Barycentric Coordinates for Convex Sets

Joe Warren[1], Scott Schaefer[1], Anil N. Hirani[2] and Mathieu Desbrun[3]

August 10, 2005

[1]Rice University
6100 Main St.
Houston, TX 77005

[2]JPL/Caltech
4800 Oak Grove Drive
Pasadena, CA 91109

[3]Caltech
1200 E. California Boulevard
Pasadena, CA 91125

Abstract

In this paper we provide an extension of barycentric coordinates from simplices to arbitrary convex sets. Barycentric coordinates over convex 2D polygons have found numerous applications in various fields as it allows smooth interpolation of data located on vertices. However, no explicit formulation valid for arbitrary convex polytopes has been proposed to extend this interpolation in higher dimensions. Moreover, there has been no attempt to extend these functions into the continuous domain, where barycentric coordinates are related to Green's functions and construct functions that satisfy a boundary value problem.

First, we review the properties and construction of barycentric coordinates in discrete domain for convex polytopes. Next, we show how these concepts extend into the continuous domain to yield barycentric coordinates for continuous functions. We then provide a proof that our functions satisfy all the desirable properties of barycentric coordinates in arbitrary dimensions. Finally, we provide an example of constructing such barycentric functions over regions bounded by parametric curves and show how they can be used to perform freeform deformations.

Keywords: barycentric coordinates, convex polyhedra, convex sets
Classification: 52B55 [Convex and discrete geometry]: Computational aspects related to convexity

1 Introduction

Introduced by Möbius in 1827 as *mass points* to define a coordinate-free geometry, barycentric coordinates over simplices are a very common tool in many computations. In addition to their coordinate-free expressions, barycentric coordinates

are extremely helpful for interpolating discrete scalar fields, vector fields, or arbitrary multidimensional fields over irregular tessellations: they naturally interpolate values at vertices to the whole space via multilinear interpolation.

Since its inception, the graphics community has made extensive use of barycentric coordinates. In early work barycentric coordinates were routinely used for triangles, with applications such as polygon rasterization, texture mapping, ray-triangle intersection in raytracing, spline patches, interpolation, etc. More recently, barycentric coordinates for tetrahedra have been used for interpolation of 3D fields for volume rendering and isosurface extraction, as well as for simulation purposes since they define convenient linear basis functions over simplices. More generally, many applied fields such as computational physics and mechanics rely heavily on barycentric coordinates since it corresponds to linear basis functions in the finite element method. Note that even higher dimensional graphics-related data require appropriate interpolation between discrete samples, such as for lightfield applications.

A natural question arises when interpolation is needed over more complex shapes, such as polygons or polytopes: can we extend this notion of barycentric coordinates to arbitrary polytopes? The common way to deal with irregular polygons in 2D or general polyhedra in 3D is to triangulate them first, and apply barycentric coordinates on each simplex. However this solution is unacceptable for many applications: the results depend on the choice of triangulation, and contain unnecessary artifacts, mostly due to the restrictive C^0 continuity across simplices.

Despite some recent work on *generalized* barycentric coordinates, valid for arbitrary polytopes, no explicit formulation has been given for polytopes of arbitrary dimension. Furthermore, there has been, to the best of our knowledge, no attempt at extending these coordinates to *smooth* convex sets. This paper addresses these current limitations, by providing geometric and computationally-convenient explicit formulations along with proofs of their validity.

1.1 Problem Definition for Convex Polytopes

Given a bounded, convex polytope P, our problem is to construct one coordinate function $b_v(x)$ per vertex v of P for all $x \in P$. These functions are *barycentric coordinates* with respect to P if they satisfy three properties. First, the coordinate functions are *non-negative* on P,

$$b_v(x) \geq 0,$$

for all $x \in P$. Second, the functions form a *partition of unity*,

$$\sum_v b_v(x) = 1,$$

for all x. Finally, the functions act as coordinates in that, given a value of x, weighting each vertex v by $b_v(x)$ returns back x; i.e.,

$$\sum_v v\, b_v(x) = x. \tag{1}$$

This last property is also sometimes referred to as *linear precision* since the coordinate functions can reproduce the linear function x.

A typical application of barycentric coordinates is to interpolate *any* data (not just positions) provided at the vertices of P. Given a set of values g_v associated with each v, we build a new function $\hat{g}(x)$ defined over P by

$$\hat{g}(x) = \sum_v g_v b_v(x). \tag{2}$$

The constructed function $\hat{g}(x)$ satisfies the property that $\hat{g}(v) = g_v$.

1.2 Problem Definition for Convex Sets with Smooth Boundaries

Another goal of this paper is to provide an extension of barycentric coordinates to smooth, convex regions. Given a convex region P whose boundary is a smooth $(d-1)$-dimensional manifold $S = \partial P$, we define a *smooth* barycentric coordinate function $b(v,x)$ analogous to the discrete case. First, the continuous coordinate function will also be non-negative on V,

$$b(v,x) \geq 0,$$

for all $v \in S$ and $x \in P$. The partition of unity property for a continuous barycentric function is stated as

$$\int_{v \in S} b(v,x) dS = 1.$$

Finally, these coordinate functions should also satisfy a linear precision property; namely,

$$\int_{v \in S} v\, b(v,x) dS = x.$$

Similar to the discrete case, this coordinate function can be used to build a solution to a boundary value problem on S. Given a function $g(v)$ defined for $v \in S$, we construct a function $\hat{g}(x)$ defined for $x \in P$ as

$$\hat{g}(x) = \int_{v \in S} g(v) b(v,x) dS \tag{3}$$

where $\hat{g}(v)$ interpolates $g(v)$ on S.

