Kai Hormann, N. Sukumar

Generalized Barycentric Coordinates in Computer Graphics and Computational Mechanics

Contents

S-Patches

Scott Schaefer

Texas A&M University

CONTENTS

CHAPTER **??** introduced several constructions of barycentric coordinates as well as their properties. In this chapter, we explore the deep connection between barycentric coordinates and higher order parametric representations of curves, surfaces, volumes in arbitrary dimension. In the case of curves, these curves are known as Bézier curves, which are used in applications from font representations to controlling animations. The extension to convex surface patches, called S-Patches [7], is more recent. As originally proposed, S-Patches were parametric, multi-sided surface patches restricted to convex domains. However, these restrictions were more of a function of the limited set of generalized barycentric coordinates, namely Wachspress coordinates, available at that time. Today we have generalized barycentric coordinate functions that do not require convexity and extend to arbitrary dimension. Hence, we will investigate S-Patches within their full generality afforded by modern barycentric coordinates with generalized domains and in arbitrary dimension.

1.1 INTRODUCTION

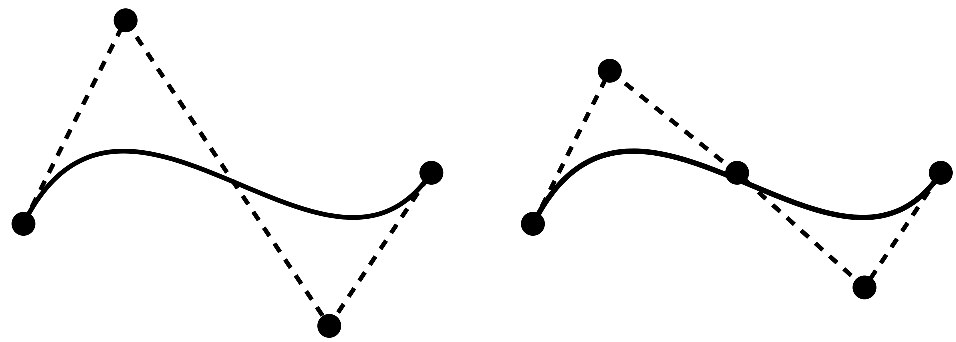

Figure 1.1 An example cubic Bézier curve (left) and the curve degree elevated to a quartic (right).

1.1.1 Bézier Form of Curves

To begin we consider one of the simplest instantiations of barycentric coordinates. Consider the domain defined by the interval $[0, 1]$. This interval yields barycentric coordinate functions ϕ_i where

$$\begin{aligned} \phi_0(x) &= (1-x), \\ \phi_1(x) &= x. \end{aligned} \tag{1.1}$$

Note that these functions satisfy all of the barycentric coordinate properties from Chapter **??**; in particular, these functions reproduce constant and linear functions,

$$\begin{aligned} \phi_0(x) + \phi_1(x) &= 1, \\ 0 \cdot \phi_0(x) + 1 \cdot \phi_1(x) &= x. \end{aligned}$$

If we examine the terms of the binomial expansion of $(\phi_0(x) + \phi_1(x))^m$, we obtain functions of the form

$$B_j^m(x) = \binom{m}{j}(1-x)^{m-j}x^j, \tag{1.2}$$

where m is the degree of the functions $B_j^m(x)$. These functions are special functions called Bernstein basis functions. Associating these functions with control points f_j yields a Bézier curve

$$F(x) = \sum_{j=0}^{m} B_j^m(x) f_j.$$

Note that $F(x)$ trivially reproduces constant functions due to the fact that ϕ_0 and ϕ_1 form a partition of unity. For example, if $f_j = c$, then

$$c = \sum_{j=0}^{m} B_j^m(x)c = c((1-x)+x)^m.$$

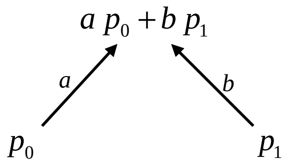

Figure 1.2 An example pyramid diagram. The values at the base are multiplied by the constants on the arrows and summed to produce the result at the apex of the pyramid.

