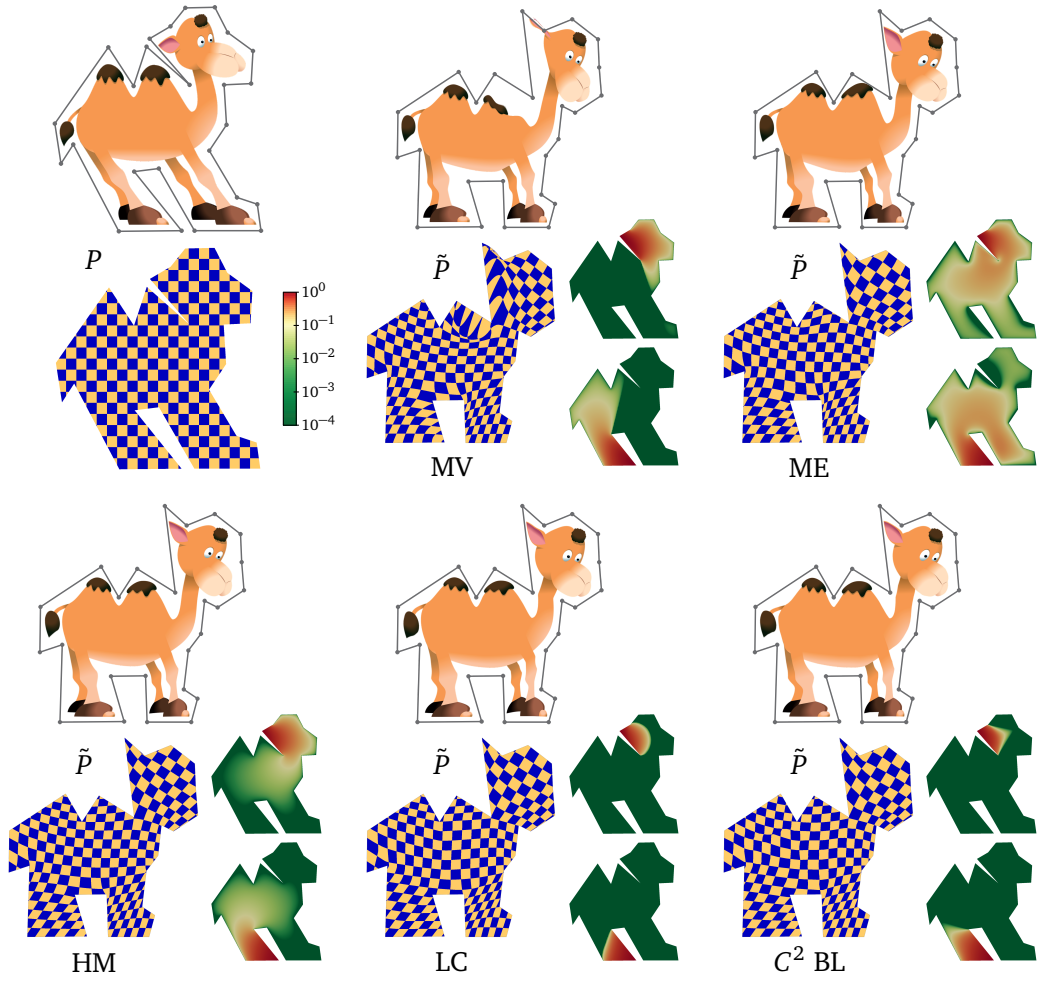


Figure 5.16. Comparison of image deformation with different generalized barycentric coordinates. The inset shows the coordinate functions for two different vertices of the polygon, where the colour scale is logarithmic, ranging from 10^{-4} to 10^{0}.

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 for 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

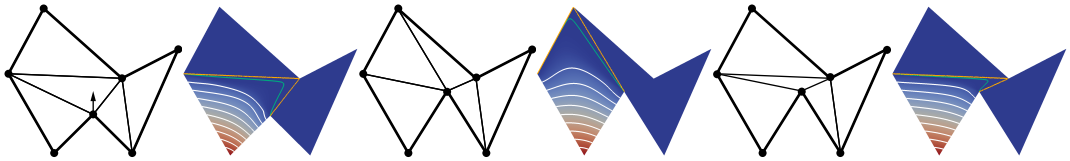


Figure 5.17. Moving a vertex of the polygon (left) can lead to a different constrained Delaunay triangulation and a discontinuous change of the coordinate functions (centre). This can be overcome by keeping the triangulation (right), as long as no triangle folds over, even if the triangulation becomes non-Delaunay.

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 $\boldsymbol{v}_i$ can lead to a different constrained Delaunay triangulation and a discontinuous change of the coordinate functions b_i (see Figure 5.17, centre). This problem can be overcome to some extent by keeping the triangulation, because the construction clearly works for non-Delaunay triangulations, too (see Figure 5.17, right), but only as long as the triangles of the triangulation do not

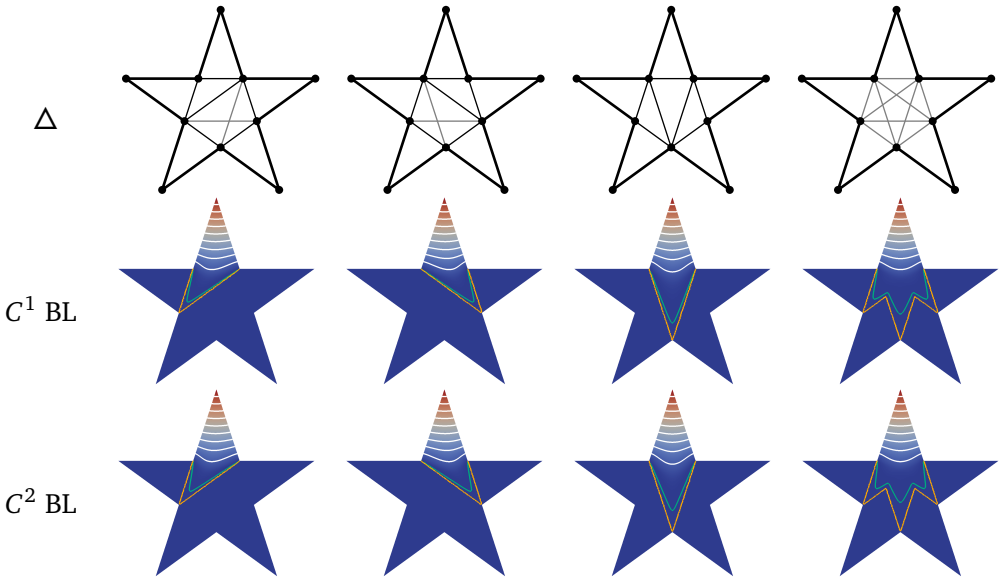


Figure 5.18. C^1 and C^2 blended coordinates computed for the top vertex with respect to different initial triangulations Δ (the first three columns) and their averaged version over all possible (shown in grey) initial triangulations (last column).

flip. But in many applications it is the data associated with the vertices of $\boldsymbol{v}_i$ 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 deformation application (see Sections 2.3.2 and 5.3.2) will not introduce any unexpected, discontinuous behaviour. Second, even for perfectly symmetric polygons, like the ones in Figure 5.10, 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 (see Figure 5.18), 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 [Schönhardt, 1928]. By reformulating this problem, we show in Chapter 6 how the coordinates with similar properties to our blended coordinates can be obtained for polygons with interior points.

Chapter 6

Subdividing barycentric coordinates

As we know from Chapter 4, convex combinations of generalized barycentric coordinates keep their key properties (1.8) and (1.9). We also know from Chapter 5 that such combinations can be used for constructing non-negative, local, and smooth coordinate functions, but these functions are not well-defined for polygons with interior points. In this chapter, we investigate the idea of combining barycentric coordinates in more detail and show how to obtain new coordinates with favourable properties both for polygons with and without interior points.

We start from presenting a novel construction that can be used for improving some properties of harmonic (see Section 2.1.8) and local (see Section 2.1.10) coordinates. The main idea is to start with a piecewise linear approximation of these coordinates over a coarse triangulation of P and then to use *subdivision* [Zorin and Schröder, 2000] to refine the coordinate functions (Section 6.1). While the coordinate functions remain piecewise linear after any finite number of subdivision steps, the refinement process gives C^1 continuous and non-negative coordinates in the limit for many common subdivision schemes, and these limit coordinates can be evaluated like maximum entropy coordinates (see Section 2.1.9) by solving a local convex optimization problem (Section 6.1.1). In particular, we focus on *Loop subdivision* [Loop, 1987] and show that the resulting C^1 limit coordinate functions combine the favourable shape properties of harmonic coordinates or the small support of local coordinates with the possibility of evaluating them and their derivatives efficiently at any $\boldsymbol{p} \in \Omega$ (Section 6.2). We further discuss briefly how to obtain similar results with *Catmull–Clark subdivision* [Catmull and Clark, 1978] (Section 6.3).

We continue by showing that Loop (Section 6.4) and Catmull–Clark (Section 6.5) subdivision can also be used for generating new barycentric coordinates in the polygon's interior starting from the polygon's tessellation without interior

vertices. This process defines non-negative, local, and C^2 coordinate functions in the polygon's interior and we apply them to colour interpolation and image deformation in Section 6.6.

We finally conclude the chapter by discussing the proposed approach and future work in Section 6.7.

6.1 Refining piecewise linear barycentric coordinates

Our main observation, which motivated us to investigate the idea of subdividing barycentric coordinates, is that affine combinations of points and barycentric coordinates commute in the following sense.

Lemma 6. *Suppose we are given m points $\boldsymbol{p}_1, \dots, \boldsymbol{p}_m \in \mathbb{R}^2$ with barycentric coordinates $\boldsymbol{b}(\boldsymbol{p}_1), \dots, \boldsymbol{b}(\boldsymbol{p}_m)$ and some weights $\omega_1, \dots, \omega_m \in \mathbb{R}$ that sum to one, $\sum_{j=1}^m \omega_j = 1$. Let $\boldsymbol{p} = \sum_{j=1}^m \omega_j \boldsymbol{p}_j \in \mathbb{R}^2$ be the point given by the affine combination of the points $\boldsymbol{p}_j$ with the weights ω_j and $\boldsymbol{b}(\boldsymbol{p}) = \sum_{j=1}^m \omega_j \boldsymbol{b}(\boldsymbol{p}_j)$ be the affine combination of the coordinates $\boldsymbol{b}(\boldsymbol{p}_j)$ with the same weights. Then $\boldsymbol{b}(\boldsymbol{p})$ are barycentric coordinates of $\boldsymbol{p}$. Moreover, if the coordinates $\boldsymbol{b}(\boldsymbol{p}_j)$ and the weights ω_j are non-negative, then so are the coordinates $\boldsymbol{b}(\boldsymbol{p})$.*

Proof. To prove the first statement, we refer to Definition 1 and observe that

$$\sum_{i=1}^n b_i(\boldsymbol{p}) = \sum_{i=1}^n \sum_{j=1}^m \omega_j b_i(\boldsymbol{p}_j) = \sum_{j=1}^m \omega_j \sum_{i=1}^n b_i(\boldsymbol{p}_j) = \sum_{j=1}^m \omega_j = 1$$

and

$$\sum_{i=1}^n b_i(\boldsymbol{p}) \boldsymbol{v}_i = \sum_{i=1}^n \sum_{j=1}^m \omega_j b_i(\boldsymbol{p}_j) \boldsymbol{v}_i = \sum_{j=1}^m \omega_j \sum_{i=1}^n b_i(\boldsymbol{p}_j) \boldsymbol{v}_i = \sum_{j=1}^m \omega_j \boldsymbol{p}_j = \boldsymbol{p}.$$

The second statement follows, because convex combinations of non-negative values are non-negative. □

Another well known fact, which turns out to be useful in this context, is that affine combinations in general, and in particular those of barycentric coordinates, commute with linear functions.

Lemma 7. *If the barycentric coordinates $\boldsymbol{b}(\boldsymbol{p}_j)$ of the points $\boldsymbol{p}_j$ in Lemma 6 lie on a common linear function, that is, $\boldsymbol{b}(\boldsymbol{p}_j) = M\boldsymbol{p}_j + \boldsymbol{a}$ for some matrix $M \in \mathbb{R}^{n \times 2}$, vector $\boldsymbol{a} \in \mathbb{R}^n$, and $j = 1, \dots, n$, then so does their affine combination $\boldsymbol{b}(\boldsymbol{p})$.*

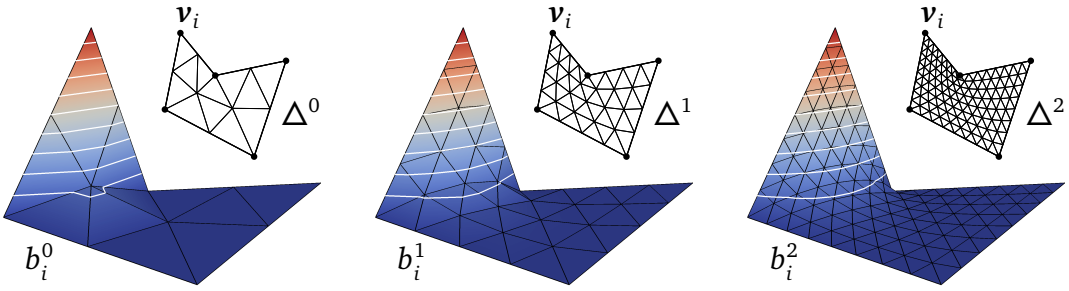


Figure 6.1. Main idea of refining piecewise linear barycentric coordinates: A triangulation Δ^0 of the polygon P is refined by a linear subdivision scheme with special rules to keep the boundary fixed. In parallel, the same subdivision rules are applied to the barycentric coordinates associated with the vertices of the triangulation, thus creating a sequence of piecewise linear barycentric coordinate functions b_i^k (shown for the vertex $\boldsymbol{v}_i$) with a C^1 continuous limit. The white curves are the contour lines at $0.1, 0.2, \ldots, 0.9$.

Proof. The statement holds, because

$$\boldsymbol{b}(\boldsymbol{p}) = \sum_{j=1}^{m} \omega_j \boldsymbol{b}(\boldsymbol{p}_j) = \sum_{j=1}^{m} \omega_j (M\boldsymbol{p}_j + \boldsymbol{a}) = M\left(\sum_{j=1}^{m} \omega_j \boldsymbol{p}_j\right) + \boldsymbol{a}\left(\sum_{j=1}^{m} \omega_j\right) = M\boldsymbol{p} + \boldsymbol{a}.$$

□

Suppose now that Δ^0 is a triangulation of P and that we are given for each vertex $\boldsymbol{p}$ of Δ^0 some initial barycentric coordinates $\boldsymbol{b}(\boldsymbol{p})$. We then consider the piecewise linear function $\boldsymbol{b}^0 = [b_1^0, \ldots, b_n^0] \colon \Delta^0 \to \mathbb{R}^n$ which interpolates the given barycentric coordinates at the vertices of Δ^0.

Corollary 1. *If the initial barycentric coordinates at the vertices $\boldsymbol{v}_i$ of P are*

$$\boldsymbol{b}(\boldsymbol{v}_i) = \boldsymbol{\delta}_i = [\delta_{1,i}, \ldots, \delta_{n,i}], \qquad i = 1, \ldots, n, \tag{6.1}$$

then the components b_i^0 of $\boldsymbol{b}^0$ are barycentric coordinate functions. If all initial barycentric coordinates are non-negative, then so are the functions b_i^0.