1.3 Previous Work

Most of the previous work on barycentric coordinates focuses on convex polygons in the plane. For the case of *regular* polygons, Loop and De Rose [13], Kuriyama [11] and Lodha [12] propose a simple construction that yield smooth basis functions. Their expressions nicely extend the well known area-based formula for barycentric coordinates in a triangle. Unfortunately, none of the proposed constructions have linear precision when applied to irregular polygons.

Pinkall and Polthier [16], and later Eck et al. [1], present a conformal parameterization for triangulated surfaces that actually provides a natural extension of barycentric coordinates to arbitrary polygons. However, the weights they define can be negative even when the polygon is convex [15], which is often problematic for interpolation applications.

Sibson [17] proposes a natural neighbor interpolant based on Voronoi diagram that yields coordinate functions that are non-negative and have linear precision; note also that Gotsman and colleagues proposed a minimization-driven barycentric coordinates [8]. The drawback with these constructions is that the resulting coordinate functions are not smooth.

Floater [3] gives an algorithmic construction coordinates over star-shaped regions in 2D. However, this construction suffers from the drawback that the resulting coordinate functions are not smooth within the polygon. In subsequent work, Floater and colleagues [4, 5] also present smooth coordinates for non-convex polygons based on the mean value theorem. Another related approach was presented by Malsch [14], allowing the design of barycentric coordinates for arbitrary 2D polygons, with or without holes, and even with interior vertices. The idea of mean value coordinates was recently extended to 3D and proven very convenient for shape deformation [9, 6]. Alas, this series of approaches do not end up providing multilinear interpolation when applied to cartesian grids, a severe drawback for multiple computational tasks.

Finally, Warren [19] presents a construction for barycentric coordinates that extend the results of Wachspress [18] to convex polytopes in *arbitrary* dimensions. The functions Warren presents are smooth functions that have linear precision, are positive on the interior of V, and coincide with the natural multilinear interpolation on cartesian grids. These functions are also rational of minimal degree as proved in Warren [20]. Unfortunately, *no explicit formulation* of these functions has been provided for practical implementations.

1.4 Contributions

Despite much work in the discrete 2D case, no explicit formulation of barycentric coordinates for convex polytopes valid in arbitrary dimension is currently available. Additionally, there has been no generalization of these coordinate functions to continuous domains that the authors are aware of. In this paper, we first review the coordinate functions for arbitrary convex polytopes of Warren [19] and offer a simple, computationally-convenient expression of these functions; we then generalize this formulation to smooth, convex regions of arbitrary dimension. In the process, we show that our weight functions for both the polytope and the continuous case satisfy the three properties of Sections 1.2 and 1.1. Finally, we show how the weight function has applications in defining (free-form) deformations over smooth, convex regions before concluding.

2 Defining Coordinates for Convex Polytopes

2.1 Setup and Notations

A *convex* polytope P in R^d is the convex hull of k affinely independent points where $k > d$. The polytope P is bounded by a set of $(d-1)$ dimensional facets with outward unit normal vectors n_j. Let $ind(v)$ denote the set of indices j such that the facet normal to n_j contains the vertex v.

Now, a vertex v of P is *simple* if $|ind(v)| = d$. Similarly, P is *simple* if every vertex of P is simple. Note that convex polygons are always simple while only a subset of convex polyhedra are simple. For example, tetrahedra, cubes and triangular prisms are simple while square pyramids and octahedra are not.

In [19], Warren shows that an appropriate extension of barycentric coordinate functions $b_v(x)$ for simple polytopes can be defined using the concept of dual polyhedral cones. In the remainder of this section, we will further *formulate* these coordinate functions to allow practical computations.

2.2 An Explicit Formulation

We first define the *weight function* $w_v(x)$ at a simple vertex v as being:

$$w_v(x) = \frac{\kappa(v)}{\prod_{j \in ind(v)} (n_j \cdot (v - x))} \tag{4}$$

where $\kappa(v)$ is the volume of the parallelepiped span by the normals to the d facets incident on v, expressed via a determinant as:

$$\kappa(v) = |Det(n_{ind(v)})| \tag{5}$$

where $n_{ind(v)}$ is a matrix whose rows are the vectors n_j where $j \in ind(v)$.

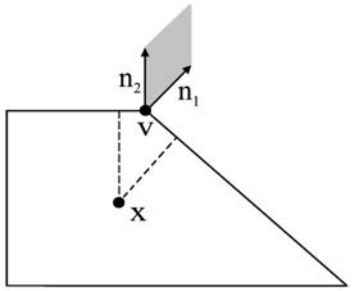

Figure 1: An example calculation of $w_v(x)$ at a vertex for a trapezoid with normals labeled (n_1, n_2). The areas of the shaded parallelograms formed by the normals correspond to the quantity $|Det(n_{i(v)})|$.

Note that this weight function depends only on the facets incident on v. In particular, the numerator corresponds to the volume of the parallelepiped spanned by the outward normal vectors n_j associated with the facets incident on v while the denominator is the product of the distances between x and the d facets adjacent to v. Figure 1 illustrates the different quantities for a parallelogram.

For non-simple vertices (where $|ind(v)| > d$), the weight functions are constructed by infinitesimally perturbing the facets touching v such that the non-simple vertex is decomposed into simple vertices. The weight function is built by then summing the weight function for the simple vertices together. Warren [19] provides a more detailed description of non-simple vertices and a proof that the weight functions are invariant under the decomposition of the non-simple vertex: so any pertubation does the trick.