In addition, $F(x)$ can also reproduce all polynomial functions up to degree m. This last property follows directly from the linear reproduction property of barycentric coordinates. In particular, to reproduce a function x^k where $k \leq m$ there exists coefficients f_j such that

$$F(x) = (\phi_0(x) + \phi_1(x))^{m-k}(0 \cdot \phi_0(x) + 1 \cdot \phi_1(x))^k = 1^{m-k}x^k = x^k$$

where the coefficients f_j are given by collecting the coefficients of $B_j^m(x)$ in the polynomial expansion above. For example, to reproduce the function x, the coefficients are given by $f_j = \frac{j}{m}$. Such properties also follow directly from the theory of blossoming/polar forms [8], which is beyond the scope of this chapter.

In addition, the barycentric coordinate functions ϕ_i also impart a number of useful geometric properties on the resulting Bézier curves. For example, Bézier curves interpolate their end points due to the Lagrange property of barycentric coordinates. Bézier curves also fall within the convex hull of their control points over the interval $[0,1]$ since $0 \leq \phi_i(x) \leq 1$ for all $x \in [0,1]$. The curves are also affinely invariant; that is, transforming each control point by an affine transformation T transforms the Bézier curve by T. Figure 1.1 shows an example of a Bézier curve where the f_j are points in $\mathbb{R}^2$.

1.1.2 Evaluation

While we can evaluate Bézier curves by evaluating the polynomial expressions in the Bernstein basis functions, a more elegant solution exists via de Casteljau's algorithm. To do so, we introduce a graphical notation for a linear combination of two control points. Figure 1.2 shows a simple pyramid diagram. The arrows denote taking the product of the value at the base of the arrow with the scalar value listed along the arrow. The result of this product is then added to the sum at the end of the arrow. Hence, this figure denotes taking f_0, f_1 and multiplying these values by a, b respectively to form the result $af_0 + bf_1$. Using this notation, de Casteljau's algorithm can be written in a very elegant fashion as a pyramid diagram shown in Figure 1.3 [3]. Note that each level of the pyramid produces lower order Bézier functions with the value of the curve appearing at the apex of the pyramid.

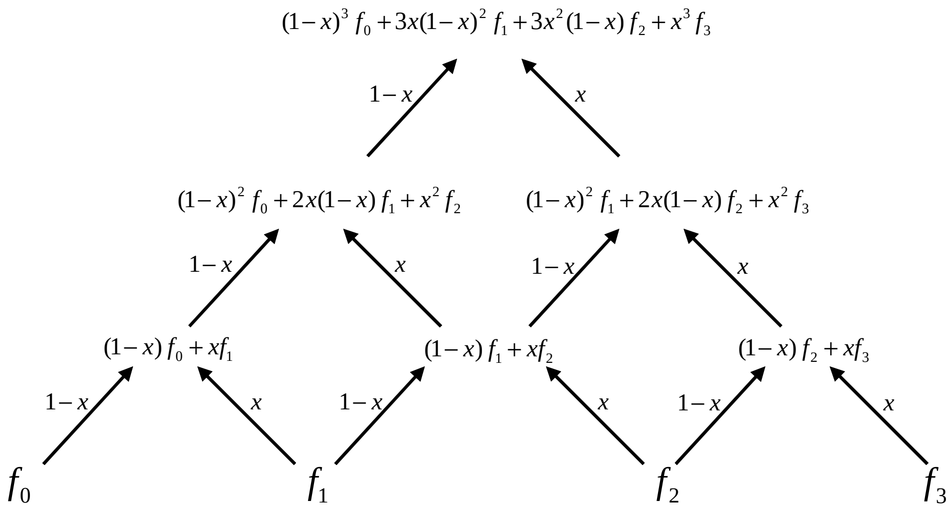

Figure 1.3 The de Casteljau algorithm for cubic Bézier curves.