Proof. For any $\boldsymbol{p} \in P$, let $\triangle = [\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3]$ be the triangle in Δ^0 that contains $\boldsymbol{p}$, so that $\boldsymbol{p} = \sum_{j=1}^{3} \lambda_j \boldsymbol{p}_j$, where λ_j are the unique barycentric coordinates of $\boldsymbol{p}$ with respect to $\triangle$ (see Section 1.1). By the definition of $\boldsymbol{b}^0$ we have $\boldsymbol{b}^0(\boldsymbol{p}) = \sum_{j=1}^{3} \lambda_j \boldsymbol{b}(\boldsymbol{p}_j)$, and Lemma 6 assures not only that $\boldsymbol{b}^0(\boldsymbol{p})$ are valid barycentric coordinates of $\boldsymbol{p}$ in the sense of Definition 1 but also the statement about non-negativity (1.10a). Condition (6.1) further guarantees that b_i^0 satisfies the Lagrange property (1.10b). □

We then refine Δ^0 successively with some linear subdivision scheme S [Zorin and Schröder, 2000] to generate the sequence of triangulations $\Delta^0, \Delta^1, \ldots$ and apply the subdivision rules not only to the (x, y) coordinates of the vertices, but also to the associated barycentric coordinates. That is, if the vertex $\boldsymbol{p}$ of Δ^{k+1} is generated by the affine combination $\boldsymbol{p} = \sum_{j=1}^{m} \omega_j \boldsymbol{p}_j$ of some vertices $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m$ of Δ^k, then we associate with $\boldsymbol{p}$ the values $\boldsymbol{b}(\boldsymbol{p}) = \sum_{j=1}^{m} \omega_j \boldsymbol{b}(\boldsymbol{p}_j)$, and it follows from Lemma 6 that $\boldsymbol{b}(\boldsymbol{p})$ are valid barycentric coordinates of $\boldsymbol{p}$. As above, we now consider at each level k the piecewise linear function $\boldsymbol{b}^k = [b_1^k, \ldots, b_n^k] \colon \Delta^k \to \mathbb{R}^n$, which interpolates the generated barycentric coordinates at the vertices of Δ^k (see Figure 6.1).

Theorem 3. *Let S be a subdivision scheme that*

1. *is convergent,*
2. *generates C^1 continuous limits,*
3. *is equipped with boundary rules for interpolating corner vertices and preserving straight boundary segments.*

Further assume that the triangulations Δ^k are regular in the sense that they do not contain any degenerate or flipped triangles, even in the limit. Then the components b_i^k of $\boldsymbol{b}^k$ converge to C^1 continuous barycentric coordinate functions $b_i^\infty \colon \bar{\Omega} \to \mathbb{R}$ as $k \to \infty$. Moreover, if the initial barycentric coordinates at the vertices of Δ^0 and the weights of the subdivision rules are non-negative, then so are the b_i^∞.

Proof. Using the appropriate boundary rules along the edges of P ensures that Δ^k is a triangulation of P. Moreover, tagging the vertices of P as corners and applying to them the interpolating subdivision rule guarantees that condition (6.1) is preserved at any level k. With the same reasoning as in the proof of Corollary 1 we then conclude that the b_i^k are barycentric coordinate functions. Note that we have to assume here that Δ^k is regular, because otherwise it could happen that some $\boldsymbol{p} \in P$ is contained in more than one triangle of Δ^k and then b_i^k would not be well-defined.

To study the limit behaviour of this subdivision process, we recall that the *natural parameterization* of a subdivision surface is the one with respect to the midpoint-subdivided control mesh [Zorin and Schröder, 2000]. In our setting, this means that we use P as our domain, consider the sequence of triangulations $T^0, T^1, \ldots$, where $T^0 = \Delta^0$ and T^{k+1} is derived from T^k by midpoint subdivision, and regard Δ^k as the image of the piecewise linear function $\varphi^k \colon P \to P$ that maps from each triangle in T^k to the corresponding triangle in Δ^k (see Figure 6.2).

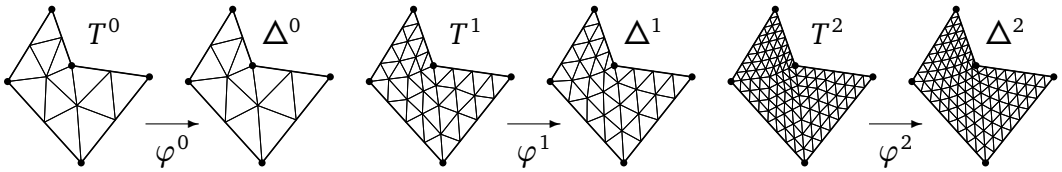


Figure 6.2. Natural parameterization φ^k of the subdivided triangulation Δ^k over the refined domain T^k for $k = 0, 1, 2$.

Under the given conditions on S, this sequence of functions converges to a C^1 continuous mapping $\bar{\varphi}\colon P \to P$. Likewise, subdividing the initial barycentric coordinates gives a C^1 continuous mapping $\bar{\boldsymbol{b}} = [\bar{b}_1, \ldots, \bar{b}_n]\colon P \to \mathbb{R}^n$ in the limit. Putting both together, we conclude that the barycentric coordinate functions b_i^k converge to the C^1 continuous functions $b_i^\infty = \bar{b}_i \circ \bar{\varphi}^{-1}$. Note that we have to assume here the regularity of Δ^k in the limit in order to ensure that $\bar{\varphi}^{-1}$ exists and is C^1 continuous, according to the inverse function theorem. The functions b_i^∞ are barycentric coordinate functions, because

$$\sum_{i=1}^{n} b_i^\infty(\boldsymbol{p}) = \sum_{i=1}^{n} \lim_{k\to\infty} b_i^k(\boldsymbol{p}) = \lim_{k\to\infty} \sum_{i=1}^{n} b_i^k(\boldsymbol{p}) = \lim_{k\to\infty} 1 = 1$$

and

$$\sum_{i=1}^{n} b_i^\infty(\boldsymbol{p})\boldsymbol{v}_i = \sum_{i=1}^{n} \lim_{k\to\infty} b_i^k(\boldsymbol{p})\boldsymbol{v}_i = \lim_{k\to\infty} \sum_{i=1}^{n} b_i^k(\boldsymbol{p})\boldsymbol{v}_i = \lim_{k\to\infty} \boldsymbol{p} = \boldsymbol{p},$$

and the statement about the non-negativity of b_i^∞ follows immediately from the given conditions. □

The conditions on S in Theorem 3 are not very restrictive and satisfied by many popular subdivision schemes [Zorin and Schröder, 2000; Cashman, 2012]. However, we recommend to use *approximating* schemes, because *interpolating* schemes, like the *butterfly scheme* [Dyn et al., 1990; Zorin et al., 1996] have subdivision rules with negative coefficients, so that the non-negativity of the limit coordinates $\boldsymbol{b}^\infty$ is not guaranteed. We further note that Corollary 1 and Theorem 3 work for any initial data, but in our examples we mainly focus on the setting where the initial piecewise linear barycentric coordinates $\boldsymbol{b}^0$ are either harmonic or local coordinates, computed for some triangulation Δ^0 of P. By construction, the initial coordinate functions b_i^0 are linear along the edges of P in these cases, and it follows from Lemma 7 and the third condition on S in Theorem 3 that the same is true for b_i^k and the limit coordinate functions b_i^∞.

6.1.1 Evaluation

For the evaluation of the limit coordinates $\boldsymbol{b}^\infty$, there are three possible scenarios. First, there are many applications, where it is sufficient to have a piecewise linear approximation of the coordinates. In this situation, we simply carry out a finite number of, say $k = 5$ or $k = 6$ subdivision steps, and take $\boldsymbol{b}^k$ as the desired piecewise linear approximation over Δ^k. We can further use the limit rules of S to snap the vertices $\boldsymbol{p}$ of Δ^k to their limit positions $\bar{\boldsymbol{p}}$, thus giving a new triangulation $\bar{\Delta}^k$. Concurrently we apply the same limit rules to the corresponding coordinates $\boldsymbol{b}^k(\boldsymbol{p})$ to compute $\boldsymbol{b}^\infty(\bar{\boldsymbol{p}})$. Overall this results in piecewise linear coordinates $\bar{\boldsymbol{b}}^k$ over $\bar{\Delta}^k$, which *interpolate* the limit coordinates at the vertices $\bar{\boldsymbol{p}}$ of $\bar{\Delta}^k$ instead of only *approximating* them.

The other two scenarios require the availability of a general routine for evaluating the limit surfaces generated by S at arbitrary parameter values, which in our setting allows to compute $\bar{\varphi}(\boldsymbol{p})$ and $\bar{\boldsymbol{b}}(\boldsymbol{p})$ at any $\boldsymbol{p} \in P$. For spline subdivision schemes which generate polynomial patches in regular regions, such a routine with constant time complexity can be designed by following the ideas of Stam [1998a,b], and non-polynomial schemes can be evaluated with the approach of Schaefer and Warren [2007, 2008]. On the one hand, we can then map any $\boldsymbol{p} \in P$ to its limit position $\bar{\boldsymbol{p}} = \bar{\varphi}(\boldsymbol{p}) \in P$ and compute the limit coordinates $\boldsymbol{b}^\infty(\bar{\boldsymbol{p}}) = \bar{\boldsymbol{b}}(\boldsymbol{p})$ of $\bar{\boldsymbol{p}}$. This is sufficient, for example, for applications which require to evaluate $\boldsymbol{b}^\infty$ at a dense set of points, but where the exact positions of the points do not matter. On the other hand, we can also determine the limit coordinates $\boldsymbol{b}^\infty(\boldsymbol{p})$ of $\boldsymbol{p} \in P$ itself by first finding $\boldsymbol{q} = \bar{\varphi}^{-1}(\boldsymbol{p})$, which in turn requires solving the local convex optimization problem

$$\min_{\boldsymbol{q} \in P} \|\boldsymbol{p} - \bar{\varphi}(\boldsymbol{q})\|^2, \tag{6.2}$$

for example with Newton's method [Nocedal and Wright, 1999]. Once $\boldsymbol{q}$ is found, we compute the limit coordinates of $\boldsymbol{p}$ as $\boldsymbol{b}^\infty(\boldsymbol{p}) = \bar{\boldsymbol{b}}(\boldsymbol{q})$.

6.1.2 Connection to standard surface subdivision

Before continuing with some concrete examples, we would like to point out a different perspective on the subdivision process described above. Suppose we attach the i-th barycentric coordinate $b_i(\boldsymbol{p})$ as a z-coordinate to each vertex $\boldsymbol{p}$ of the triangulation Δ^k. This turns Δ^k into the 3D triangle mesh G_i^k, which is nothing but the *graph* of the barycentric coordinate function b_i^k, and generating G_i^{k+1} from G_i^k is just the standard surface subdivision process. Under the given conditions on S in Theorem 3, it is then clear that the sequence of meshes $G_i^0, G_i^1, \ldots$

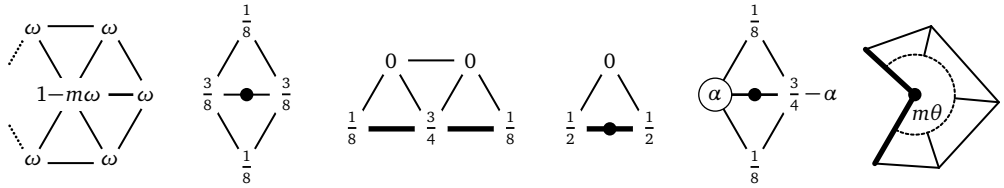


Figure 6.3. Standard Loop subdivision rules for interior vertices (a), interior edges (b), boundary vertices (c), and boundary edges (d). Corner vertices are simply interpolated. The modified edge rule (e) depends on the interior angle at the adjacent corner vertex (f).

converges to a C^1 continuous limit surface G_i^∞, and the regularity of Δ^k in the limit guarantees that G_i^∞ is the graph of a function, namely the limit coordinate function b_i^∞.

6.2 Coordinates based on Loop subdivision

In order to verify the theoretical results from the previous section, we decided to use Loop subdivision [Loop, 1987] with the modification proposed by Biermann et al. [2000]. That is, we mark vertices and edges of P as corners and creases to preserve the boundary of the polygon and use the subdivision rules in Figure 6.3, where the parameter ω for an interior vertex with valency m is $\omega = \left(\frac{5}{8} - \left(\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{m}\right)^2\right)/m$. The standard rules are used everywhere, except at interior edges adjacent to exactly one corner. For these edges, the corner is weighted by the modified coefficient $\alpha = (1 + \cos\theta)/4$, where $m\theta$ is the interior angle at the corner and m is the number of adjacent triangles.

As the subdivision rules have non-negative weights, they generate non-negative coordinate functions b_i^∞ in the limit. These rules further guarantee that the coordinate functions are C^2 almost everywhere in the interior and along the edges of P, and they are C^1 at extraordinary interior vertices with valency other than 6 and at convex corners. The b_i^∞ are only C^0 at concave corners, but this is not surprising, because non-negative coordinate functions cannot be C^1 at such corners (see Section 2.1).

6.2.1 Evaluation

To evaluate these *Loop coordinates* $\boldsymbol{b}^\infty$, we implemented the three strategies outlined in Section 6.1.1 in C++ on a MacBook Pro 2013 with 2.4 GHz Intel Core i7 processor and 8 GB 1600 MHz DDR3 memory. The first option is to subdivide the triangulation Δ^0 and the initial barycentric coordinates $\boldsymbol{b}^0$ until Δ^k has about one million vertices and to snap both the vertices $\boldsymbol{p}$ of Δ^k and the corresponding coordinates $\boldsymbol{b}^k(\boldsymbol{p})$ to the limit using the usual limit rules [Loop, 1987]. This gives a rather detailed piecewise linear interpolant of $\boldsymbol{b}^\infty$. Our implementation takes about 2 seconds for subdividing the (x, y) coordinates of the vertices of the triangulation and managing the data structures, plus another $0.02n$ seconds for subdividing the associated barycentric coordinates. By means of the tangent vector rules, we can even determine the gradients ∇b_i^∞ of the limit coordinate functions at the limit points at an additional cost of $0.2 + 0.05n$ seconds. Note that computing harmonic coordinates for a triangulation with the same number of vertices costs about $0.2n$ seconds in our implementation, which is based on EIGEN [Guennebaud et al., 2010], plus 15 seconds for assembling and factorizing the matrix. Hence, it is even a bit faster for small n, but the resulting piecewise linear coordinate functions do not interpolate the true harmonic coordinates at the vertices and gradients can only be approximated.

The second option is to evaluate for any $\boldsymbol{p} \in P$ the limit mappings $\bar{\varphi}$ and $\bar{\boldsymbol{b}}$, so as to get the limit coordinates $\boldsymbol{b}^\infty(\bar{\boldsymbol{p}}) = \bar{\boldsymbol{b}}(\boldsymbol{p})$ at $\bar{\boldsymbol{p}} = \bar{\varphi}(\boldsymbol{p})$. To this end, we first subdivide Δ^0 and $\boldsymbol{b}^0$ twice in a preprocessing step, to separate extraordinary vertices, and then find the triangle $\triangle$ in Δ^2 that contains $\boldsymbol{p}$. If $\triangle$ is not adjacent to the boundary of Δ^2, then we use Stam's algorithm [Stam, 1998b], otherwise we resort to the method of Zorin and Kristjansson [2002], which is slightly more complex but works for points near the boundary. With our implementation, which uses a quadtree to store the triangles of Δ^2, evaluating one million points this way takes about 3.2 seconds for finding the triangle $\triangle$ and computing $\bar{\boldsymbol{p}}$, plus $0.1n$ seconds for evaluating $\boldsymbol{b}^\infty(\bar{\boldsymbol{p}})$. We can further use Stam's approach to compute first derivatives of b_i^∞ at $\bar{\boldsymbol{p}}$ at roughly the same cost and even second derivatives, except when $\boldsymbol{p}$ is an extraordinary vertex of Δ^2 without well-defined second derivatives. To compute derivatives at points near the boundary, we subdivide the triangulation around $\boldsymbol{p}$ locally until the triangle that contains $\boldsymbol{p}$ is not adjacent to the boundary anymore before calling Stam's routine, because Zorin and Kristjansson do not discuss how to compute derivatives with their method. However, the cost of these local subdivisions has a negligible effect on the average runtime.