Finally, we express the barycentric coordinate function $b_v(x)$ by dividing each weight function $w_v(x)$ by the sum of all weight functions taken over P:

$$b_v(x) = \frac{w_v(x)}{\sum_v w_v(x)}.$$

2.3 Equivalence to Warren's Coordinates

At this point, we make several observations concerning the structure of these functions $b_v(x)$ that we just built. First, these functions are non-negative on P due to the fact that the weight functions $w_v(x)$ are, by construction, non-negative on P. Second, these functions trivially sum to one by construction. Third, these functions have linear precision as shown in Appendix A. Finally, due to the uniqueness

theorem in [20], a simple argument on the degree of the resulting rational polynomial coordinate functions confirms the equivalence between our newly-introduced expression and the original construction by Warren. Therefore our coordinates inherit the other secondary qualities from Warren's coordinates, such as smoothness and reproduction of tensor product coordinate functions ([19]).

2.4 Application to Interpolation

Functions that satisfy the three aforementioned properties can be used to interpolate data values at the vertices of P. In other words,

$$b_v(x) = \delta_{vx}$$

if x is restricted to the vertices of the polytope P where δ is the Kronecker function. To understand this phenomenon, first observe that the barycentric coordinate functions reproduce all linear functions (linear precision yields reproduction of scalar values of x and the fact that the functions sum to 1 generates constant functions). Now, consider a linear function $l_q(x)$ that is zero at a vertex q of P and strictly positive at all other vertices of P. $l_q(x)$ then implicitly defines a supporting plane that touches P only at q. Such a function always exists given P is convex. Therefore,

$$l_q(x) = \sum_v l_q(v) b_v(x).$$

Since $l_q(q) = 0$, then

$$l_q(q) = 0 = \sum_{v \neq q} l_q(v) b_v(q)$$

Now each $l_q(v) > 0$ and $b_v(q) \geq 0$. Therefore, $b_v(q) = 0$ for all $v \neq q$. Since the coordinate functions sum to 1, $b_q(q) = 1$.

Using this result and Equation 2, we can build a function $\hat{g}(x)$ that interpolates data values at the vertices of P. Figure 2 illustrates such an example of using discrete 2D barycentric coordinates to interpolate height and color data at the vertices of P. Along the edges of the polygon, these weight functions reduce to linear interpolation. Note that this last remark is a nice property already stressed in [19]: these barycentric coordinates on a *nD* polytope reduces to their lower dimensional $(n-1)$D version on any boundary face.

3 Coordinates for Smooth Convex Sets

As we have seen above, barycentric coordinates on a convex polytope P blend values g_v assigned to the vertices of P to define a function $\hat{g}(x)$ over all of P.

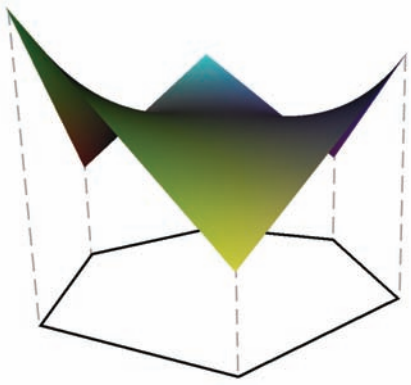

Figure 2: An example using barycentric coordinates to interpolate both the height and the color of the vertices of the hexagon.

In many applications, we would like to perform a similar blending for arbitrary convex shapes. In particular, given a function $g(x)$ defined on the boundary S of P, we desire a method for extending $g(x)$ to the interior of P that generalizes barycentric coordinates from the polytope case.

3.1 A Geometric Expression

Our solution is to construct a continuous barycentric coordinate function defined over all of P. The key to creating such a function $b(v,x)$ that satisfies these properties is to observe that Equation 4 extends to smooth functions in a very natural, geometric manner. In particular, the numerator $\kappa(v)$ in the polytope case form a discrete approximation to the Gaussian curvature at the vertex v while the denominator is the product of the distance of x to the d different facets incident on v.

Following the polytope case, we construct the continuous version of this weight function as

$$w(v,x) = \frac{\kappa(v)}{(n(v)\cdot(v-x))^d} \tag{6}$$

where $\kappa(v)$ is the Gaussian curvature of S at v (i.e; the product of the $d-1$ principal curvatures at v) and $n(v)$ is the outward unit normal to S at v. Figure 3 shows an example calculation of the denominator of the weight function $w(v,x)$.

To complete the construction, we simply need to define a barycentric coordinate function $b(v,x)$ associated with $w(v,x)$ to have the form

$$b(v,x) = \frac{w(v,x)}{\int_{v\in S} w(v,x)dS}. \tag{7}$$

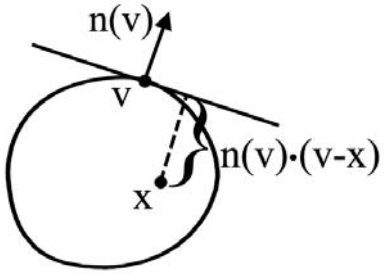

Figure 3: Calculating the denominator of $w(v,x)$ for continuous functions.

After this normalization, these coordinate functions $b(v,x)$ are non-negative and have unit integral. As the central result of this paper, we show in the next section that this function $b(v,x)$ has linear precision.

For strictly convex shapes (those whose supporting half-spaces contact the shape at a single point), a similar argument to the discrete case in Section 2 allows us to conclude that the barycentric coordinate function degenerates to the Dirac delta function on S. That is,

$$b(v,x) = \delta(v-x) \quad \forall x \in S. \tag{8}$$

Therefore, if we compute a new function $\hat{g}(x)$ using equation 3 for this class of shapes, $\hat{g}(v) = g(v)$ holds for all $v \in S$.