1.1.3 Degree Elevation

Degree elevation for Bézier curves is simply finding a Bézier curve of degree $m+1$ to represent a Bézier curve of degree m. One benefit of performing degree elevation is that the process introduces an additional control point that can be used to control the shape of the curve. Furthermore, the addition of this new control point does not change the shape of the curve. Such degree elevation is always possible since every Bézier curve of degree m is also a Bézier curve of degree $m+1$.

To derive degree elevation, it is useful to notice a connection between the Bernstein basis functions of different degrees. Using Equation (1.2), we can show that

$$B_j^{m+1}(x) = \frac{m+1}{m+1-j}\phi_0(x)B_j^m(x) = \frac{m+1}{j}\phi_1(x)B_{j-1}^m(x).$$

Therefore, we can elevate the degree of a Bézier curve by simply multiplying a degree m Bézier curve by $(\phi_0(x)+\phi_1(x))$, so that

$$\begin{aligned} F(x) &= \left(\sum_{j=0}^m B_j^m(x) f_j\right)(\phi_0(x)+\phi_1(x)) \\ &= \sum_{j=0}^{m+1} B_j^{m+1}(x)\left(\tfrac{j}{m+1} f_{j-1} + \tfrac{m+1-j}{m+1} f_j\right). \end{aligned}$$

Figure 1.1 (right) shows an example of elevating the degree of a Bézier curve.

1.2 MULTISIDED BEZIER PATCHES IN HIGHER DIMENSIONS

Section 1.1.1 provides some hint as to how we might extend the univariate curve construction to domains such as polygons or even polytopes in higher dimensions through the use of generalized barycentric coordinates. In particular, we use the extension of Equation (1.1) to more generalized domains. Given a polygon P with vertices $v_i \in \mathbb{R}^d$, the generalized barycentric coordinate functions ϕ_i satisfy

$$\begin{array}{rcl} \sum_{i=1}^{n} \phi_i(\boldsymbol{x}) & = & 1, \\ \sum_{i=1}^{n} \phi_i(\boldsymbol{x}) v_i & = & \boldsymbol{x} \end{array} \tag{1.3}$$

for all points $\boldsymbol{x} \in \mathbb{R}^d$ (although $\boldsymbol{x}$ may be restricted to P for some barycentric coordinate constructions).

To build S-Patches, we examine the functions that arise from the multinomial expansion of the barycentric basis functions

$$\left(\sum_{i=1}^{n} \phi_i(\boldsymbol{x}) \right)^m = \sum_{|\boldsymbol{\ell}|=m} \binom{m}{\boldsymbol{\ell}} \prod_{i=1}^{n} \phi_i(\boldsymbol{x})^{\boldsymbol{\ell}_i}$$

where the index $\boldsymbol{\ell}$ is a vector of n non-negative integers, $|\boldsymbol{\ell}|$ is the sum of the entries of $\boldsymbol{\ell}$, and $\binom{m}{\boldsymbol{\ell}}$ is the multinomial coefficient $\binom{m}{\boldsymbol{\ell}} = \frac{m!}{\boldsymbol{\ell}_1!\boldsymbol{\ell}_2!\ldots\boldsymbol{\ell}_n!}$. Setting $B^m_{\boldsymbol{\ell}}(\boldsymbol{x})$ to be the corresponding term in this expansion yields

$$B^m_{\boldsymbol{\ell}}(\boldsymbol{x}) = \binom{m}{\boldsymbol{\ell}} \prod_{i=1}^{n} \phi_i(\boldsymbol{x})^{\boldsymbol{\ell}_i}. \tag{1.4}$$