The third option is to compute $\boldsymbol{b}^\infty(\boldsymbol{p})$ for any $\boldsymbol{p} \in P$, which requires to solve the optimization problem (6.2). We implemented a simple Newton method with

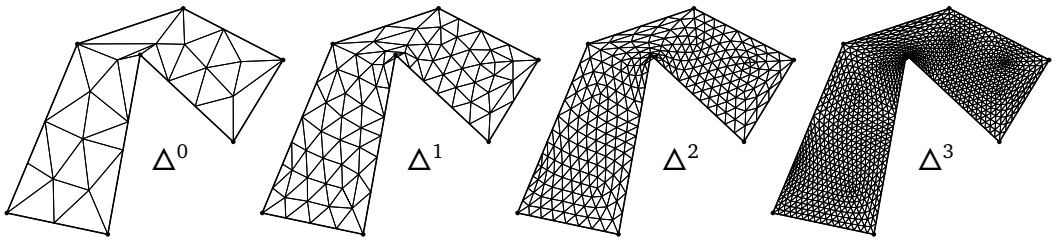


Figure 6.4. Example of an initial triangulation Δ^0 for which the subdivided triangulations Δ^k fold over at the concave corner.

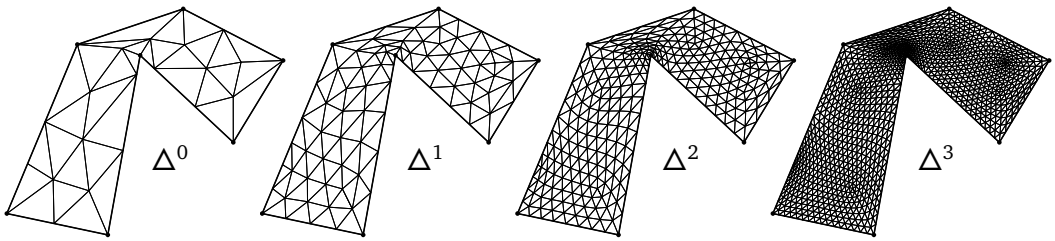


Figure 6.5. The foldovers in Figure 6.4 can be avoided by our vertex adjustment strategy (see Figure 6.6).

adaptive step size, taking advantage of the fact that we can use Stam's method as explained above to get the gradient and the Hessian of the objective function. At extraordinary vertices, where the Hessian is undefined, we resort to a finite difference approximation of the Hessian. Our experiments show that the optimal point $\boldsymbol{q} = \bar{\varphi}^{-1}(\boldsymbol{p})$ is usually found in less than three iterations with an accuracy of 10^{-7} at an average cost of $2 \cdot 10^{-6}$ seconds per point. Note that this cost does not depend on n, since it is a problem in $\mathbb{R}^2$. Once $\boldsymbol{q}$ is found, we proceed to compute $\boldsymbol{b}^\infty(\boldsymbol{p}) = \bar{\boldsymbol{b}}(\boldsymbol{q})$ as in the second option above. Overall, our implementation takes about $5+0.1n$ seconds for evaluating one million points this way. This is roughly on par with the runtime of our implementation of maximum entropy coordinates, which takes about $2+0.15n$ seconds for the same task.

While the third option is the least efficient, it is the only one that delivers the limit coordinates $\boldsymbol{b}^\infty(\boldsymbol{p})$ at an arbitrary $\boldsymbol{p} \in P$. Moreover, the additional cost with respect to the second option becomes marginal for large n, and in comparison to the first option it requires less memory, as it needs to store only the triangulation Δ^2 instead of Δ^k.

6.2.2 Concave corners

In the reasoning above, we tacitly assumed that the subdivision process gives regular triangulations Δ^k, even in the limit as $k \to \infty$. However, as noticed

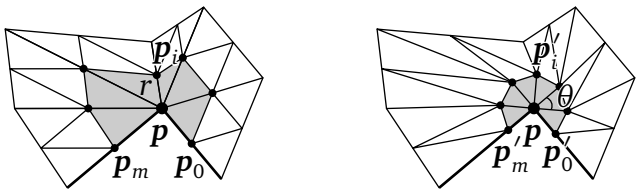


Figure 6.6. Our vertex adjustment strategy relocates the vertices in the one-ring neighbourhood of a concave corner (left) so that all adjacent triangles have the same shape and size (right).

by Biermann et al. [2000], foldovers may occur at concave corners, not only in the limit, but already after a small number of subdivision steps (see Figure 6.4). Consequently, the limit coordinates will not be well-defined in these regions. However, we can avoid this problem (see Figure 6.5) by modifying Δ^0 *before* determining the initial barycentric coordinates $\boldsymbol{b}^0$.

To this end, we adjust the positions of the vertices $\boldsymbol{p}_i$ in the one-ring neighbourhood of a concave corner $\boldsymbol{p}$ as shown in Figure 6.6. That is, we first determine the length

$$r = \min\{r_0, r_1, \ldots, r_m\},$$

where $r_i = \|\boldsymbol{p} - \boldsymbol{p}_i\|$, of the shortest edge adjacent to the concave corner. We then place all neighbours regularly spaced on a circle with radius r around $\boldsymbol{p}$,

$$\boldsymbol{p}'_0 = \boldsymbol{p} + (\boldsymbol{p}_0 - \boldsymbol{p})r/r_0, \qquad \boldsymbol{p}'_i = \boldsymbol{p} + R_{i\theta}(\boldsymbol{p}'_0 - \boldsymbol{p}), \qquad i = 1, \ldots, m, \tag{6.3}$$

where θ is defined as above and R_β denotes the rotation matrix for anticlockwise rotation by β. If this vertex adjustment strategy creates foldovers of Δ^0 in the 2-ring neighbourhood of $\boldsymbol{p}$, then we repeatedly halve r until the foldovers disappear. Note that this strategy generally requires that the one-ring neighbourhoods of the concave corners do not contain common vertices.

Theorem 4. *If the neighbours of a concave corner $\boldsymbol{p}$ have been adjusted with the strategy in (6.3), then the triangulations Δ^k are regular around $\boldsymbol{p}$, even in the limit.*

Proof. We first note that the adjusted neighbours of $\boldsymbol{p}$ satisfy

$$\boldsymbol{p}_{i\pm1} = \boldsymbol{p} + R_{\pm\theta}(\boldsymbol{p}_i - \boldsymbol{p}), \qquad i = 1, \ldots, m-1.$$

Recalling that

$$R_\theta + R_{-\theta} = 2\cos\theta\, I = (8\alpha - 2)I,$$

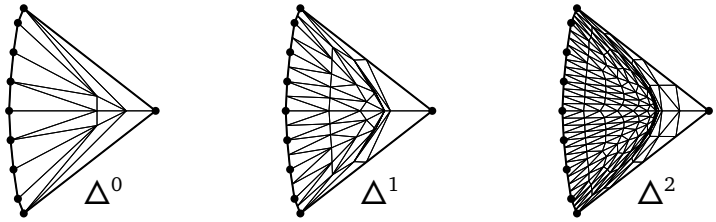


Figure 6.7. For this convex triangulation with regular interior vertices, foldovers occur in the interior after two subdivision steps.

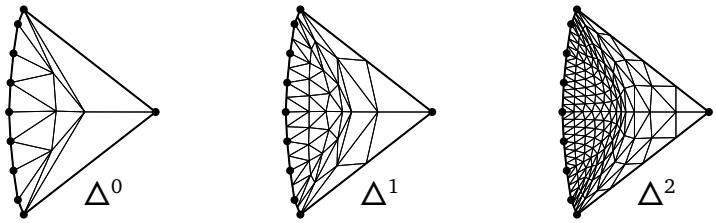


Figure 6.8. We can eliminate the foldovers in Figure 6.7 by smoothing the initial triangulation.

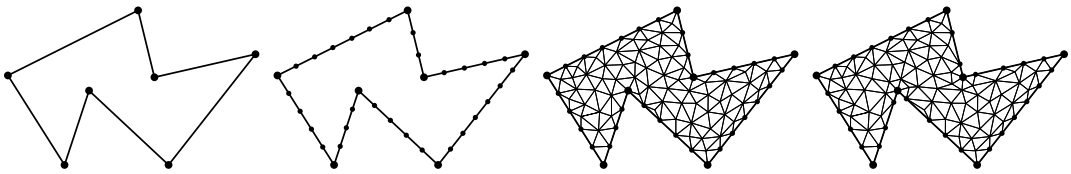
where α is defined as above and I denotes the identity matrix, we find that after one subdivision step with the modified edge rule in Figure 6.3 (e), the new interior neighbours $\tilde{\boldsymbol{p}}_i$ of $\boldsymbol{p}$ for $i = 1, \ldots, m-1$ are just the edge midpoints of the old interior edges,

$$\begin{aligned}
\tilde{\boldsymbol{p}}_i &= [8\alpha\boldsymbol{p} + (6-8\alpha)\boldsymbol{p}_i + \boldsymbol{p}_{i-1} + \boldsymbol{p}_{i+1}]/8 \\
&= [8\alpha\boldsymbol{p} + (6-8\alpha)\boldsymbol{p}_i + 2\boldsymbol{p} + (R_\theta + R_{-\theta})(\boldsymbol{p}_i - \boldsymbol{p})]/8 \\
&= [8\alpha\boldsymbol{p} + (6-8\alpha)\boldsymbol{p}_i + 2\boldsymbol{p} + (8\alpha-2)(\boldsymbol{p}_i - \boldsymbol{p})]/8 \\
&= [\boldsymbol{p} + \boldsymbol{p}_i]/2.
\end{aligned}$$

As the same holds for the new boundary neighbours $\tilde{\boldsymbol{p}}_0$ and $\tilde{\boldsymbol{p}}_m$, due to the boundary edge rule in Figure 6.3 (d), we conclude that each subdivision step simply scales the one-ring neighbourhood of $\boldsymbol{p}$ by a factor of $1/2$. This clearly avoids foldovers at $\boldsymbol{p}$, even in the limit. □

6.2.3 Internal foldovers

While our vertex adjustment strategy takes care of foldovers at concave corners, interior foldovers may still occur in the interior of Δ^k, even for convex initial triangulations Δ^0 where all interior vertices are regular with valency 6 (see Figure 6.7). A formal analysis of this problem is beyond the scope of this disser-



(a) (b) (c) (d)

Figure 6.9. To create the initial triangulation Δ^0 for a given polygon P (a), we first specify uniformly spaced vertices on the edges of P (b), then compute a constrained Delaunay triangulation (c), and finally modify the one-ring neighbourhood of each concave corner with our vertex adjustment strategy (d).

tation, but we observed that this problem does not appear if we construct the initial triangulation as shown in Figure 6.9.

Given a polygon P and a target edge length e, we first sample each edge of P with uniformly spaced vertices such that the spacing is as close as possible to e. We then use TRIANGLE [Shewchuk, 1996] to compute a *conforming constrained Delaunay triangulation* of P which contains the sample vertices, does not create any further boundary vertices, and has triangles with areas less than $A = e^2\sqrt{3}/4$, the area of the equilateral triangle with target edge length e. This usually generates a triangulation with a maximum edge length $e^\star$ close to e and not too many extraordinary vertices. In the final step, we apply the vertex adjustment strategy from Section 6.2.2 to create the initial triangulation Δ^0 of P.

To test our conjecture that the limit mapping $\bar{\varphi}$ is regular so that $\bar{\varphi}^{-1}$ exists and the limit coordinates are well-defined, we generated 100 random simple polygons and triangulated them for different values of e. We then subdivided each initial triangulation until the number of triangles was above one million and applied three tests. For each triangle with three regular vertices (usually more than 99.9% of all triangles), we checked the condition in [Ginkel et al., 2007, Lemma 3] to verify that the corresponding limit patch is regular. We further computed the limit tangents at all extraordinary vertices and checked that the limit mapping does not fold at these vertices. Both tests restrict the potential occurrence of foldovers to the triangles adjacent to extraordinary vertices and we evaluated the limit tangents at 1000 random points inside each of these triangles as explained in Section 6.2.1. All initial test triangulations passed these tests, which makes us confident that our conjecture is true.

If it is necessary to keep the original connectivity of the triangles in Figure 6.7, we observed that the internal foldovers in this triangulation can be eliminated by applying three smoothing steps with the regular vertex rule ($\omega = 1/16$ and $m = 6$) in Figure 6.3, (a) to the initial triangulation (see Figure 6.8).

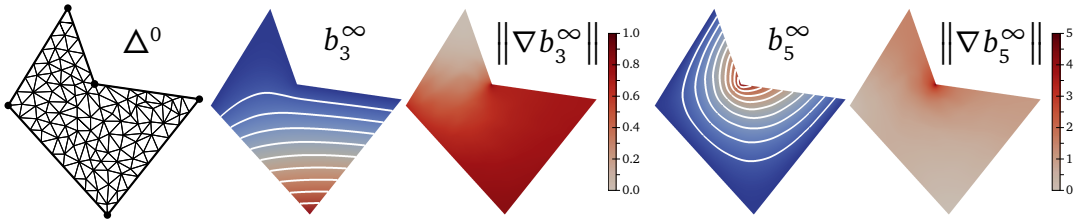


Figure 6.10. Example of Loop coordinate functions and the norm of their gradients (shown for the bottom and centre vertices) using piecewise linear harmonic coordinates over Δ^0 as initial coordinates $\boldsymbol{b}^0$.

6.2.4 Examples

Figure 6.10 shows some examples of Loop coordinate functions for harmonic initial coordinates. Despite the low resolution of the triangulation Δ^0, the functions are smooth and no visual artefacts are recognizable at the extraordinary interior vertices. Not too surprisingly, they actually look very similar to harmonic coordinates, and Figures 6.11 and 6.12 further illustrate this behaviour. In both examples, we first computed harmonic coordinates over a mesh with two million

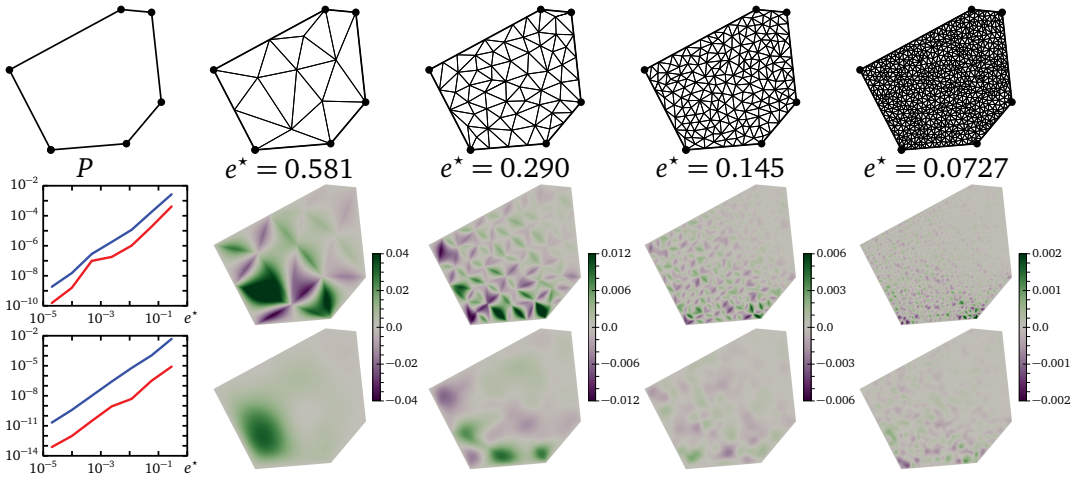


Figure 6.11. Comparison of the errors $b_i^{\mathrm{HM}} - b_i^0$ (centre) and $b_i^{\mathrm{HM}} - b_i^\infty$ (bottom) between the true harmonic coordinate function b_i^{HM}, the piecewise linear approximation b_i^0, and the Loop coordinate function b_i^∞ for the bottom vertex and different resolutions of the triangulation Δ^0 with maximum edge lengths $e^\star$. The top log-log plot shows the maximum errors $\left\|b_i^{\mathrm{HM}} - b_i^0\right\|_\infty$ (blue) and $\left\|b_i^{\mathrm{HM}} - b_i^\infty\right\|_\infty$ (red) over $e^\star$, and the bottom log-log plot shows the differences of the Dirichlet energies $D(b_i^0) - D(b_i^{\mathrm{HM}})$ (blue) and $D(b_i^\infty) - D(b_i^{\mathrm{HM}})$ (red).