3.2 Linear precision

We now prove that the continuous coordinate function $b(v,x)$ defined in Section 1.2 has linear precision, i.e;

$$\int_{v \in S} v\, b(v,x) dS = x.$$

Substituting the definition of $b(v,x)$ in terms of $w(v,x)$ from Equation 7 yields

$$x \int_{v \in S} w(v,x) dS = \int_{v \in S} v\, w(v,x) dS.$$

Combining the two integrals together generates

$$\int_{v \in S} (v-x) w(v,x) dS = 0.$$

If we let $h(v,x)$ denote $(v-x)w(v,x)$, applying Equation 6 yields

$$h(v,x) = (v-x) \frac{\kappa(v)}{(n(v) \cdot (v-x))^d}. \tag{9}$$

Now, our task is to prove that $\int_{v \in S} h(v,x)dS = 0$ for all x. To prove that this integral is zero, we proceed in three steps:

- We express S as an implicitly defined surface $f(x) = 0$ and apply the curvature formula for implicit surfaces to express h in terms of f.
- We then manipulate this expression using several technical lemmas (proven in Appendix B) and derive an equivalent expression for h in terms of cross products and dot products.
- Finally, we convert this cross product into a differential form and apply Stoke's theorem to show that the integral of this differential form over S is zero.

In step one, we observe that the $(d-1)$ dimensional manifold S can be expressed as the solution to $f(x) = 0$ where $f(x)$ defines the signed Euclidean distance from the point x to S. Observe that since $f(x)$ is a distance function, its gradient $\nabla f(x)$ has unit length on S.

In this implicit case, Goldman [7] provides a formula for Gaussian curvature. In particular, the scalar $\kappa(v)$ can be expressed as

$$\kappa(v) = \frac{\nabla f(v) H^*(v) \nabla f^T(v)}{|\nabla f(v)|^{d+1}}$$

where $H(v)$ is the Hessian of $f(v)$ and $()^*$ denotes the adjoint of a square matrix. Recall that in the context of this proof, $|\nabla f(v)| = 1$. Substituting this relation in the left-hand side of Equation 9 yields that

$$h(v,x) = (v-x)\frac{\nabla f(v) H^*(v) \nabla f^T(v)}{(\nabla f(v) \cdot (v-x))^d}. \tag{10}$$

In the second step of our proof, we consider the vector-valued function $m(v,x)$ defined via

$$m(v,x) = \frac{\nabla f(v)}{\nabla f(v) \cdot (v-x)}.$$

In Appendix B, we show that $h(v,x)$ can be equivalently expressed in terms of the gradient of $m(v,x)$ with respect to coordinates of v. (x is treated as constant.) In particular, $h(v,x)$ has the form

$$h(v,x) = (\nabla m(v,x))^* \nabla f^T(v).$$

Now, let $h_k(v,x)$ denote the kth element of $h(v,x)$. Due to the properties of adjoints, $h_k(v,x)$ can expressed as the cross-product of $d-1$ rows of the matrix

$\nabla m(v,x)$ dotted with $\nabla f^T(v)$. In particular,

$$h_k(v,x) = (\bigotimes_{i \neq k} \nabla(m_i(v,x))) \cdot \nabla f^T(v) \tag{11}$$

where $m_i(v,x)$ is the i^{th} component of $m(v,x)$.

In our final step of the proof, we apply a generalized version of Stoke's theorem [2, p.365] of the form:

$$\int_{v \in S} \omega(v) = \int_{v \in P} d(\omega(v))$$

where ω is a differential form and d is the differential operator. To construct ω, we observe that the cross product in equation 11 has an equivalent formulation as a wedge product of differentials. In particular,

$$\int_{v \in S} h_k(v,x) dS = \int_{v \in S} \bigwedge_{i \neq k} d(m_i(v,x)). \tag{12}$$

(The dot product with $\nabla f^T(v)$ in equation 11 is absorbed during the integration of the differential form on the right-hand side of equation 12 [2, p.356].) Now, applying Stoke's theorem yields that

$$\int_{v \in S} h_k(v,x) dS = \int_{v \in P} d(\bigwedge_{i \neq k} d(m_i(v,x))).$$

Due to Lemma 3 from the appendix, the integral on the right-hand side reduces to zero, which completes the proof.

$$\int_{v \in S} (v-x) w(v,x) dS = \int_{v \in S} h_k(v,x) dS = 0.$$

4 Applications to Smooth Convex Sets

We next consider two novel applications of barycentric coordinates over smooth convex sets: boundary value interpolation and freeform deformations. To do so we first specialize Equation 6 to parametric functions since evaluating the resulting integrals becomes much simpler.

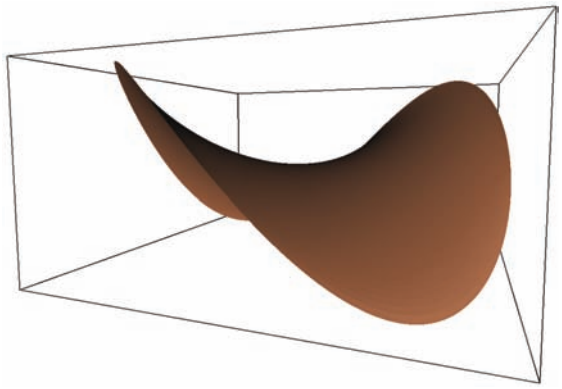

Figure 4: Barycentric interpolation of the function $x_1 x_2$ on the unit circle: notice how the values blend on the interior of the circle in a smooth, natural manner.