These basis functions are the generalization of the Bézier basis functions from Section 1.1.1. In fact, if we use the barycentric basis functions from Equation (1.1), we obtain the exact same curves albeit with a different indexing scheme. If we associate values $f_{\boldsymbol{\ell}}$ with each of these basis functions, we obtain a multi-sided Bézier function

$$f(x) = \sum_{|\boldsymbol{\ell}|=m} B^m_{\boldsymbol{\ell}}(x) f_{\boldsymbol{\ell}},$$

which is also known as an S-Patch. Like univariate Bézier curves, S-Patches can reproduce all polynomials up to total degree m, which follows directly from Equation (1.3). For example, to reproduce $\boldsymbol{x}$, the control points are given by

$$f_{\boldsymbol{\ell}} = \sum_{i=1}^{n} \frac{\boldsymbol{\ell}_i}{m} v_i. \tag{1.5}$$

Unlike univariate curves, the barycentric coordinates functions in Equation (1.3) are not necessarily polynomials. Nevertheless, we will refer to the S-Patch basis functions $B^m_{\boldsymbol{\ell}}(\boldsymbol{x})$ to be a function of degree m indicating the total degree of the individual barycentric coordinates functions ϕ_i.

1.2.1 Indexing For S-Patches

For curves, indexing control points is trivial as each basis function is simply given an index $j = 0, \ldots, m$ based on an ordering of the Bernstein basis functions. However, this simple indexing scheme does not extend to higher dimensions. As already

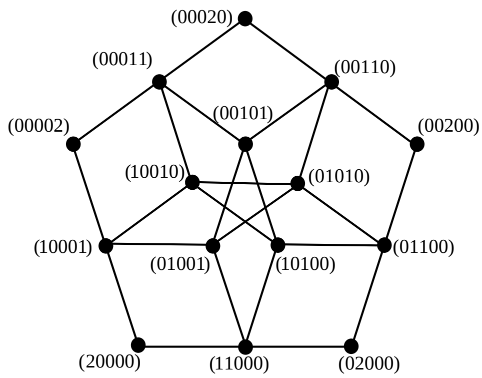

Figure 1.4 Indexing of a quadratic pentagonal S-Patch using the specified connectivity rule.

alluded to in Equation (1.4), S-Patches use a multi-index to refer to control points. Each control point $f_{\boldsymbol{\ell}}$ is associated with an index vector of length n non-negative integers whose sum is the degree m of the patch. Given a polygon P, we typically draw the control points in canonical position such that $f_{\boldsymbol{\ell}}$ are placed such that $F(\boldsymbol{x}) = \boldsymbol{x}$. Luckily, this combination is solely a function of P, independent of what barycentric basis we choose, and is given by Equation (1.5).

In addition, there is a simple rule for drawing connectivity of a simple 2D S-Patch. We connect two control points $f_{\boldsymbol{\ell}}$ and $f_{\boldsymbol{h}}$ if there exists an index $0 \leq j \leq n$ such that

$$\begin{aligned} \boldsymbol{\ell}_i &= \boldsymbol{h}_i \quad i \neq j, j+1 \\ \boldsymbol{\ell}_{j+1} - 1 &= \boldsymbol{h}_j \\ \boldsymbol{\ell}_j + 1 &= \boldsymbol{h}_j \end{aligned}$$

where the arithmetic on j is performed modulo n. Figure 1.4 shows an example of this indexing for a quadratic pentagonal S-Patch. While drawing this connectivity makes sense for 2D domains, S-Patches extend beyond 2D to polytopes in higher dimensions. There, the cyclic ordering of the indices used in 2D to identify connectivity does not generalize to higher dimensions. While we can generalize the connectivity rules to higher dimensions, the information imparted by the connectivity in higher dimensions becomes more difficult to discern. This connectivity is not essential to evaluation algorithms or other properties of S-Patches in higher dimensions, so we simply omit it.