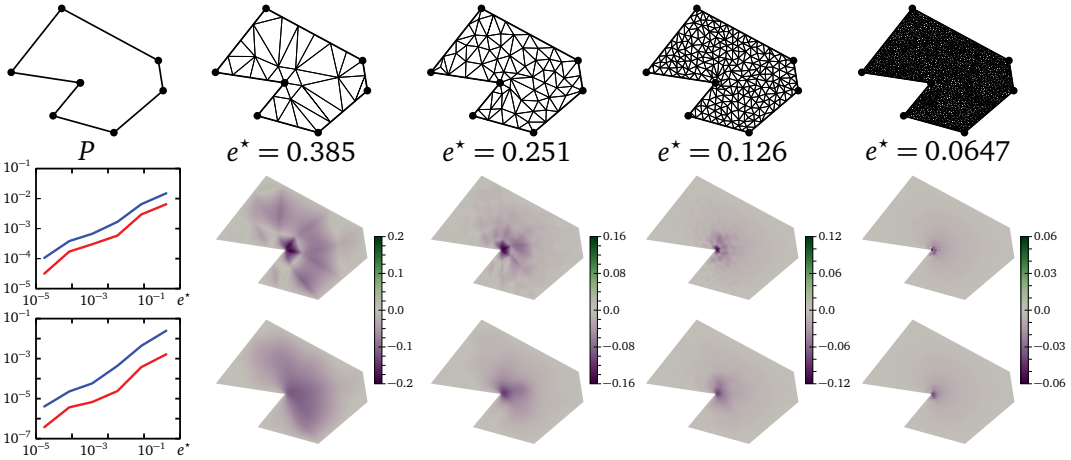


Figure 6.12. Comparison of the errors $b_i^{\text{HM}} - b_i^0$ (centre) and $b_i^{\text{HM}} - b_i^\infty$ (bottom) between the true harmonic coordinate function b_i^{HM}, the piecewise linear approximation b_i^0, and the Loop coordinate function b_i^∞ for the concave vertex and different resolutions of the triangulation Δ^0 with maximum edge lengths $e^\star$. The top log-log plot shows the maximum errors $\left\| b_i^{\text{HM}} - b_i^0 \right\|_\infty$ (blue) and $\left\| b_i^{\text{HM}} - b_i^\infty \right\|_\infty$ (red) over $e^\star$, and the bottom log-log plot shows the differences of the Dirichlet energies $D(b_i^0) - D(b_i^{\text{HM}})$ (blue) and $D(b_i^\infty) - D(b_i^{\text{HM}})$ (red).

triangles and took this approximation as referential true harmonic coordinates $\boldsymbol{b}^{\text{HM}}$. As expected, the log-log plots show that the piecewise linear harmonic coordinate functions b_i^0 over Δ^0 converge to b_i^{HM} as the maximum edge length $e^\star$ of Δ^0 tends to 0, and that the same holds for the *Dirichlet energy*

$$D(g) = \frac{1}{2} \int_P \|\nabla g\|^2, \qquad g : P \to \mathbb{R},$$

which is of course minimal for b_i^{HM}. The plots also show that the Loop coordinate functions b_i^∞ with b_i^0 as initial coordinates and their Dirichlet energies converge at the same rate and are consistently closer to b_i^{HM}. The behaviour is confirmed by the error visualizations which illustrate that Loop subdivision effectively smoothes out the error between approximate and true harmonic coordinates.

In Figure 6.13, we compare local (computed with the code provided by Deng and Liu [2014]) and the corresponding Loop coordinates for different resolutions of Δ^0. Although the theory suggests that local coordinate functions are locally supported, the numerical solver used in [Deng and Liu, 2014] generates small function values even outside the probable support and Zhang et al. [2014] suggest to consider all values below 10^{-4} as numerically zero. We modify their

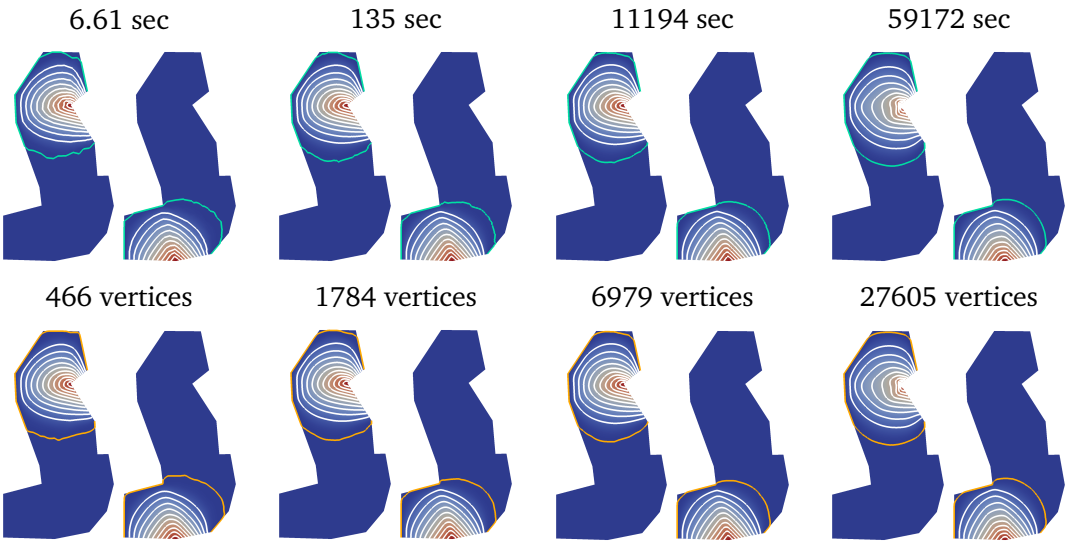


Figure 6.13. Comparison of local coordinates (top) and the corresponding Loop coordinates (bottom) for different resolutions of the initial triangulation Δ^0. For local coordinates, the contour line at 10^{-4} is shown in green, and the orange line marks the support of the Loop coordinates. The timings for computing local coordinates are given at the top.

data in the following way. For each vertex $\boldsymbol{p}$ of Δ^0 with one or more coordinates $b_i(\boldsymbol{p}) < 10^{-4}$ we set $b_i(\boldsymbol{p})$ to exact zero and perturb the other coordinates in a least squares sense to restore the key barycentric properties in Definition 1. We then use these modified coordinates as b^0. The plots show that the corresponding Loop coordinates are truly locally supported and that the support is slightly larger, but also smoother than the numerical support of the original local coordinates. We further observe that the shape of the Loop coordinates for the initial triangulations with 1784 and 6979 vertices is visually the same, which suggests that, given the exponential cost of computing local coordinates, it is better to smooth them with Loop subdivision instead of further increasing the resolution of the initial triangulation. The fact that the coordinate functions for the triangulation with 27605 vertices look apparently different from the others is due to the fact that the solver had not fully converged, even after the indicated 59172 seconds.

One potential drawback of our approach is that the Loop coordinates $\boldsymbol{b}^\infty$ depend on the initial triangulation Δ^0. An example of this effect is given in Figure 6.14, which shows two coordinate functions for two different initial triangulations as well as the difference between them. For convex corners, this

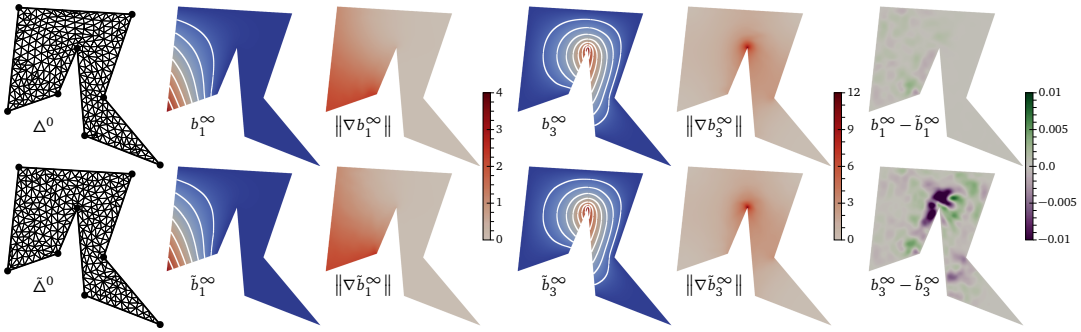


Figure 6.14. Comparison of barycentric coordinate functions for one convex and one concave vertex and different initial triangulations Δ^0 and $\tilde{\Delta}^0$ using piecewise linear harmonic coordinates as initial coordinates.

difference is usually less than 0.5% and less than 2% for concave corners, but the contour and gradient plots confirm that the global shapes of the coordinate functions are very similar.

6.3 Coordinates based on Catmull–Clark subdivision

The refinement process described in Section 6.1 works analogously with linear subdivision schemes for quadrilateral meshes. We start from an initial quadrangulation $\square^0$ of P with given barycentric coordinates at the vertices $\boldsymbol{p}$ of $\square^0$, and use the scheme to generate the sequence of quadrangulations $\square^0, \square^1, \ldots$ as well as to compute barycentric coordinates at the vertices of each $\square^k$. Under the same conditions as in Theorem 3, this gives C^1 continuous barycentric coordinate functions in the limit.

The main difference is that we need to be careful with the definition of the functions $\boldsymbol{b}^k = [b_1^k, \ldots, b_n^k] \colon \square^k \to \mathbb{R}^n$, which interpolate the initial or computed barycentric coordinates at the vertices of $\square^k$, so as to guarantee the equivalent of Corollary 1. One possibility is to split each quadrilateral of $\square^k$ into two regular triangles and let $\boldsymbol{b}^k$ be piecewise linear over the triangles obtained this way. Another choice is to let $\boldsymbol{b}^k$ be smooth over each quadrilateral by utilizing mean value coordinates (see Section 2.1.3) in the following way. For any $\boldsymbol{p} \in P$, let $\square = [\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3, \boldsymbol{p}_4]$ be the quadrilateral in $\square^k$ that contains $\boldsymbol{p}$, let $\omega_1, \ldots, \omega_4$ be the mean value coordinates of $\boldsymbol{p}$ with respect to $\square$, that is, $\boldsymbol{p} = \sum_{j=1}^4 \omega_j \boldsymbol{p}_j$, and set $\boldsymbol{b}^k(\boldsymbol{p}) = \sum_{j=1}^4 \omega_j \boldsymbol{b}^k(\boldsymbol{p}_j)$. Lemma 6 then guarantees that $\boldsymbol{b}^k(\boldsymbol{p})$ are valid barycentric coordinates of $\boldsymbol{p}$, and since the weights ω_j are non-negative, even for a concave quadrilateral $\square$ [Hormann and Tarini, 2004] (see also Chapter 5), the

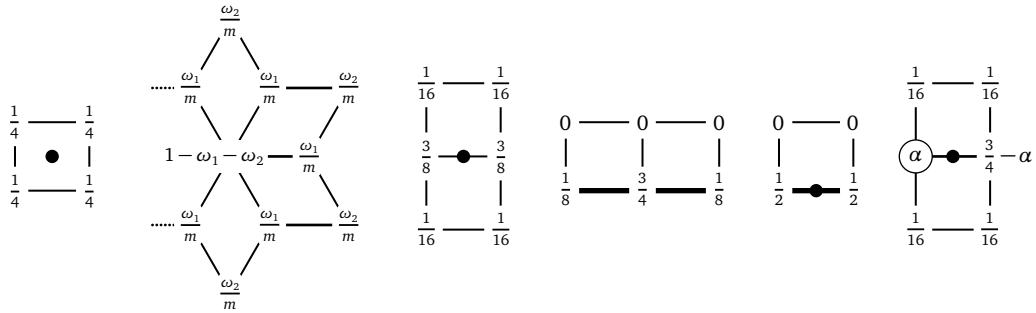


Figure 6.15. Standard Catmull–Clark subdivision rules for faces (a), interior vertices (b), interior edges (c), boundary vertices (d), boundary edges (e), and modified edge rule (f). Corner vertices are simply interpolated.

non-negativity statement in Corollary 1 carries over to the quadrilateral setting.

As a case study for this quadrilateral setting we decided to use Catmull–Clark subdivision [Catmull and Clark, 1978] with the modifications proposed by Biermann et al. [2000]. As in Section 6.2, we mark vertices and edges of P as corners and creases to preserve the boundary of the polygon and use the subdivision rules in Figure 6.15, where the parameters for an interior vertex with valency m are $\omega_1 = \frac{3}{2m}$ and $\omega_2 = \frac{1}{4m}$ and the coefficient for the modified edge rules is $\alpha = (3 + 2\cos\theta)/8$ with θ as in Section 6.2. Like Loop coordinates, these *Catmull–Clark coordinates* are C^2 almost everywhere, except at extraordinary interior vertices and convex corners, where they are only C^1 and at concave corners, where they are C^0. To evaluate them, we implemented the same three options as described in Section 6.2.1 with similar runtimes, using Stam's algorithm [Stam, 1998a] for the evaluation of $\bar{\varphi}$ and $\bar{\boldsymbol{b}}$ in the interior and the method of Zorin and Kristjansson [2002] near the boundary. The vertex adjustment around a concave corner $\boldsymbol{p}$ is done as in (6.3) for the adjacent neighbours $\boldsymbol{p}_0, \ldots, \boldsymbol{p}_m$ of $\boldsymbol{p}$, and the opposite corners $\boldsymbol{q}_0, \ldots, \boldsymbol{q}_{m-1}$ of the adjacent quadrilaterals are moved to

$$\boldsymbol{q}_i' = \boldsymbol{p}_i' + \boldsymbol{p}_{i+1}' - \boldsymbol{p}, \qquad i = 0, \ldots, m-1,$$

so that all adjacent quadrilaterals become congruent parallelograms. With the same arguments as in Theorem 4, one can then show that this configuration scales by a factor of $1/2$ with each subdivision step, thus avoiding foldovers at $\boldsymbol{p}$, even in the limit. We did not further investigate the issue of internal foldovers, but did not experience any problems in our numerical examples.

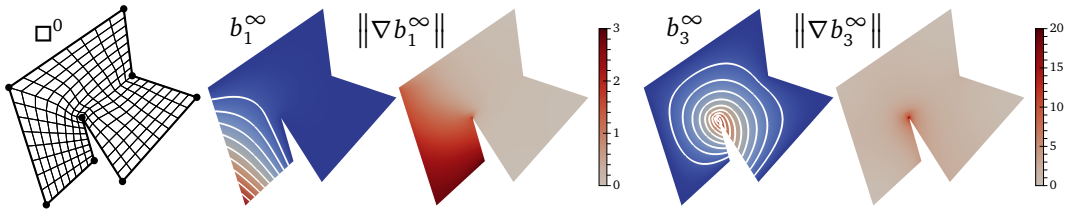


Figure 6.16. Example of Catmull–Clark coordinate functions and their gradients (shown for the bottom and middle vertices) with harmonic coordinates over $\square^0$ as initial coordinates $\boldsymbol{b}^0$.

6.3.1 Examples

Figure 6.16 is the analogue to Figure 6.10 and shows some examples of Catmull–Clark coordinate functions for harmonic initial coordinates. Since the initial quadrangulation has no extraordinary interior vertices, these functions are C^2 in the interior of P, and the plots confirm that they are also visually smooth.

In Figure 6.17, we computed harmonic coordinates over a triangulation $\triangle^0$ and a quadrangulation $\square^0$ of the same polygon and used them as initial coordinates for Loop and Catmull–Clark coordinates, respectively. The example shows that both subdivision schemes have a very similar smoothing effect and that the coordinate functions are almost identical. Since quadrangulating a given polygon is much harder than triangulating it, this suggests that Loop coordinates are probably the method of choice in most cases. However, for certain polygons like the one in Figure 6.25 it is more natural to use Catmull–Clark coordinates.

6.4 Loop coordinates

Loop coordinates can also be computed for polygons without interior points. The key advantage of this approach is that the initial triangulation $\triangle^0$ is very coarse,

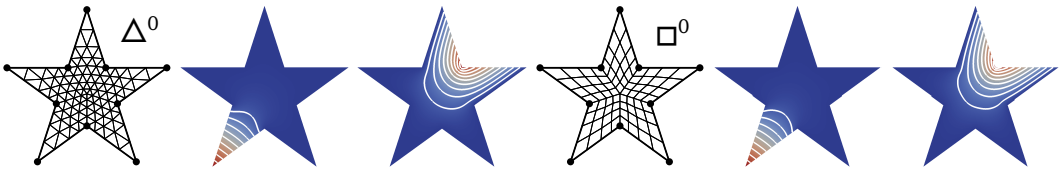


Figure 6.17. Comparison of Loop (left) and Catmull–Clark (right) coordinate functions (shown for one convex and one concave vertex) with harmonic coordinates over $\triangle^0$ and $\square^0$, respectively, as initial coordinates $\boldsymbol{b}^0$.