4.1 Boundary value interpolation

Given a function $g(v)$ defined over some manifold $v \in S$ of co-dimension one where $v \in S$, we desire a function $\hat{g}(x)$ defined over the interior of S that interpolates $g(v)$ on S. This function $\hat{g}(x)$ can be used to build a surface patch that interpolates a given curve or even to extend functions such as heat distribution over a surface to the interior of the volume.

Equation 3 already defines how to build such a function $\hat{g}(x)$ to interpolate $g(v)$ on S. However, to make this construction practical, we show how to interpret this barycentric coordinate function in a parametric form. In particular, we provide an explicit example of specializing this formula for two-dimensional parametric curves.

Given a region P in two-dimensions bounded by a curve with parameterization $p(t) = (p_1(t), p_2(t))$, the curvature $\kappa(t)$ is given by

$$\kappa(t) = \frac{p_1'(t)p_2''(t) - p_2'(t)p_1''(t)}{(p_1'(t)^2 + p_2'(t)^2)^{\frac{3}{2}}}.$$

Likewise, the unit normal $n(t)$ is simply

$$n(t) = \frac{1}{(p_1'(t)^2 + p_2'(t)^2)^{\frac{1}{2}}}(-p_2'(t), p_1'(t)).$$

Therefore, Equation 6 for the weight function $w(v,x)$ becomes

$$w(t,x) = \frac{p_1'(t)p_2''(t) - p_2'(t)p_1''(t)}{(p_1'(t)^2 + p_2'(t)^2)^{\frac{1}{2}}(-p_2'(t)(p_1(t) - x_1) + p_1'(t)(p_2(t) - x_2))^2}$$

where $x = (x_1, x_2)$. Notice that when integrating a scalar function over a space curve, we must include a factor of $(p_1'(t)^2 + p_2'(t)^2)^{\frac{1}{2}}$ to make the integral invariant under the parameterization of the boundary $p(t)$. For example, the normalization factor for the basis function $b(v,x)$ becomes:

$$\begin{aligned}\int_{v\in S} w(v,x)dS &= \int w(t,x)(p_1'(t)^2 + p_2'(t)^2)^{\frac{1}{2}}dt \\ &= \int \frac{p_1'(t)p_2''(t) - p_2'(t)p_1''(t)}{(-p_2'(t)(p_1(t)-x_1) + p_1'(t)(p_2(t)-x_2))^2}dt.\end{aligned}$$

When applying equation 3, a similar factor appears in the integral to account for the parameterization of the boundary.

To illustrate this formula, consider the problem of interpolating the function x_1x_2 over a patch whose boundary consists of the unit circle. To parameterize the circle let $p(t) = (Cos(t), Sin(t))$. By construction, the weight function $w(t,x)$ has the form

$$w(t,x) = \frac{1}{(x_1 Cos(t) + x_2 Sin(t) - 1)^2}.$$

The corresponding barycentric coordinate function $b(t,x)$ is then

$$b(t,x) = \frac{(1 - x_1^2 - x_2^2)^{\frac{3}{2}}}{2\pi(x_1 Cos(t) + x_2 Sin(t) - 1)^2}.$$

To construct a function $\hat{g}(x)$ that interpolates the function x_1x_2 on the unit circle, we must build a function $g(t)$ parameterized over the boundary that interpolates x_1x_2. Notice that $g(t) = Cos(t)Sin(t)$ since $(x_1, x_2) = (Cos(t), Sin(t))$. Now Equation 3 can be computed analytically and has the form

$$\hat{g}(x) = \frac{x_1 x_2 \left(-2 + 3{x_1}^2 + 3{x_2}^2 + 2\left(1 - {x_1}^2 - {x_2}^2\right)^{\frac{3}{2}}\right)}{\left({x_1}^2 + {x_2}^2\right)^2}.$$

Figure 4 shows a plot of this function restricted to the unit circle. Observe that the function $\hat{g}(x)$ interpolates the function $x_1 x_2$ on the unit circle while blending these values on the interior of the circle in a natural manner.

4.2 Freeform deformations

Continuous barycentric coordinates can be used to perform freeform deformations of images as well. Given a convex region P bounded by a smooth curve $p(t)$, we wish to deform P into another region G bounded by the curve $g(t)$ (see Figure 5). If we use the construction from equation 3 we obtain:

$$\hat{g}(x) = \int_v g(t)b(t,x)dt$$

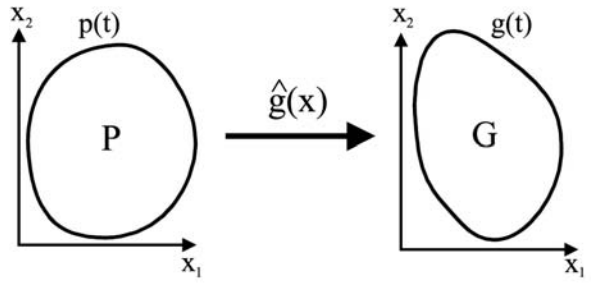

Figure 5: $\hat{g}(x_1, x_2)$ provides a map between $p(t)$ and $g(t)$. This map can be used to perform freeform deformations.