Note that something curious happens to the S-Patch control points for certain polygons. Figure 1.5 shows an example of two quadratic, quadrilateral patches. One example has 10 control points while the other appears to have 9 control points. The issue is that two of the control points corresponding to the (1010) and (0101) index overlap according to Equation (1.5) in the left example and not the right example. To reproduce polynomial functions, these control points must have the same value, which leads to a loss of degrees of freedom. However, in the more

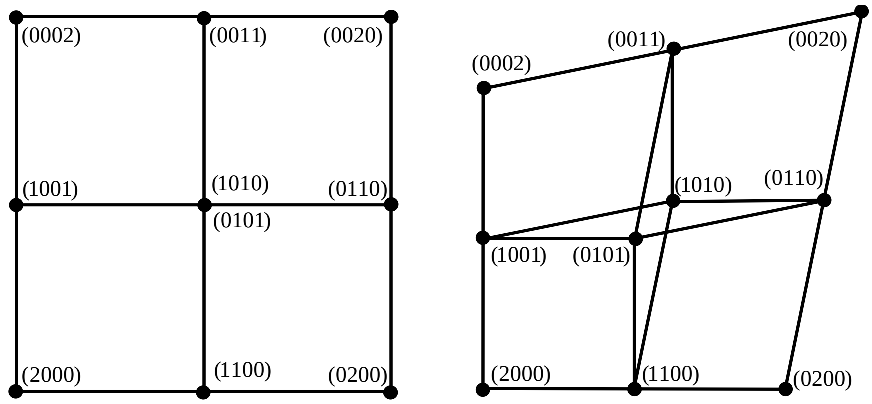

Figure 1.5 An example of ambiguous indexing for S-Patches. The left shows an example where two control points overlap in the parametric domain. The right shows the same indexing for a different base polygon where the overlapping control points separate.

general case of modeling shapes with S-Patches, these two control points may have different values despite being drawn in the same location. It is useful to note that, in the situation on the left of the figure, if we require the overlapping control points to have the same function value, which effectively reduces the number of control points to 9, the S-Patch is identical to a tensor-product Bézier patch when using Wachspress coordinates as the barycentric coordinate functions. Therefore, tensor product Bézier patches as well as triangular Bézier patches (S-Patches with a simplicial domain) are all special cases of S-Patches [7]. Indeed, the basis functions for Wachspress coordinates are identical for the (1010) and (0101) control points in this example. However, the basis functions for two overlapping control points do not have to be identical and, in fact, rarely are in the general case.

1.2.2 Evaluation

Similar to Bézier curves, one possible method for evaluating an S-Patch is simply to multiply the control points by the corresponding basis functions from Equation (1.4) evaluated at the point in question. However, there also exists a de Casteljau-like algorithm that utilizes the hierarchical relationship between the control points. Given an evaluation point $\boldsymbol{x}$, we compute the value of the barycentric basis functions $\phi_i(\boldsymbol{x})$ at the evaluation point using the polytope P. Now, we proceed in a recursive fashion. Let $f_{\boldsymbol{j}}$ be the control points at level $k-1$ where $|\boldsymbol{j}| = k-1$ and $f_{\boldsymbol{\ell}}$ with $|\boldsymbol{\ell}| = m$ be the initial control points. Then, we compute the $f_{\boldsymbol{j}}$ for all $|\boldsymbol{j}| = k-1$

$$f_{\boldsymbol{j}} = \sum_{i=1}^{n} \phi_i(\boldsymbol{x}) f_{\boldsymbol{j}+I_i^n}$$