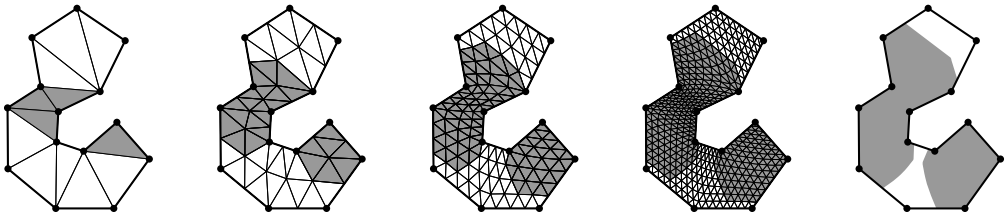


Figure 6.18. Support (shaded in grey) of the Loop coordinate functions b_i^k for one convex and one concave vertex after $k = 0, \ldots, 3$ subdivision steps and in the limit (from left to right).

including only the vertices of the given polygon. In addition, we do not need to compute harmonic, local, or any other generalized barycentric coordinates as initial coordinates $\boldsymbol{b}^0$ over Δ^0, and the resulting coordinate functions are C^2 everywhere except at the vertices of P. At convex and concave corners they are C^1 and C^0, respectively.

The locality of the subdivision rules in Figure 6.3 also implies that these coordinates are locally supported. In particular, $b_i^k(\boldsymbol{p}) = 0$ for all vertices $\boldsymbol{p}$ of Δ^k with topological edge distance $d_e(\boldsymbol{p}, \boldsymbol{v}_i) \geq 2^{k+1} - 1$, and even less in the vicinity of the other vertices $\boldsymbol{v}_j$ of P. Therefore, the functions b_i^k are local, as shown in Figure 6.18. In the limit, the support of b_i^∞ is contained in $\bar{\varphi}(N_i)$, where N_i is the union of all triangles in the two-ring neighbourhood of $\boldsymbol{v}_i$ in Δ^0.

For the example in Figure 6.19, we computed Loop coordinates for a triangu-

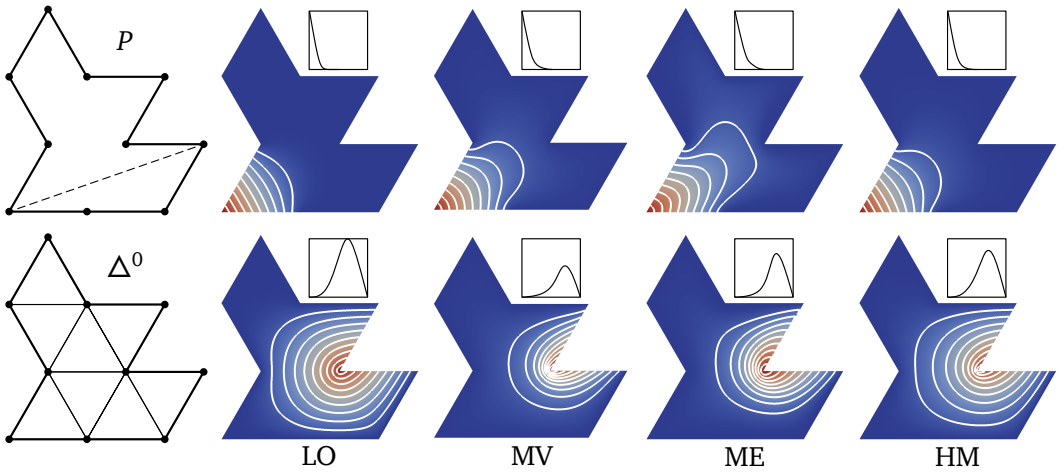


Figure 6.19. Comparison of different barycentric coordinate functions for one convex and one concave vertex. Loop coordinates were computed for the initial triangulation Δ^0 shown in the bottom left. The inset shows the cross section along the dashed line.

lation Δ^0 without interior vertices, using only the barycentric coordinates in (6.1) as initial coordinates at the vertices $\boldsymbol{v}_i$ of P. Because of the lack of extraordinary interior vertices, the resulting coordinate functions are C^2 in the interior of P. The comparison to harmonic (HM), maximum entropy (ME), and mean value coordinates (MV) shows that Loop coordinates (LO) are more local at convex and less steep at concave corners.

6.4.1 Limitations

Though Loop coordinates for polygons without interior points possess many nice properties, they have quite a few limitations. First of all, we cannot avoid foldovers at concave corners of the polygon (see Figure 6.20, left) without adding extra vertices to the initial triangulation Δ^0. To avoid such foldovers and keep the number of extra vertices to minimum, we propose two different approaches.

The first approach is to increase the valency of a concave corner $\boldsymbol{v}_i$ by adding boundary vertices to Δ^0 and connecting them with $\boldsymbol{v}_i$ (see Figure 6.21, left). As a rule of thumb, each triplet of adjacent triangles should form an angle less than π at $\boldsymbol{v}_i$. Before starting the subdivision process, we must assign initial barycentric coordinates to the new vertices. Since they lie on the edges of P, we simply interpolate the barycentric coordinates at the neighbouring corners linearly as in

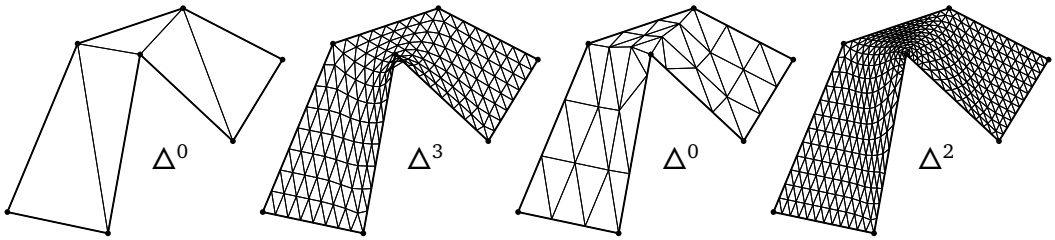


Figure 6.20. Subdividing the initial triangulation Δ^0 creates the foldover at the concave corner (left). We can fix it by applying a ternary linear subdivision step of Δ^0 with the vertex adjustment strategy from Section 6.2.2 (right).

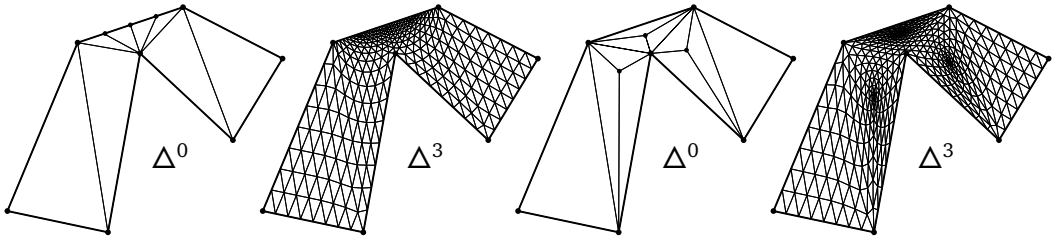


Figure 6.21. We can also fix the foldover at the concave corner in Figure 6.20 by adding extra boundary (left) or interior (right) vertices to Δ^0.

Equation (3.32), so as to maintain the linear behaviour along the edges.

The second approach is to increase the valency of concave corners by adding interior vertices to Δ^0 (see Figure 6.21, right). To compute the initial barycentric coordinates at an interior vertex $\boldsymbol{p}$, we consider the polygon $\tilde{P}$, formed by the neighbours $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m$ of $\boldsymbol{p}$, and determine the mean value coordinates (see Section 2.1.3) $\omega_1, \ldots, \omega_m$ of $\boldsymbol{p}$ with respect to $\tilde{P}$. As $\boldsymbol{p}$ lies in the kernel of $\tilde{P}$, these ω_j are positive, so that setting $\boldsymbol{b}(\boldsymbol{p}) = \sum_{j=1}^{m} \omega_j \boldsymbol{b}(\boldsymbol{p}_j)$ gives non-negative coordinates at $\boldsymbol{p}$, according to the Lemma 6.

In general, these approaches work well in practice, however they introduce irregular vertices to Δ^0, where the limit functions are only C^1, if we subdivide interior edges adjacent to irregular boundary vertices with the modified edge rule [Biermann et al., 2000] and irregular interior vertices with the appropriate rules [Loop, 1987]. They also do not guarantee the absence of foldovers in the limit, although they usually work well for up to $k = 6$ subdivision steps, which is good enough for many applications.

If we want to keep the C^2 continuity of Loop coordinates everywhere except at the polygon's vertices and avoid foldovers at concave corners also in the limit, we need to apply a ternary linear subdivision step of Δ^0 and get back to our vertex adjustment strategy in Section 6.2.2 (see Figure 6.20, right). To update the barycentric coordinates associated with the neighbours of concave corners after adjusting their positions, we sample the piecewise linear coordinate functions that we obtain after the ternary step. This strategy not only guarantees bijectiv-

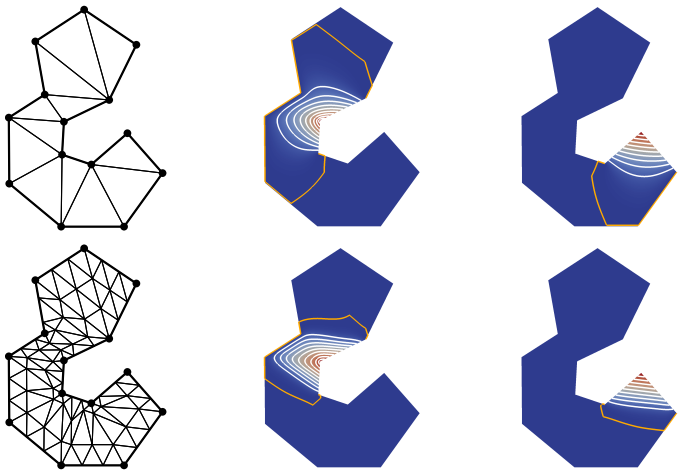


Figure 6.22. Comparison of Loop coordinate functions for one convex and one concave vertex without (top row) and with vertex adjustment (bottom row). In addition to the white contour lines at $0.1, 0.2, \ldots, 0.9$, the support is marked by the orange line.

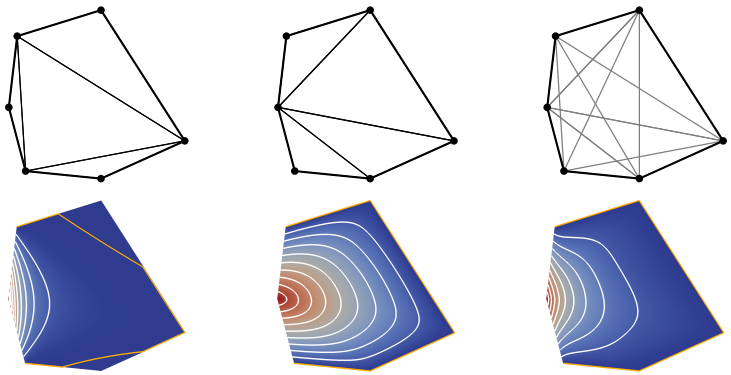


Figure 6.23. Loop coordinates for an arbitrary triangulation of P (left), for the constrained Delaunay triangulation (centre), and for the average of all possible (shown in grey) triangulations (right).

ity at concave corners, but also shrinks the support of the coordinate functions, as shown in Figure 6.22.

Another limitation of Loop coordinates is that their shape depends on the initial triangulation Δ_0 (see Figure 6.23). As thin triangles with small angles

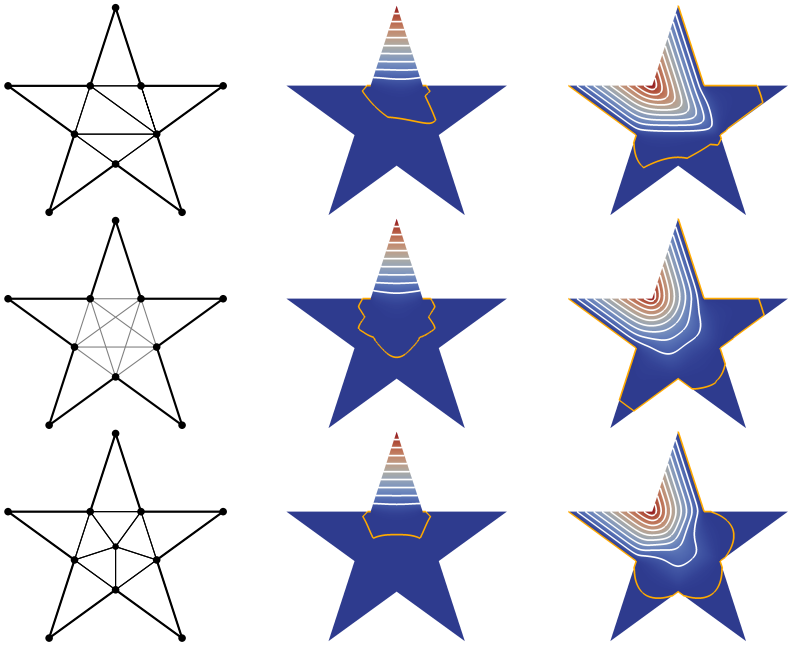


Figure 6.24. Loop coordinates for this star-shaped polygon can be symmetrized by averaging over all possible (shown in grey) initial triangulations (middle row) or by adding an extra interior vertex to the initial triangulation (bottom row).

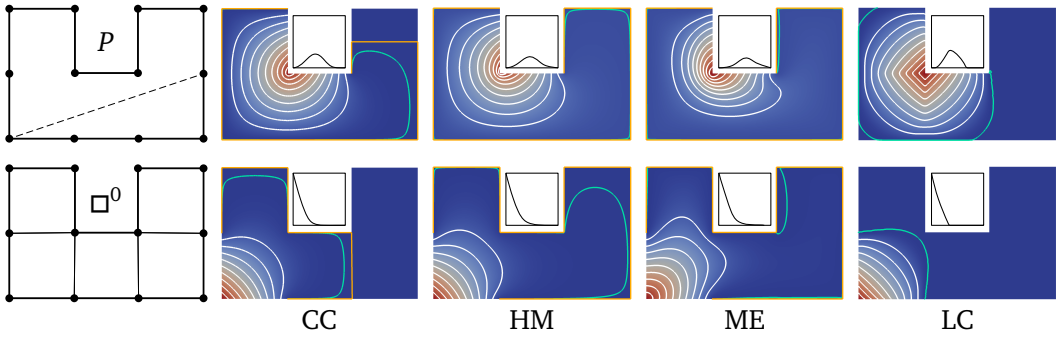


Figure 6.25. Comparison of different barycentric coordinate functions for one convex and one concave vertex. Catmull–Clark coordinates were computed for the initial quadrangulation $\square^0$ shown in the bottom left. The inset shows the cross section along the dashed line. The contour line at 10^{-4} is shown in green, and the orange line marks the support.

usually lead to coordinate functions with less natural shapes, we suggest to use the constrained Delaunay triangulation of P as default. It is also possible to avoid this dependence on Δ_0 by averaging the coordinate functions for all possible initial triangulations of P, but as the number of these triangulations grows exponentially, this approach is computationally feasible only for polygons with a small number of vertices.

Figure 6.24 shows that Loop coordinates can also be non-symmetric for symmetric polygons. This can be fixed either by the averaging procedure described above or by adding interior vertices to the initial triangulation, so as to symmetrize Δ_0 (see also Figure 6.17). However, while the first option preserves all the properties of Loop coordinates, including C^2 continuity, the second option may introduce irregular vertices, where the coordinate functions are only C^1, as explained earlier in the text.