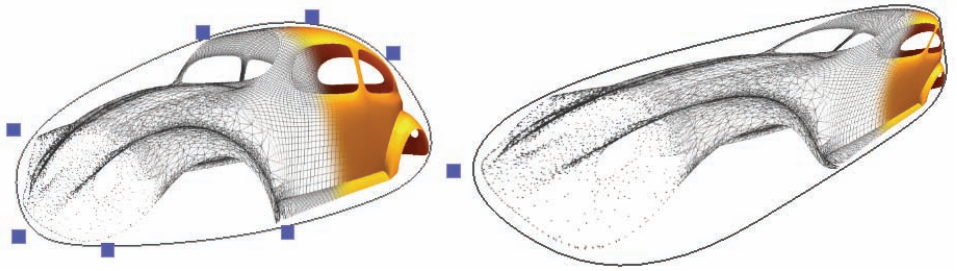

Figure 6: Car before deformation and bounding quadratic B-spline curve defining $p(t)$ (left). Deformed car generated by altering the control points with bounding curve $g(t)$ (right).

where $x \in P$. The resulting function $\hat{g} : P \rightarrow G$ smoothly maps points in P to points in G. Furthermore, $\hat{g}(x)$ maps points on $p(t)$ to points on $g(t)$, that is, $\hat{g}(p(t)) = g(t)$. Since $b(v,x)$ has linear precision, if $p(t) = g(t)$, then $\hat{g}(x)$ becomes the identity function and generates no deformation.

In our example, we define P and G as the regions bounded by closed quadratic B-splines $p(t)$ and $g(t)$ having k control points on the periodic interval $0 \leq t \leq k$. Though B-splines are only piecewise polynomial, equation 3 still applies. In fact, any B-spline curve can be represented as a piecewise polynomial function of the form

$$\begin{array}{rcl} p(t) & = & p_i(t-i) \\ g(t) & = & g_i(t-i) \end{array}, \quad i \leq t \leq i+1$$

where $p_i(t)$, $g_i(t)$ are the i^{th} polynomial functions comprising the respective B-splines.

To compute Equation 3 we construct $w(t,x)$, which is also a piecewise function, and has the form

$$w(t,x) = w_i(t-i,x), \quad i \le t \le i+1$$

where $w_i(t,x)$ is formed using Equation 6 for the function $p_i(t)$. With this result we can calculate the normalization factor in Equation 7 as

$$\int w(t,x)dt = \sum_{i=0}^{k-1} \int_0^1 w_i(t,x)dt.$$

Now we compute $\hat{g}(x)$ using Equation 3 as a piecewise integral that has the form

$$\begin{aligned} \hat{g}(x) &= \frac{1}{\int w(t,x)dt} \int g(t)w(t,x)dt \\ &= \frac{1}{\int w(t,x)dt} \sum_{i=0}^{k-1} \int_0^1 g_i(t)w_i(t,x)dt. \end{aligned}$$

We can explicitly calculate the integrals above, using a symbolic software package such as *Mathematica*, to obtain a closed form solution in terms of (x_1,x_2) and the control points of the B-splines forming $p(t)$ and $g(t)$. Though each $w_i(t,x)$ is a rational polynomial function, the resulting $\hat{g}(x)$ is more complicated and is in terms of functions such as *Arctan*. However, the function is still fast to evaluate (since no integrals need be computed) and image deformation can be computed in realtime.

The user performs image deformation by first placing the control points of the curve $p(t)$ about the convex area that they wish to deform (see Figure 6, left). Once the user is satisfied, the control points are duplicated to form the curve $g(t)$. The user then drags on the control points of $g(t)$ to generate the desired deformation. Due to the fact that barycentric coordinates interpolate the boundary (as shown in Equation 8), the deformed image will follow the boundary of $g(t)$. Figure 6 (right) shows an example deformation of the car from the left portion of the figure. The entire application and source code for performing these deformations can be downloaded from `http://www.cs.rice.edu/~sschaefe/barywhite.zip`.

5 Conclusion

In this paper, we have provided an extension of barycentric coordinates first to convex polytopes, then to smooth convex sets in arbitrary dimension, both in the form of explicit, geometric formulas. Furthermore, we provided a proof that the coordinate functions are non-negative, have unit integral and reproduce linear functions. Finally, we showed how this function could be used to build solutions to boundary value problems and perform deformations as well.

Of special interest is the similarity that our barycentric coordinate function bears towards Green's functions. Green's functions are typically used in a similar manner to our barycentric coordinate functions to build solutions to (typically elliptical) partial differential equations with boundary value constraints. While our function builds solutions to boundary value problems, we know of no differential equation that the functions satisfy. In the future, we would like to explore this connection further. A potential approach could be to leverage the geometric interpretation of these coordinates in terms of polar duals as recently introduced in [10].

Acknowledgements We would like to thank Ron Goldman for his helpful discussions, and for his help in creating the proof for Lemma 2.

References

[1] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Interactive Multiresolution Surface Viewing. In *ACM Siggraph'95 Conference*, pages 91–98, August 1995.

[2] W. Fleming. *Functions of Several Variables*. Springer-Verlag, 1977.

[3] M. Floater. Parametrization and smooth approximation of surface triangulations. *CAGD*, 14(3):231–250, 1997.

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

[5] M. Floater, K. Hormann, and G. Kós. A general construction of barycentric coordinates over convex polygons. *To appear in Advances in Computational Mathematics*, 2004.

[6] M. Floater, G. Kos, and M. Reimers. Mean Value Coordinates in 3D. *CAGD*, 2005.

[7] R. Goldman. Curvature formulas for implicit curves and surfaces. In *Workshop on Geometric Modeling and Differential Geometry*, October 2003.

[8] C. Gotsman and V. Surahhsky. Guaranteed Intersection-free Polygon Morphing. *Computer and Graphics*, 25(1):67–75, 2001.

[9] Tao Ju, Scott Schaefer, and Joe Warren. Mean Value Coordinates for Closed Triangular Meshes. In *ACM Trans. on Graphics (SIGGRAPH Proceedings)*, pages 561–566, 2005.