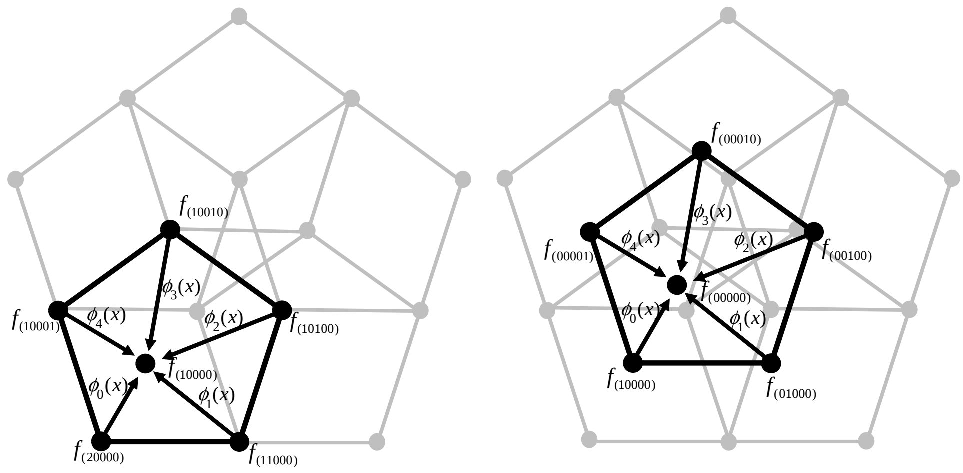

Figure 1.6 The de Casteljau algorithm applied to a quadratic, pentagonal S-Patch. The left image depicts the calculation for one point of the first level of the algorithm. The right image shows the computation of the final evaluation point.

where I_i^n is the i^{th} row of the $n \times n$ identity matrix. The value of the S-Patch is then given by $f_{\mathbf{0}}$. Figure 1.6 depicts this hierarchical evaluation algorithm.

1.2.3 Degree Elevation

While Loop et al. [7] mention degree elevation for S-Patches, the authors provide no explicit formula. However, degree elevation is not difficult to derive. Let $f_{\boldsymbol{\ell}}$ be the values of the control points for an S-Patch function of degree $|\boldsymbol{\ell}| = m$ and $\hat{f}_{\boldsymbol{j}}$ be control points of the same S-Patch function of degree $|\boldsymbol{j}| = m+1$. Then

$$\left(\sum_{|\boldsymbol{\ell}|=m} B_{\boldsymbol{\ell}}^m(\boldsymbol{x}) f_{\boldsymbol{\ell}} \right) \left(\sum_{i=1}^{n} \phi_i(\boldsymbol{x}) \right) = \sum_{|\boldsymbol{j}|=m+1} B_{\boldsymbol{j}}^{m+1}(\boldsymbol{x}) \hat{f}_{\boldsymbol{j}}. \tag{1.6}$$

Note that the basis functions of various degrees are related to each other via

$$B_{\boldsymbol{j}}^{m+1}(\boldsymbol{x}) = B_{\boldsymbol{j}-I_i^n}^{m}(\boldsymbol{x}) \frac{m+1}{\boldsymbol{j}_i} \phi_i(\boldsymbol{x}) \tag{1.7}$$

for $\boldsymbol{j}_i > 0$. Expanding the left-hand side of Equation (1.6) and using Equation (1.7) yields the formula for the $m+1$ degree control points $\hat{f}_{\boldsymbol{j}}$,

$$\hat{f}_{\boldsymbol{j}} = \sum_{\boldsymbol{j}_i > 0} \frac{\boldsymbol{j}_i}{m+1} f_{\boldsymbol{j}-I_i^n}. \tag{1.8}$$

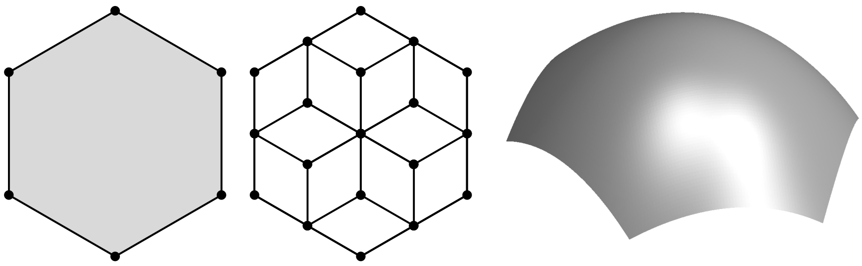

Figure 1.7 An example of an S-Patch with a convex domain. From left to right: the base polygon P shaded gray, the control point structure for the quadratic S-Patch, and an example S-Patch in 3D.