6.5 Catmull–Clark coordinates

Similarly to Loop coordinates in Section 6.4, we can also obtain Catmull–Clark coordinates for polygons without interior points. For example, in Figure 6.25, we show such coordinates for a quadrangulation $\square^0$ without interior vertices, using only the barycentric coordinates in (6.1) as initial coordinates at the vertices $\boldsymbol{v}_i$ of P. Consequently, the resulting coordinate functions are C^2 everywhere except at $\boldsymbol{v}_i$. The comparison to harmonic (HM) and maximum entropy coordinates (ME) shows that Catmull–Clark coordinates (CC) are also more local at convex and less steep at concave corners, and they are smoother, but less local than local coordinates (LC).

Figure 6.26. Catmull–Clark coordinates for this star-shaped polygon can be symmetrized by adding an extra interior vertex to the initial quadrangulation.

These Catmull–Clark coordinates have the same limitations as Loop coordinates in Section 6.4.1 and they can be fixed similarly, for example, the symmetry of the coordinate functions for the star polygon in Figure 6.24 can be restored by adding an interior vertex to Δ^0 (see Figure 6.26). However, since it is not common to quadrangulate polygons, it is more natural to use Loop coordinates instead.

6.6 Applications

It follows from (1.10b) that the function $\boldsymbol{f}^k \colon P \to \mathbb{R}^d$ with

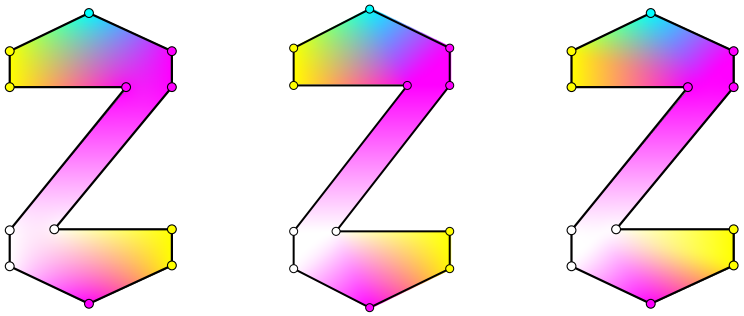
$$\boldsymbol{f}^k(\boldsymbol{p}) = \sum_{i=1}^{n} b_i^k(\boldsymbol{p}) \boldsymbol{f}_i \tag{6.4}$$

interpolates the data $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_n \in \mathbb{R}^d$ at the vertices $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ and from Definition 1 that the *Loop interpolant* $\boldsymbol{f}^k$ reproduces affine functions. Moreover, $\boldsymbol{f}^k$ converges to $\boldsymbol{f}^\infty = \bar{\boldsymbol{f}} \circ \bar{\varphi}^{-1}$, where $\bar{\boldsymbol{f}} \colon P \to \mathbb{R}^d$ is the d-dimensional Loop surface generated by subdividing the initial data $\boldsymbol{f}_i$. We now use Loop coordinates from Section 6.4 and further conclude from their properties that the Loop interpolant is linear along the edges of P, lies in the convex hull of the data $\boldsymbol{f}_i$, depends on a small subset of these data, and is C^2 continuous everywhere except at the vertices of P.

Analogously to Chapter 5, we use colour interpolation (see Section 2.3.1) and image deformation (see Section 2.3.2) as two example applications for demonstrating the behaviour of Loop coordinates.

6.6.1 Colour interpolation

Given a polygon P with RGB colour data $\boldsymbol{f}_i \in [0,1]^3$ specified at the vertices $\boldsymbol{v}_i$ of P, we want to propagate these colours to the interior of P. Note that, since the



HM LC LO

Figure 6.27. Comparison of colour interpolation with different generalized barycentric coordinates.

Loop coordinates $\boldsymbol{b}^k(\boldsymbol{q})$ are non-negative for any $\boldsymbol{q} \in P$, the interpolated RGB values are always inside the valid range $[0,1]$.

To determine the Loop interpolant at level k, it is actually not necessary to compute the n coordinate functions b_i^k by subdividing the initial data $(\boldsymbol{v}_i, \boldsymbol{\delta}_i) \in \mathbb{R}^{2+n}$ and evaluating $\boldsymbol{f}^k$ as in (6.4). Instead, we apply the subdivision process to the initial data $(\boldsymbol{v}_i, \boldsymbol{f}_i) \in \mathbb{R}^5$, which is faster, because usually $n > 3$. An argument similar to the one used in the proof of Lemma 6 then guarantees that we obtain $\boldsymbol{f}^k$ after k subdivision steps.

Figure 6.27 shows a comparison between the colour interpolants based on harmonic, local, and on Loop coordinates at level $k = 6$. All three functions give visually smooth results, but due to the global support of harmonic coordinates, the pink colour propagates well into the triangle spanned by the three white-coloured vertices. At the same time, local and Loop coordinates behave more locally and better separate regions with different colours.

6.6.2 Image deformation

Using Loop coordinates, we can also deform an image that is contained in the source polygon P by specifying a target polygon $\tilde{P}$ with vertices $\tilde{\boldsymbol{v}}_1, \ldots, \tilde{\boldsymbol{v}}_n$ and using (6.4) with the data $\boldsymbol{f}_i = \tilde{\boldsymbol{v}}_i \in \mathbb{R}^2$. After k subdivision steps, each vertex $\boldsymbol{p}$ of Δ^k has an associated vertex $\tilde{\boldsymbol{p}} = \boldsymbol{f}^k(\boldsymbol{p})$ and replacing all $\boldsymbol{p}$ with $\tilde{\boldsymbol{p}}$ defines the target mesh $\tilde{\Delta}^k$. The image is then deformed by rendering $\tilde{\Delta}^k$ using the vertices $\boldsymbol{p}$ as texture coordinates and the source image as texture (see Figure 6.28). Note that the deformed image depends smoothly on the target vertex positions, because the coordinate functions b_i^k are defined with respect to the source polygon, which remains unchanged.

Figure 6.29 shows that local and Loop coordinates lead to very similar results. In this example, the hat undergoes a rigid transformation, and the neck of the

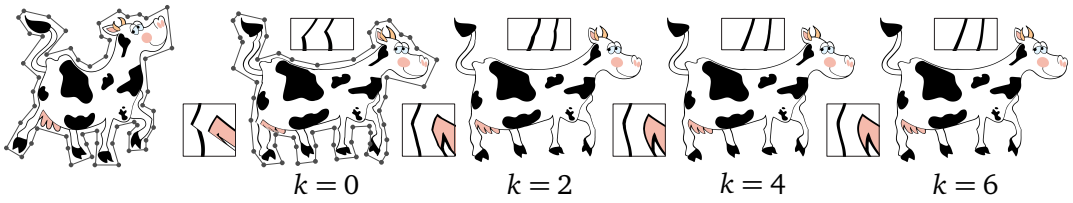


Figure 6.28. Loop coordinates can be used to deform a source image (left) by moving the vertices of the control polygon. The quality of the deformed image improves as the level k of subdivision increases, and the result is visually smooth for $k \geq 4$, which can be seen in particular in the close-ups, which zoom to the upper part of the tail and the region next to the udder of the cow.

cactus is deformed non-rigidly. While the property (1.9) of barycentric coordinates guarantees the reproduction of rigid transformations, we can see that the hat is deformed by maximum entropy and harmonic coordinates, because it is also influenced by the control points in the neck region. In contrast, the locality of local and Loop coordinates guarantees the mapping to be isometric in the hat region. Deformation with mean value coordinates suffers high distortions and is visually inappropriate for this example.

Figure 6.30 gives an example of image deformation using different initial triangulations. The constrained Delaunay triangulation of P leads to foldovers at concave corners, but they can be fixed either by adding to Δ^0 a few extra boundary vertices or by our vertex adjustment strategy (see Section 6.4.1). Both approaches give visually similar results, but vertex adjustment does not require any manual interaction. In addition, both methods improve the locality of the coordinate functions.

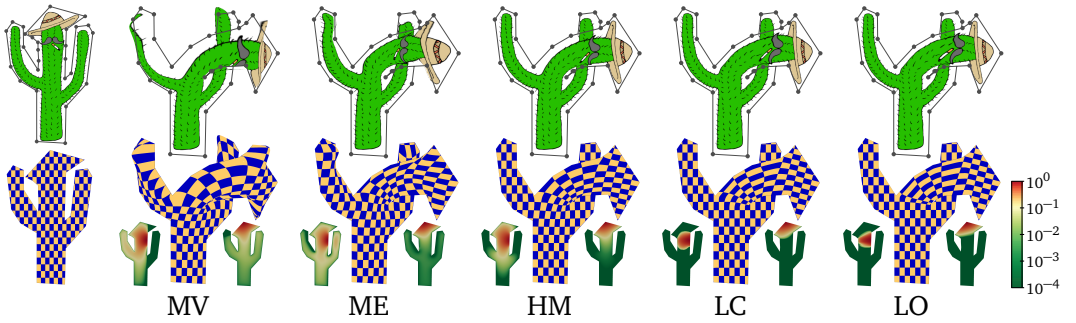


Figure 6.29. Comparison of image deformation with different barycentric coordinates. The inset shows the coordinate functions for two different vertices of the polygon, where the colour scale is logarithmic, ranging from 10^{-4} to 10^{0}.

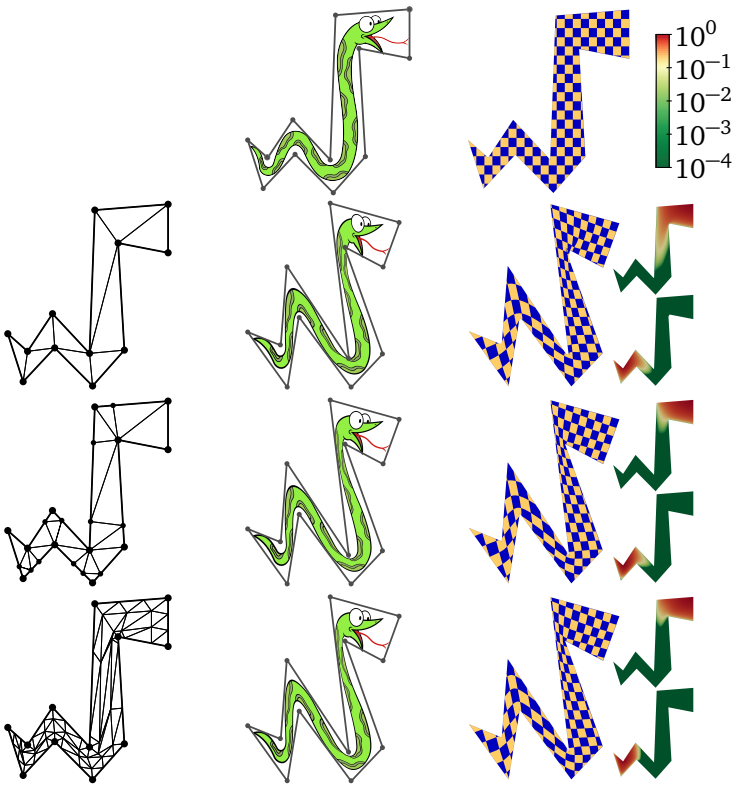


Figure 6.30. Comparison of image deformation with Loop coordinates for different initial meshes: the constrained Delaunay triangulation of P (second row) and the modified triangulations after adding boundary vertices (third row) or vertex adjustment (bottom row). The right column shows the coordinate functions for one convex and one concave vertex, where the colour scale is the same as in Figure 6.29.

We implemented two variants of this image deformation algorithm in C++ on a MacBook Pro 2013 with 2.4 GHz Intel Core i7 processor and 8 GB 1600 MHz DDR3 memory. For a given subdivision level k, the first variant computes and stores the mesh Δ^k as well as the coordinate functions b_i^k in a preprocessing step and determines the target mesh $\tilde{\Delta}^k$ by evaluating (6.4) for all vertices $\boldsymbol{p}$ of Δ^k whenever the target polygon changes. The second variant computes and stores the k meshes $\Delta^1, \ldots, \Delta^k$ in a preprocessing step and then determines $\tilde{\Delta}^k$ by subdividing the initial data $\boldsymbol{f}_i = \tilde{\boldsymbol{v}}_i$ whenever required. Figure 6.31 shows that both methods are interactive up to $k = 6$, which is sufficient for achieving visually smooth results (see Figure 6.28), and that the first variant is generally faster, even though the preprocessing is more costly. However, it is much less memory efficient, because it has to store the precomputed coordinates for all

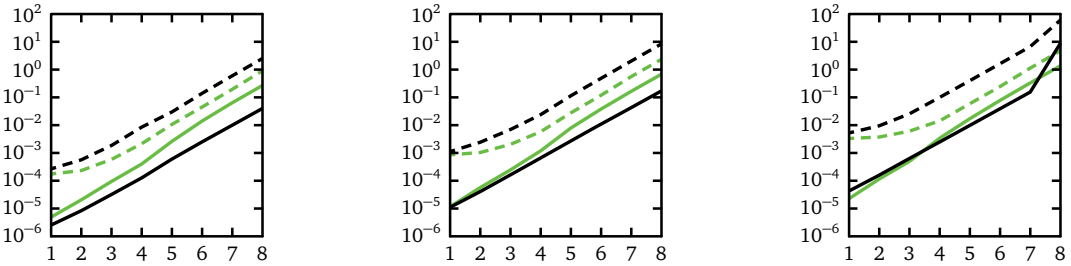


Figure 6.31. Timings for image deformation for $k = 1, \ldots, 8$, using (6.4) (black) and by subdividing the target vertex data (green). The dashed lines refer to the preprocessing costs.

vertices of the subdivided mesh Δ^k. For the cow example with $k = 8$, this data does not fit into the main memory and has to be swapped to the hard disk, which explains the unexpected increase in the corresponding plot. Note that image deformation with harmonic or local coordinates would face the same problem, because it requires to precompute and store the same data.

We should note that subdivision has been used before for image deformation [DeRose et al., 1998; Van den Bergh et al., 2002], but neither in the context of barycentric coordinates nor addressing the problem of foldovers at concave corners.

6.7 Discussion

Mesh subdivision is widely known in computer graphics as a technique for creating smooth surfaces with arbitrary topology by repeatedly refining an initial base mesh with simple local rules. In this chapter, we show that subdivision can also be used to construct barycentric coordinates with favourable properties. While the theory developed in Section 6.1 is general and works for a large class of subdivision schemes, we believe that Loop subdivision is the method of choice, for two reasons. On the one hand, it is simple and comes with well-understood boundary rules and exact evaluation routines. On the other hand, our examples confirm that the main shape of the limit coordinate functions b_i^∞ is dictated by the initial functions b_i^0, and we do not expect other subdivision schemes to yield qualitatively better results.

However, it still remains future work to develop a strategy for constructing initial triangulations Δ^0, for which it can be formally proven that the refined triangulations Δ^k are regular in the interior, even in the limit. Note that this

problem is not restricted to the construction of well-defined Loop coordinates, as it addresses the general question under which conditions the two-dimensional Loop mapping $\bar{\varphi}: P \to P$ is bijective. Another direction for future work is the extension of our approach to 3D by using volumetric subdivision schemes [Chang et al., 2002; Schaefer et al., 2004].

Chapter 7

Conclusion

The main goal of this dissertation is to give an extensive overview of generalized barycentric coordinates in 2D, analyse and improve some properties of several well-known coordinate functions, provide their numerically stable implementation, and present new constructions with favourable properties. We also show that barycentric coordinates have the well-developed theoretical basis and are often used in practice as a building block for different applications in computer graphics and geometry processing.

The key observation that we use throughout this dissertation is that the *affine combination*

$$\sum_{j=1}^{m} \omega_j b_j = \omega_1 b_1 + \omega_2 b_2 + \cdots + \omega_m b_m, \qquad \sum_{j=1}^{m} \omega_j = 1 \tag{7.1}$$

of barycentric coordinates b_j with some coefficients ω_j keeps the defining properties (1.8) and (1.9) of the coordinates. Moreover, if both b_j and ω_j are non-negative, that is, (7.1) is the *convex combination*, the non-negativity property also holds for the new coordinates formed by this combination. Finally, the correct choice of the coefficients above gives some freedom to calibrate other important properties of the new coordinates, and we give three examples of such choices in Chapters 4, 5, and 6.