[10] Tao Ju, Scott Schaefer, Joe Warren, and Mathieu Desbrun. A geometric Construction of Coordinates for Convex Polyhedra using Polar Duals. In *ACM/EG Symposium on Geometry Processing*, pages 181–186, 2005.

[11] S. Kuriyama. Surface Generation from an Irregular Network of Parametric Curves. *Modeling in Computer Graphics, IFIP Series on Computer Graphics*, pages 256–274, 1993.

[12] S. Lodha. Filling N-sided Holes. *Modeling in Computer Graphics, IFIP Series on Computer Graphics*, pages 319–345, 1993.

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

[14] E. Malsch. *Test functions for elliptic operators satisfying essential edge conditions on both convex and concave polygonal domains*. PhD thesis, Columbia University, 2003.

[15] M. Meyer, H. Lee, A. Barr, and M. Desbrun. Generalized Barycentric Coordinates for Irregular Polygons. *Journal of Graphics Tools*, 7(1):13–22, 2002.

[16] U. Pinkall and K. Polthier. Computing Discrete Minimal Surfaces and Their Conjugates. *Experimental Math.*, 2:15–36, 1993.

[17] R. Sibson. A brief description of natural neighbor interpolation. In V. Barnett, editor, *Interpreting Multivariate Data*, pages 21–36. John Wiley & Sons, 1981.

[18] E. Wachpress. A Rational Finite Element Basis. *Manuscript*, 1975.

[19] J. Warren. Barycentric Coordinates for Convex Polytopes. *Advances in Computational Mathematics*, 6:97–108, 1996.

[20] J. Warren. On the uniqueness of barycentric coordinates. Proceedings of the Vilnius Workshop on Algebraic Geometry and Geometric Modeling, 2002.

A Linear Precision for Convex Polytopes

As observed in the paper, proving that the coordinate functions $b_v(x)$ have linear precision reduces to showing that Equation 1 holds; that is the weight functions $w_v(x)$ satisfy:

$$\sum_v (v-x) w_v(x) = 0.$$

To this end, we observe that at a simple vertex v of P, the following *vector* relationship holds:

$$\frac{n_{ind(v)}(v-x)}{\prod\limits_{j\in ind(v)}(n_j\cdot(v-x))} = \sum_{k\in ind(v)} \frac{e_k}{\prod\limits_{\substack{q\in ind(v)\\ q\neq k}}(n_q\cdot(v-x))}$$

where e_k is the k-th canonical basis vector in d dimensions. Multiplying the numerator of both sides of this equation by $n^{-1}_{ind(v)}$, the resulting equation has the form:

$$\frac{v-x}{\prod\limits_{j\in ind(v)}(n_j\cdot(v-x))} = \sum_{k\in ind(v)} \frac{n^{-1}_{ind(v)}e_k}{\prod\limits_{\substack{q\in ind(v)\\ q\neq k}}(n_q\cdot(v-x))}. \tag{13}$$

Now, recall that the k-th column of $n^{-1}_{ind(v)}$ corresponds to the cross product of the $d-1$ rows of $n_{ind(v)-k}$ (denoted $Cross(n_{ind(v)-k})$ below) divided by the determinant of $n_{ind(v)}$. Applying this observation and multiplying both sides of equation 13 by $Det(n_{ind(v)})$ yields that

$$(v-x)\frac{Det(n_{ind(v)})}{\prod\limits_{j\in ind(v)}(n_j\cdot(v-x))} = \sum_{k\in ind(v)} \frac{Cross(n_{ind(v)-k})}{\prod\limits_{\substack{q\in ind(v)\\ q\neq k}}(n_q\cdot(v-x))} \tag{14}$$

Note that each of the cross products in this last equation corresponds to a vector lying parallel to an edge of P incident to v. Taking the sum of both sides of Equation 14 over all v of P yields:

$$\sum_{v\in P}\frac{(v-x)\,Det(n_{ind(v)})}{\prod\limits_{j\in ind(v)}(n_j\cdot(v-x))} = \sum_{v\in P}\sum_{k\in ind(v)} \frac{Cross[n_{ind(v)-k}]}{\prod\limits_{\substack{q\in ind(v)\\ q\neq k}}(n_q\cdot(v-x))}. \tag{15}$$

Now, we assume (without loss of generality) that the indices in $ind(v)$ are ordered such that the determinant of $n_{ind(v)}$ is always positive. Since each edge of P is shared by two vertices of P, the cross product on the right-hand side of equation 15 appears twice in the double summation, once for each possible orientation of the edge. Since these vector then cancel, the left hand side of equation 15 must be identically zero. Observing that the fraction on the left-hand side of this same equation is exactly the weight function $w_v(x)$ defined by equation 4 completes the proof.

B Linear Precision for Smooth Convex Sets

Recall that $h(v,x)$ from section 3.2 is written as

$$h(v,x) = (v-x)\frac{\nabla f(v) H^*(v) \nabla f^T(v)}{(\nabla f(v)\cdot(v-x))^d}$$

where $H(x)$ is the Hessian of $f(x)$, $()^*$ denotes the adjoint of a square matrix, $\nabla f(v)$ is a row vector and x, v are column vectors.

Theorem 1:

$$h(v,x) = (\nabla m(v,x))^* \nabla f^T(v)$$

where $m(v,x) = \frac{\nabla f(v)}{\nabla f(v)(v-x)}$.