1.3 APPLICATIONS

While the original definition of S-Patches [7] used convex, multi-sided polygon domains, this restriction was more a function of the barycentric coordinates available at that time [11] than any limitation of the construction. Many different types of barycentric coordinates have been developed since then (see Chapter ??). The majority of these constructions are not limited to a particular dimension for the domain and do not require convex domains. While any of these constructions can be used to create S-Patches, we utilize Mean Value Coordinates [1, 2, 4, 6] here in these examples.

1.3.1 Surface patches

Perhaps the most commonly used application of S-Patches is to fill multi-sided holes in surfaces. Figure 1.7 shows an example of such a multi-sided patch using a convex domain. Due to the interpolatory properties of barycentric coordinates, the curves along the boundary of the domain are solely a function of the control points along that boundary and form a Bézier curve. In this case, the six-sided patch is bounded by six quadratic Bézier curves.

Given that the original definition of S-Patches [7] used Wachspress coordinates [11], the S-Patch domain was required to be convex. Convexity is no longer a requirement with many barycentric coordinate constructions, though the misconception of this requirement for S-Patches persists. Figure 1.8 (right) shows an example of a quadratic S-Patch with a concave domain. Figure 1.8 (middle) shows a 2D image of the control point structure of this patch. Unlike convex domains, the control points of S-Patches with concave domains may lie outside of the original polytope.

S-Patches can be formed with domains that are even more general than these convex or concave examples. In fact, S-Patches can support domains of nearly arbitrary shape that may even contain one or more holes. The restrictions on the domain follow directly from the barycentric coordinates used to construct the S-

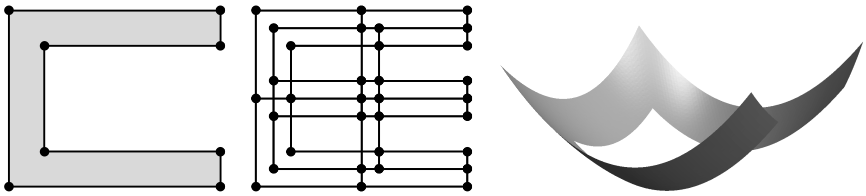

Figure 1.8 An example of an S-Patch with a concave domain. From left to right: the base polygon P shaded gray, the control point structure for the quadratic S-Patch, and an example S-Patch in 3D. Note that, unlike the example in Figure 1.7, the control points do not all lie within the domain P.

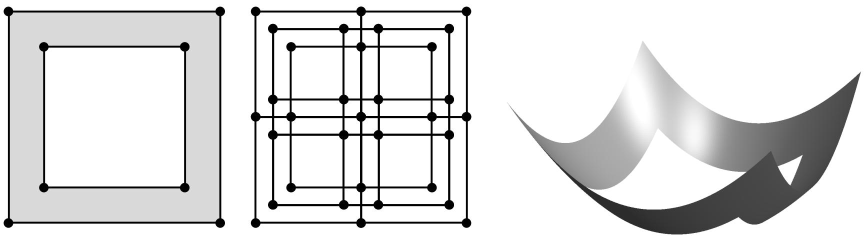

Figure 1.9 An example of an S-Patch whose domain contains a hole. From left to right: the base polygon P shaded gray, the control point structure for the quadratic S-Patch, and an example S-Patch in 3D.

Patch. For barycentric coordinate constructions like mean value coordinates, the domain may even contain holes like the example in Figure 1.9.

1.3.2 Spatial Deformation

S-Patches can be used for applications other than filling multi-sided holes on surfaces as well. Though rarely thought of in this way, S-Patches can be used for image and surface deformation as well. In fact, free-form deformations [9] (FFDs) as well as cage-based deformations [5, 6] are all special cases of deformation using S-Patches.