While the construction in Chapter 4 is used only to improve the smoothness property at the vertices of a polygon, it gives a basic idea of how to combine different barycentric coordinates and becomes a motivation for our new construction in Chapter 5. We further generalize the idea of constructing new coordinates using convex combinations in Chapter 6.

Both constructions in Chapters 5 and 6 provide generalized barycentric coordinates in terms of Definition 1 that are non-negative inside any simple polygon,

satisfy the Lagrange property (1.10b), are linear along the polygon's edges, and are $C^1 \backslash C^2$ for any $\boldsymbol{p} \in \Omega \cup E_1 \cup \ldots \cup E_n$, where $E_i = (\boldsymbol{v}_i, \boldsymbol{v}_{i+1})$ is the i-th open edge of P. At the polygon's vertices these coordinates are at least C^0, but we show in Chapter 4 that this property can be improved and C^∞ coordinates at the convex corners of P can be constructed. At concave corners, the C^∞ continuity can be achieved only if we drop the non-negativity property (1.10a), because non-negative coordinate functions cannot be more than C^0 at such corners (see Chapter 2). Finally, both constructions give locally supported coordinates and this support is well-defined, which, to the best of our knowledge, was not achieved by any other construction before.

In addition, the coordinate functions above are also easy to evaluate for any $\boldsymbol{p} \in \Omega$. While the blended coordinates in Chapter 5 are the first, to the best of our knowledge, that have a closed form and can be computed in constant time for any simple polygon, the other construction in Chapter 6 does not offer such an efficient evaluation but has an advantage of being well-defined for polygons with interior points.

However, it still remains future work to extend these or construct new coordinates with similar properties that are also well-defined in the polygon's exterior and in higher dimensions. Note that in this case the non-negativity property cannot be guaranteed for any $\boldsymbol{p} \in \mathbb{R}^d \setminus \bar{\Omega}$.

Except for new constructions, we also analysed in Chapter 3 some well-known closed-form coordinates, namely the family of exponential three-point coordinates. We formally proved that these coordinates are well-defined inside strictly convex polygons and satisfy the Lagrange property for any parameter $p \in \mathbb{R}$. We further analysed some numerical issues that arise when implementing these coordinates and introduced the package in the Computational Geometry Algorithm Library [CGAL, 2016] with an efficient and accurate implementation of three important members of this family that are Wachspress, discrete harmonic, and mean value coordinates.

However, it still remains future work to add other generalized barycentric coordinates to our package and investigate numerical problems that can happen when computing these coordinates with the standard floating point representation of numbers. In addition, it would also be interesting to explore the problem of finding new coordinates with all properties from Section 1.2 that have a closed form and can be evaluated symbolically for any simple polygon.

We finally remark that the pseudocodes for some closed-form coordinates discussed in this dissertation can be found in Appendix A, and a C++ implementation of all coordinates can be found in the supplementary material on:
www.anisimovdmitry.com

Appendix A

Pseudocodes

In this appendix, we provide the pseudocodes for the efficient evaluation of some closed-form barycentric coordinate functions $\boldsymbol{b}$ discussed in this dissertation. In all pseudocodes, we use i to denote the vertex index, and the superindices "+" and "−" refer to the "next" and "previous" indices. For other notation, we denote the vector from the query point $\boldsymbol{p}$ to the vertex $\boldsymbol{v}_i$ by $\boldsymbol{s}_i$, the length of this vector by r_i, the vector from the vertex $\boldsymbol{v}_i$ to the vertex $\boldsymbol{v}_{i^+}$ by $\boldsymbol{e}_i$, the dot product of two vectors by "$\cdot$", the signed area of the triangle $[\boldsymbol{v}_i, \boldsymbol{v}_{i^+}, \boldsymbol{p}]$ by $A_i/2 = (\boldsymbol{s}_i \times \boldsymbol{s}_{i^+})/2$, where "$\times$" denotes the cross product, and the products $\mathcal{A}_{i-1,i}$, $\mathcal{A}_i$, and $\mathcal{A}_{i-1}$ from Equation (3.5) by X_i, Y_i, and Z_i, respectively.

We want to remark that the last three pseudocodes 9, 10, and 11 adopt slightly different notation and are used for the efficient evaluation of the *non-zero* blended coordinate functions b_i from Chapter 5 at a given point $\boldsymbol{p}$ inside a triangle $\triangle$ with one, two, or three overlapping quadrilaterals. In these pseudocodes, we use j to denote the edge index as shown in Figure A.1, 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 these pseudocodes and, therefore, an ac-

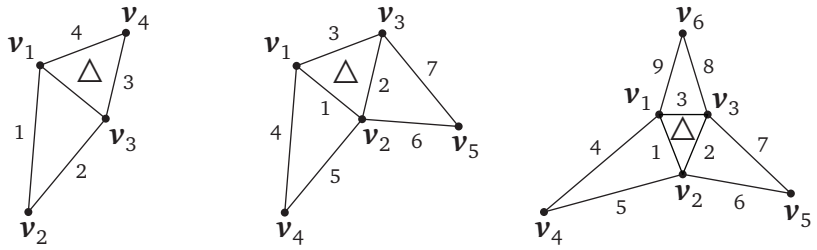


Figure A.1. Local vertex and edge indices used in the Pseudocodes 9 (left), 10 (centre), and 11 (right).

Pseudocode 2 Wachspress coordinates with the $O(n)$ time complexity (see Section 3.2.2).

```
1:  function b(p)
2:      for i = 1 to n do                          ▷ Iterate over all vertices
3:          s_i := v_i − p
4:          e_i := v_{i+} − v_i
5:      for i = 1 to n do                          ▷ Compute all necessary areas
6:          A_i := s_i × s_{i+}
7:      W := 0                                     ▷ Compute Wachspress weights
8:      for i = 1 to n do
9:          w_i := (e_i × −e_{i−}) / (A_{i−} A_i)
10:         W := W + w_i
11:     for i = 1 to n do                          ▷ Compute Wachspress coordinates
12:         b_i := w_i / W
```

Pseudocode 3 Wachspress coordinates with the $O(n^2)$ time complexity (see Section 3.2.2).

```
1:  function b(p)
2:      for i = 1 to n do                          ▷ Iterate over all vertices
3:          s_i := v_i − p
4:          e_i := v_{i+} − v_i
5:      W := 0                                     ▷ Compute Wachspress weights
6:      for i = 1 to n do
7:          w_i := e_i × −e_{i−}
8:          for j = 1 to n do
9:              if j ≠ i^− and j ≠ i then
10:                 w_i := w_i (s_j × s_{j+})
11:         W := W + w_i
12:     for i = 1 to n do                          ▷ Compute Wachspress coordinates
13:         b_i := w_i / W
```

tual implementation must take special care of correctly mapping all local indices to the corresponding global indices. For example, in Pseudocode 10, 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 A.1, centre), which in turn refer to certain global vertex indices, depending on $\triangle$.

Pseudocode 4 Discrete harmonic coordinates with the $O(n)$ time complexity (see Section 3.2.3).

function $\boldsymbol{b}(\boldsymbol{p})$
  for $i = 1$ to n **do** ▷ Iterate over all vertices
    $\boldsymbol{s}_i := \boldsymbol{v}_i - \boldsymbol{p}$
    $r_i^2 := \boldsymbol{s}_i \cdot \boldsymbol{s}_i$
  for $i = 1$ to n **do** ▷ Compute all necessary areas
    $A_i := \boldsymbol{s}_i \times \boldsymbol{s}_{i^+}$
  $W := 0$ ▷ Compute discrete harmonic weights
  for $i = 1$ to n **do**
    $w_i := (r_{i^+}^2 A_{i^-} - r_i^2 (\boldsymbol{s}_{i^-} \times \boldsymbol{s}_{i^+}) + r_{i^-}^2 A_i) / (A_{i^-} A_i)$
    $W := W + w_i$
  for $i = 1$ to n **do** ▷ Compute discrete harmonic coordinates
    $b_i := w_i / W$

Pseudocode 5 Discrete harmonic coordinates with the $O(n^2)$ time complexity (see Section 3.2.3).

function $\boldsymbol{b}(\boldsymbol{p})$
  for $i = 1$ to n **do** ▷ Iterate over all vertices
    $\boldsymbol{s}_i := \boldsymbol{v}_i - \boldsymbol{p}$
    $r_i^2 := \boldsymbol{s}_i \cdot \boldsymbol{s}_i$
  for $i = 1$ to n **do** ▷ Compute the products of areas
    $X_i := 1$
    for $j = 1$ to n **do**
      if $j \neq i^-$ and $j \neq i$ **then**
        $X_i := X_i (\boldsymbol{s}_j \times \boldsymbol{s}_{j^+})$
    $Y_i := X_i (\boldsymbol{s}_{i^-} \times \boldsymbol{s}_i)$, $Z_i := X_i (\boldsymbol{s}_i \times \boldsymbol{s}_{i^+})$
  $W := 0$ ▷ Compute discrete harmonic weights
  for $i = 1$ to n **do**
    $w_i := Y_i r_{i^+}^2 - X_i r_i^2 (\boldsymbol{s}_{i^-} \times \boldsymbol{s}_{i^+}) + Z_i r_{i^-}^2$
    $W := W + w_i$
  for $i = 1$ to n **do** ▷ Compute discrete harmonic coordinates
    $b_i := w_i / W$

Pseudocode 6 Mean value coordinates with the $O(n)$ time complexity (see Section 3.2.4).

function $\boldsymbol{b}(\boldsymbol{p})$
  for $i = 1$ to n **do** ▷ Iterate over all vertices
    $\boldsymbol{s}_i := \boldsymbol{v}_i - \boldsymbol{p}$
    $r_i := \|\boldsymbol{s}_i\|$
  for $i = 1$ to n **do** ▷ Compute all necessary data
    $t_i := (\boldsymbol{s}_i \times \boldsymbol{s}_{i^+}) / (r_i\, r_{i^+} + \boldsymbol{s}_i \cdot \boldsymbol{s}_{i^+})$
  $W := 0$ ▷ Compute mean value weights
  for $i = 1$ to n **do**
    $w_i := (t_{i^-} + t_i) / r_i$
    $W := W + w_i$
  for $i = 1$ to n **do** ▷ Compute mean value coordinates
    $b_i := w_i / W$

Pseudocode 7 Mean value coordinates with the $O(n^2)$ time complexity (see Section 3.2.4), where the sign σ_i is computed as in Pseudocode 1.

function $\boldsymbol{b}(\boldsymbol{p})$
  for $i = 1$ to n **do** ▷ Iterate over all vertices
    $\boldsymbol{s}_i := \boldsymbol{v}_i - \boldsymbol{p}$
    $r_i := \|\boldsymbol{s}_i\|$
  for $i = 1$ to n **do** ▷ Compute mean value weights
    $w_i := r_{i^-}\, r_{i^+} - \boldsymbol{s}_{i^-} \cdot \boldsymbol{s}_{i^+}$
    for $j = 1$ to n **do**
      if $j \neq i^-$ and $j \neq i$ **then**
        $w_i := w_i\, (r_j\, r_{j^+} + \boldsymbol{s}_j \cdot \boldsymbol{s}_{j^+})$
    $w_i := \sigma_i \sqrt{w_i}$
    $W := W + w_i$
  for $i = 1$ to n **do** ▷ Compute mean value coordinates
    $b_i := w_i / W$

Pseudocode 8 Convex combination of Wachspress and mean value coordinates from Chapter 4, where the functions "`getdWx()`" and "`getdWy()`" return partial derivatives dW_x and dW_y of the Wachspress denominator W^{WP} with respect to x and y coordinates.

function $\boldsymbol{b}(\boldsymbol{p})$
  $\{\boldsymbol{b}^{\text{WP}}, W^{\text{WP}}\} :=$ `computeWPCoordinates(`$\boldsymbol{p}$`)` ▷ See Pseudocode 3
  $dW_x :=$ `getdWx(`$\boldsymbol{p}$`)` ▷ Compute blending function
  $dW_y :=$ `getdWy(`$\boldsymbol{p}$`)`
  $\sigma_1 := dW_x\, dW_x + dW_y\, dW_y,\ \sigma_2 := 1$
  for $i = 1$ to n **do**
    $\boldsymbol{s}_i := \boldsymbol{v}_i - \boldsymbol{p}$
    $\sigma_2 := \sigma_2 (\boldsymbol{s}_i \cdot \boldsymbol{s}_i)$
  $\sigma_1 := (W^{\text{WP}})^4 / (\sigma_1)^2$
  $\Sigma := \sigma_1 + \sigma_2,\ \mu := 0$
  if $\Sigma > 0$ **then**
    $\mu := \sigma_1 / \Sigma$
  $\boldsymbol{b}^{\text{MV}} :=$ `computeMVCoordinates(`$\boldsymbol{p}$`)` ▷ See Pseudocode 7
  for $i = 1$ to n **do** ▷ Final convex combination
    $b_i := \mu\, b_i^{\text{WP}} + (1-\mu)\, b_i^{\text{MV}}$

Pseudocode 9 Blended coordinates from Chapter 5, case: one quadrilateral.

function $\boldsymbol{b}(\boldsymbol{p})$
  for $i = 1$ to 4 **do** ▷ Iterate over all vertices
    $\boldsymbol{s}_i := \boldsymbol{v}_i - \boldsymbol{p}$
    $r_i := \|\boldsymbol{s}_i\|$
  for $j = 1$ to 4 **do** ▷ Iterate over all edges
    $t_j := (\boldsymbol{s}_i \times \boldsymbol{s}_{i^+}) / (r_i\, r_{i^+} + \boldsymbol{s}_i \cdot \boldsymbol{s}_{i^+})$
  $W := 0$ ▷ Mean value weights
  for $k = 1$ to 4 **do**
    $w_k := (t_{j^-} + t_j) / r_i$
    $W := W + w_k$
  for $i = 1$ to 4 **do** ▷ Blended coordinates
    $b_i := w_k / W$

Pseudocode 10 Blended coordinates from Chapter 5, case: two overlapping quadrilaterals.

function $\boldsymbol{b}(\boldsymbol{p})$
  for $i = 1$ to 5 **do** ▷ Iterate over all vertices
    $\boldsymbol{s}_i := \boldsymbol{v}_i - \boldsymbol{p}$
    $r_i := \|\boldsymbol{s}_i\|$
  for $j = 1$ to 7 **do** ▷ Iterate over all edges
    $A_j := \boldsymbol{s}_i \times \boldsymbol{s}_{i^+}$
    $t_j := A_j \,/\, (r_i\, r_{i^+} + \boldsymbol{s}_i \cdot \boldsymbol{s}_{i^+})$
  $A := A_1 + A_2 + A_3,\ a_1 := A_1 / A,\ a_2 := A_2 / A$ ▷ Blending functions
  $\sigma_1 := q(a_2),\quad \sigma_2 := q(a_1),\quad \Sigma := \sigma_1 + \sigma_2$
  $\mu_1 := \sigma_1 / \Sigma,\ \mu_2 := \sigma_2 / \Sigma$
  $W_1 := 0,\ W_2 := 0$ ▷ Mean value weights
  for $k = 1$ to 4 **do**
    $w_k^1 := (t_{j^-}^1 + t_j^1) / r_i^1,\ w_k^2 := (t_{j^-}^2 + t_j^2) / r_i^2$
    $W_1 := W_1 + w_k^1, \qquad W_2 := W_2 + w_k^2$
  for $k = 1$ to 4 **do** ▷ Mean value coordinates
    $b_k^1 := w_k^1 / W_1, \qquad b_k^2 := w_k^2 / W_2$
  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 11 Blended coordinates from Chapter 5, case: three overlapping quadrilaterals.