Proof: Since

$$(\nabla m(v,x))^* \nabla f^T(v) = (\nabla(\frac{\nabla f^T(v)}{\nabla f(v)(v-x)}))^* \nabla f^T(v),$$

we expand the gradient of our matrix using the product rule and obtain

$$\left(\frac{H(v)((\nabla f(v)(v-x))I - (v-x)\nabla f(v)) - \nabla f^T(v)\nabla f(v)}{(\nabla f(v)(v-x))^2}\right)^* \nabla f^T(v).$$

Applying lemma 1 reduces this expression to

$$\left(\frac{H(v)((\nabla f(v)(v-x))I - (v-x)\nabla f(v))}{(\nabla f(v)(v-x))^2}\right)^* \nabla f^T(v).$$

Next, we use the product rule for adjoints to rewrite the adjoint as

$$\frac{1}{(\nabla f(v)(v-x))^{2(d-1)}}((\nabla f(v)(v-x))I - (v-x)\nabla f(v))^* H^*(v)\nabla f^T(v).$$

Simplifying using lemma 2 yields

$$\frac{1}{(\nabla f(v)(v-x))^{2(d-1)}}(\nabla f(v)(v-x))^{d-2}(v-x)\nabla f(v) H^*(v)\nabla f^T(v).$$

Canceling the appropriate powers generates the final form and completes the proof

$$\frac{1}{(\nabla f(v)(v-x))^{d)}}(v-x)\nabla f(v) H^*(v)\nabla f^T(v).$$

Lemma 1: $aB^* = a(B + c^T a)^*$ where a, c are row vectors with n entries.

Proof: The proof follows by properties of the determinant and is simply and application of Kramer's rule. The i^{th} entry in the vector aB^* can be found as determinant where the i^{th} row in B is replaced by a. Therefore, if we consider the vector $a(B+c^Ta)^*)$, the i^{th} entry in this vector is given by the determinant of $B+c^Ta$ with the i^{th} row replaced with a. Since c^Ta adds a scalar multiple of a to each row of B, c^Ta is linearly dependent on a. Therefore, c^Ta will not contribute to the determinant and $aB^* = a(B+c^Ta)^*$.

Lemma 2: $a^Tb(ab^T)^{n-2} = ((ab^T)I - a^Tb)^*$ where a,b are row vectors and B is a matrix.

Proof: The adjoint is a matrix of cofactors, each of which is a determinant of a sub-piece of the matrix. Therefore, we instead prove a result about determinates of matrices of the given form that trivially extends to yield the adjoint rule above.

Claim: $Det(\alpha I - a^Tb) = \alpha^n - \alpha^{n-1}Tr(a^Tb)$ and $Det(\alpha I_1 - a^Tb) = -\alpha^{n-1}a_1b_1$ where α is a scalar, I is the identity matrix, I_i is the identity matrix with the i^{th} row uniformly zero.

Proof: The proof is inductive on the size of the vectors (n).

Base Case: $n = 1$

$$Det(\alpha - a_1b_1) = \alpha^1 - \alpha^0 a_1b_1 = \alpha - a_1b_1$$

$$Det(\alpha I_1 - a_1b_1) = -\alpha^0 a_1b_1 = -a_1b_1$$

Inductive Step:

Starting with the first identity, we assume that a,b are vectors with n entries and expand the determinant in terms of a sum of determinants of size $n-1$.

$$Det(\alpha I - a^Tb) = (\alpha - a_1b_1)Det((a^Tb)_{1,1}) - \sum_{i=2}^{n}(-1)^{i-1}a_1b_iDet((a^Tb)_{1,i})$$

where $M_{i,j}$ yields the matrix M with row i and column j deleted. Using the inductive hypothesis and rearranging rows in the matrices we obtain

$$(\alpha - a_1b_1)(\alpha^{n-1} - \alpha^{n-2}\sum_{i=2}^{n}a_ib_i) - \sum_{i=2}^{n}(-1)^{i-1}a_1b_i(\alpha^{n-2}a_ib_1(-1)^{i-1}).$$

Simplifying this expression yields the final result

$$\alpha^n - \alpha^{n-1}\sum_{i=1}^{n}a_ib_i.$$

The second recurrence follows in a similar manner. First, we rewrite the determinant as a sum of determinants of size $n-1$

$$Det(\alpha I_1 - a^T b) = -a_1 b_1 Det((a^T b)_{1,1}) - \sum_{i=2}^{n} (-1)^{i-1} a_1 b_i Det((a^T b)_{1,i}).$$

Applying the inductive hypothesis generates

$$-a_1 b_1 (\alpha^{n-1} - \alpha^{n-2} \sum_{i=2}^{n} a_i b_i) - \sum_{i=2}^{n} (-1)^{i-1} a_1 b_i (\alpha^{n-2} a_i b_1 (-1)^{i-1}).$$

Finally, simplifying this expression yields the desired result

$$-\alpha^{n-1} a_1 b_1.$$

Lemma 3: $d(\bigwedge_{i=1..n} d(f_i)) = 0$ for all $f_1 ... f_n$.

Proof: The proof is inductive on the number of entries n.

Base Case: $n = 1$. $d(d(f_1)) = 0$ since the derivative of the derivative of a differential form is zero.

Inductive Step: First, we use the product rule for wedge products [2, p.292–293] to expand out $d(d(f_1) \wedge (\bigwedge_{i=2..n} d(f_i)))$ to

$$d(d(f_1)) \wedge (\bigwedge_{i=2..n} d(f_i))) - d(f_1) \wedge d(\bigwedge_{i=2..n} d(f_i))).$$

Using the fact that $d(d(f_1)) = 0$ and the inductive hypothesis, we simplify this expression to

$$0 \wedge (\bigwedge_{i=2..n} d(f_i))) - d(f_1) \wedge 0 = 0.$$