1: **function** $\boldsymbol{b}(\boldsymbol{p})$
2: **for** $i = 1$ to 6 **do** ▷ Iterate over all vertices
3: $\boldsymbol{s}_i := \boldsymbol{v}_i - \boldsymbol{p}$
4: $r_i := \|\boldsymbol{s}_i\|$
5: **for** $j = 1$ to 9 **do** ▷ Iterate over all edges
6: $A_j := \boldsymbol{s}_i \times \boldsymbol{s}_{i^+}$
7: $t_j := A_j \,/\, (r_i\, r_{i^+} + \boldsymbol{s}_i \cdot \boldsymbol{s}_{i^+})$
8: ▷ Blending functions
9: $A := A_1 + A_2 + A_3,\ a_1 := A_1 / A,\ a_2 := A_2 / A,\ a_3 := A_3 / A,$
10: $\sigma_1 := q(a_2)\,q(a_3),\ \sigma_2 := q(a_1)\,q(a_2),\ \sigma_3 := q(a_1)\,q(a_3),\ \Sigma := \sigma_1 + \sigma_2 + \sigma_3$
11: $\mu_1 := \sigma_1 / \Sigma,\ \ \mu_2 := \sigma_2 / \Sigma,\ \ \mu_3 := \sigma_3 / \Sigma$
12: $W_1 := 0,\ W_2 := 0,\ W_3 := 0$ ▷ Mean value weights
13: **for** $k = 1$ to 4 **do**
14: $w_k^1 := (t_{j^-}^1 + t_j^1) / r_i^1,\ w_k^2 := (t_{j^-}^2 + t_j^2) / r_i^2,\ w_k^3 := (t_{j^-}^3 + t_j^3) / r_i^3$
15: $W_1 := W_1 + w_k^1, \qquad W_2 := W_2 + w_k^2, \qquad W_3 := W_3 + w_k^3$
16: **for** $k = 1$ to 4 **do** ▷ Mean value coordinates
17: $b_k^1 := w_k^1 / W_1, \qquad b_k^2 := w_k^2 / W_2, \qquad b_k^3 := w_k^3 / W_3$
18: **for** $i = 1$ to 3 **do** ▷ Blended coordinates
19: $b_i := b_k^1\,\mu_1 + b_k^2\,\mu_2 + b_k^3\,\mu_3$
20: $b_4 := b_4^1\,\mu_1,\ b_5 := b_4^2\,\mu_2,\ b_6 := b_4^3\,\mu_3$

Bibliography

Arroyo, M. and Ortiz, M. [2006]. Local maximum-entropy approximation schemes: A seamless bridge between finite elements and meshfree methods, *International Journal for Numerical Methods in Engineering* **65**(13): 2167–2202.

Belikov, V. V., Ivanov, V. D., Kontorovich, V. K., Korytnik, S. A. and Semenov, A. Y. [1997]. The non-Sibsonian interpolation: A new method of interpoaltion of the values of a function on an arbitrary set of points, *Computational Mathematics and Mathematical Physics* **37**(1): 9–15.

Belyaev, A. [2006]. On transfinite barycentric coordinates, *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, pp. 89–99.

Belyaev, A. and Fayolle, P.-A. [2015]. On transfinite Gordon–Wixom interpolation schemes and their extensions, *Computers & Graphics* **51**: 74–80.

Ben-Chen, M., Weber, O. and Gotsman, C. [2009]. Spatial deformation transfer, *in* E. Grinspun and J. Hodgins (eds), *Proceedings of SCA 2009*, ACM SIGGRAPH / Eurographics Symposium Proceedings, New Orleans, LA, pp. 67–74.

Biermann, H., Levin, A. and Zorin, D. [2000]. Piecewise smooth subdivision surfaces with normal control, *Proceedings of SIGGRAPH 2000*, Annual Conference Series, New Orleans, LA, pp. 113–120.

Bobach, T., Hering-Bertram, M. and Umlauf, G. [2006]. Comparison of Voronoi based scattered data interpolation schemes, *Proceedings of International Conference on Visualization, Imaging and Image Processing*, pp. 342–349.

Budninskiy, M., Liu, B., Tong, Y. and Desbrun, M. [2016]. Power coordinates: A geometric construction of barycentric coordinates on convex polytopes, *ACM Transactions on Graphics* **35**(6).

Cashman, T. J. [2012]. Beyond Catmull–Clark? A survey of advances in subdivision surface methods, *Computer Graphics Forum* **31**(1): 42–61.

Catmull, E. and Clark, J. [1978]. Recursively generated B-spline surfaces on arbitrary topological meshes, *Computer-Aided Design* **10**(6): 350–355.

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

CGAL [2016]. Computational Geometry Algorithms Library, http://www.cgal.org.

Chang, Y.-S., McDonnell, K. T. and Qin, H. [2002]. A new solid subdivision scheme based on box splines, *in* K. Lee and N. M. Patrikalakis (eds), *Proceedings of the Seventh ACM Symposium on Solid Modeling an Applications*, Saarbrücken, Germany, pp. 226–233.

Chen, R. and Gotsman, C. [2016]. On pseudo-harmonic barycentric coordinates, *Computer Aided Geometric Design* **44**: 15–35.

Christ, N. H., Friedberg, R. and Lee, T. D. [1982]. Weights of links and plaquettes in a random lattice, *Nuclear Physics B* **210**(3): 337–346.

d. Monantheuil, H. [1599]. *Aristotelis mechanica*, Jeremias Perier, Paris. In Greek, corrected, translated into Latin and illustrated with commentary.

Deng, B. and Liu, Z. [2014]. Code for computing local barycentric coordinates, https://github.com/bldeng/LBC. [Online; accessed 28-September-2016].

DeRose, T., Kass, M. and Truong, T. [1998]. Subdivision surfaces in character animation, *Proceedings of SIGGRAPH*, Annual Conference Series, Orlando, FL, pp. 85–94.

Desbrun, M., Meyer, M. and Alliez, P. [2002]. Intrinsic parameterizations of surface meshes, *Computer Graphics Forum* **21**(3): 209–218.

Devillers, O. [1998]. Improved incremental randomized Delaunay triangulation, *Proceedings of the 14th Annual ACM Symposium on Computational Geometry*, pp. 106–115.

Dyn, N., Levin, D. and Gregory, J. A. [1990]. A butterfly subdivision scheme for surface interpolation with tension control, *ACM Transactions on Graphics* **9**(2): 160–169.

Eck, M., DeRose, T., Duchamp, T., Hoppe, H., Lounsbery, M. and Stuetzle, W. [1995]. Multiresolution analysis of arbitrary meshes, *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, pp. 173–182.

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

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

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

Floater, M. S. [2014]. Wachspress and mean value coordinates, *in* G. Fasshauer and L. L. Schumaker (eds), *Approximation Theory XIV: San Antonio 2013*, Springer, pp. 81–102.

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

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

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

Gamma, E., Helm, R., Johnson, R. and Vlissides, J. [1995]. *Design patterns: Elements of reusable object-oriented software*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Ginkel, I., Peters, J. and Umlauf, G. [2007]. Normals of subdivision surfaces and their control polyhedron, *Computer Aided Geometric Design* **24**(2): 112–116.

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

Guennebaud, G., Jacob, B. et al. [2010]. Eigen v3, http://eigen.tuxfamily.org. [Online; accessed 28-September-2016].

Hiyoshi, H. and Sugihara, K. [2000]. Voronoi-based interpolation with higher continuity, *Proceedings of the sixteenth annual symposium on Computational geometry*, pp. 242–250.

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

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

Hormann, K. and Tarini, M. [2004]. A quadrilateral rendering primitive, Eurographics Symposium Proceedings, pp. 7–14.

Jaynes, E. T. [1957]. Information theory and statistical mechanics, *Physical Review* **106**(4): 620–630.

Jaynes, E. T. [1963]. Information theory and statistical mechanics, *in* K. W. Ford (ed.), *Statistical Physics*, Vol. 3 of *Brandeis University Summer Institute Lectures in Theoretical Physics*, W. A. Benjamin, New York, pp. 181–218.

Joshi, P., Meyer, M., DeRose, T., Green, B. and Sanocki, T. [2007]. Harmonic coordinates for character articulation, *ACM Transactions on Graphics* **26**(3): 71:1–9.

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

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

Ju, T., Schaefer, S., Warren, J. and Desbrun, M. [2005]. A geometric construction of coordinates for convex polyhedra using polar duals, *Symposium on Geometry Processing*, pp. 181–186.

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

Kullback, S. and Leibler, R. A. [1951]. On information and sufficiency, *The Annals of Mathematical Statistics* **22**(1): 79–86.

Langer, T., Belyaev, A. and Seidel, H.-P. [2006]. Spherical barycentric coordinates, *Symposium on Geometry Processing*, pp. 81–88.

LEDA [2016]. Library of Efficient Data Types and Algorithms, http://www.algorithmic-SBsolutions.com/leda/.

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

Li, X.-Y., Ju, T. and Hu, S.-M. [2013]. Cubic mean value coordinates, *ACM Transactions on Graphics* **32**(4): 126:1–10.

Lipman, Y., Kopf, J., Cohen-Or, D. and Levin, D. [2007]. GPU-assisted positive mean value coordinates for mesh deformations, *in* A. Belyaev and M. Garland (eds), *Proceedings of SGP 2007*, Eurographics Symposium Proceedings, pp. 117–123.

Lipman, Y., Levin, D. and Cohen-Or, D. [2008]. Green coordinates, *ACM Transactions on Graphics* **27**(3): 78:1–10.

Loop, C. T. [1987]. *Smooth subdivision surfaces based on triangles*, Master's thesis, Department of Mathematics, The University of Utah.

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

Malsch, E. A. and Dasgupta, G. [2004]. Interpolations for temperature distributions: A method for all non-concave polygons, *International Journal of Solids and Structures* **41**(8): 2165–2188.

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

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

Manson, J. and Schaefer, S. [2010]. Moving least squares coordinates, *Computer Graphics Forum* **29**(5): 1517–1524.

Martin, S., Kaufmann, P., Botsch, M., Wicke, M. and Gross, M. [2008]. Polyhedral finite elements using harmonic basis functions, *Computer Graphics Forum* **27**(5): 1521–1529.

Meyer, M., Desbrun, M., Schröder, P. and Barr, A. H. [2003]. Discrete differential-geometry operators for triangulated 2-manifolds.

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

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

MPFR [2016]. GNU Multiple Precision Floating-Point Reliably, http://www.mpfr.org.

Nocedal, J. and Wright, S. J. [1999]. *Numerical optimization*, Springer Series in Operations Research, Springer, New York.

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

Rustamov, R. M. [2008]. Boundary element formulation of harmonic coordinates, *Technical report*, Purdue university.

Schaefer, S., Hakenberg, J. and Warren, J. [2004]. Smooth subdivision of tetrahedral meshes, *in* R. Scopigno and D. Zorin (eds), *Proceedings of SGP 2004*, Eurographics Symposium Proceedings, Nice, France, pp. 147–154.

Schaefer, S. and Warren, J. [2007]. Exact evaluation of non-polynomial subdivision schemes at rational parameter values, *in* M. Alexa, S. Gortler and T. Ju (eds), *Proceedings of Pacific Graphics 2007*, Maui, HI, pp. 321–330.

Schaefer, S. and Warren, J. [2008]. Exact evaluation of limits and tangents for non-polynomial subdivision schemes, *Computer Aided Geometric Design* **25**(8): 607–620.

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

Shewchuk, J. R. [1996]. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator, *in* M. C. Lin and D. Manocha (eds), *Applied Computational Geometry: Towards Geometric Engineering*, Vol. 1148 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, Heidelberg, pp. 203–222.

Shore, J. E. and Johnson, R. W. [1980]. Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy, *IEEE Transactions on Information Theory* **26**(1): 26–37.

Sibson, R. [1980]. A vector identity for the Dirichlet tessellation, *Mathematical Proceedings of the Cambridge Philosophical Society* **87**(1): 151–155.

Sibson, R. [1981]. A brief description of natural neighbor interpolation, *Interpreting Multivariate Data* **21**: 21–36.

Stam, J. [1998a]. Exact evaluation of Catmull–Clark subdivision surfaces at arbitrary parameter values, *Proceedings of SIGGRAPH 1998*, Annual Conference Series, Orlando, FL, pp. 395–404.

Stam, J. [1998b]. Exact evaluation of Loop subdivision surfaces, *Proceedings of SIGGRAPH 1998*, CD-ROM, Orlando, FL, pp. 1–15.

Standards Committee, I. et al. [2008]. 754-2008 ieee standard for floating-point arithmetic, *IEEE Computer Society Std* .

Sugihara, K. [1999]. Surface interpolation based on new local coordinates, *Computer-Aided Design* **31**(1): 51–58.

Sukumar, N. [2003]. Voronoi cell finite difference method for the diffusion operator on arbitrary unstructured grids, *International Journal for Numerical Methods in Engineering* **57**(1): 1–34.

Sukumar, N. [2004]. Construction of polygonal interpolants: A maximum entropy approach, *International Journal for Numerical Methods in Engineering* **61**(12): 2159–2181.

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

Sukumar, N., Moran, B. and Belytschko, T. [1998]. The natural element method in solid mechanics, *International Journal for Numerical Methods in Engineering* **43**(5): 839–887.

Sukumar, N., Moran, B., Semenov, A. Y. and Belikov, V. V. [2001]. Natural neighbor Galerkin methods, *International Journal for Numerical Methods in Engineering* **50**(1): 1–27.

Sukumar, N. and Wets, R. J.-B. [2007]. Deriving the continuity of maximum-entropy basis functions via variational analysis, *SIAM Journal on Optimization* **18**(3): 914–925.

Sukumar, N. and Wright, R. W. [2007]. 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.

Takayama, K., Sorkine, O., Nealen, A. and Igarashi, T. [2010]. Volumetric modeling with diffusion surfaces, *ACM Transactions on Graphics* **29**(6): Article 180, 8 pages.

Taubin, G. [1994]. Distance approximations for rasterizing implicit curves, *ACM Transactions on Graphics* **13**(1): 3–42.

Van den Bergh, J., Di Fiore, F., Claes, J. and Van Reeth, F. [2002]. Interactively morphing irregularly shaped images employing subdivision techniques, *Proceedings of the 1st Ibero-American Symposium on Computer Graphics 2002*, Guimarães, Portugal, pp. 315–321.

Voronoi, G. [1908]. Nouvelles applications des paramètres continus à la théorie de formes quadratiques, *Journal Für Die Reine und Angewandte Mathematik* **134**: 198–287.

Wachspress, E. L. [1975]. *A rational finite element basis*, Mathematics in Science and Engineering, Academic Press, New York.

Waldron, S. [2011]. Affine generalised barycentric coordinates, *Jaen Journal on Approximation* **3**(2): 209–226.

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

Warren, J. [2003]. On the uniqueness of barycentric coordinates, *Contemporary Mathematics* **334**: 93–100.

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

Weber, O., Ben-Chen, M. and Gotsman, C. [2009]. Complex barycentric coordinates with applications to planar shape deformation, *Computer Graphics Forum* **28**(2): 587–597.

Weber, O., Ben-Chen, M., Gotsman, C. and Hormann, K. [2011]. A complex view of barycentric mappings, *Computer Graphics Forum* **30**(5): 1533–1542.

Weber, O. and Gotsman, C. [2010]. Controllable conformal mappings for shape deformation and interpolation, *ACM Transactions on Graphics* **29**(4): 78.

Weber, O., Poranne, R. and Gotsman, C. [2012]. Biharmonic coordinates, *Computer Graphics Forum* **31**(8): 2409–2422.

Yvinec, M. [2016]. 2D triangulation, *CGAL User and Reference Manual*, 4.9 edn, CGAL Editorial Board.

Zhang, J., Deng, B., Liu, Z., Patanè, G., Bouaziz, S., Hormann, K. and Liu, L. [2014]. Local barycentric coordinates, *ACM Transactions on Graphics* **33**(6): 188:1–12.

Zorin, D. and Kristjansson, D. [2002]. Evaluation of piecewise smooth subdivision surfaces, *The Visual Computer* **18**(5–6): 299–315.

Zorin, D. and Schröder, P. [2000]. *Subdivision for modeling and animation*, number 23 in *SIGGRAPH 2000 Course Notes*, ACM Press.

Zorin, D., Schröder, P. and Sweldens, W. [1996]. Interpolating subdivision for meshes with arbitrary topology, *Proceedings of SIGGRAPH 1996*, Annual Conference Series, New Orleans, LA, pp. 189–192.