To perform such spatial deformation, we use S-Patches to construct a map $F : \mathbb{R}^d \to \mathbb{R}^d$ where $d = 2$ for images and $d = 3$ for surfaces and volumes. F is then given by

$$F(\boldsymbol{x}) = \sum_{|\boldsymbol{\ell}|=m} B^m_{\boldsymbol{\ell}}(\boldsymbol{x}) p_{\boldsymbol{\ell}}.$$

Hence, we control the deformation by manipulating the control points $p_{\boldsymbol{\ell}} \in \mathbb{R}^d$. The key to creating a map that is useful for deformation is to construct such a function that can produce the identity transformation; that is, $F(\boldsymbol{x}) = \boldsymbol{x}$ for some

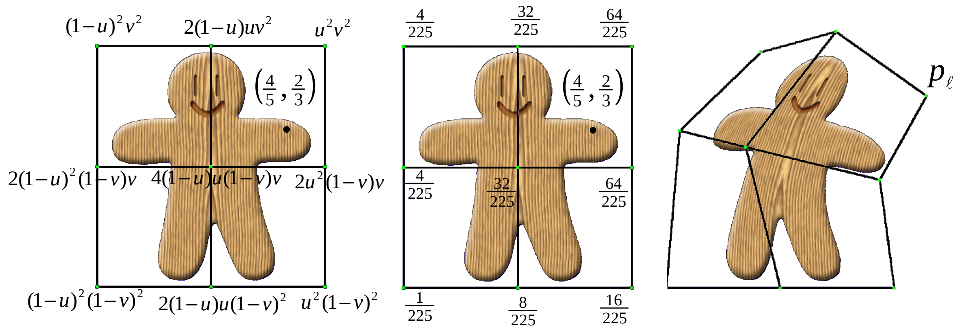

Figure 1.10 Image deformation using S-Patches. The left image shows the image and a point to evaluate the deformation at where each control point lists its basis functions using a quadratic S-Patch with Wachspress coordinates. The middle image shows the weights associated with the evaluation point after evaluating the basis functions. Moving the control points induces a deformation on the image shown on the right.

configuration of control points. Luckily, Equation (1.5) already provides the location of the control points to reproduce this function. We refer to this configuration of control points as the *bind pose*.

Now, given a shape, for each point $\boldsymbol{x}$ in the shape, we compute $\boldsymbol{x}$ as a weighted combination of the control points $p_{\boldsymbol{\ell}}$ where the weights are simply given by the values of the basis functions $B^m_{\boldsymbol{\ell}}(\boldsymbol{x})$. Notice that the weights $B^m_{\boldsymbol{\ell}}(\boldsymbol{x})$ are constant for each point $\boldsymbol{x}$ and can be precomputed, which yields a fast deformation function. When $p_{\boldsymbol{\ell}}$ satisfy Equation (1.5), $F(\boldsymbol{x})$ is the identity map. However, as the user manipulates the control points away from the bind pose, the map produces a smooth deformation of the underlying shape.

Deforming a shape such as an image is simple with this function. Given an image such as the one shown in Figure 1.10, we surround the image with some base polygon P. In this example, we use a rectangle and compute the deformation using a quadratic S-Patch. Then, we sample the deformation using a regular grid over the image. For each grid point, we compute the weights of the deformation function. As the user manipulates the control points, we simply apply the weights to the deformed control point positions to produce a deformed grid. Bilinearly interpolating the deformation within each grid cell generates the final deformation. Notice that the grid in this case may be as fine as the pixels in the input image, although such fine resolution is not typically necessary to create a smooth-looking deformation. Figure 1.10 shows the result of such a deformation. In this case, the deformation is equivalent to using a tensor-product Bézier patch when using Wachspress coordinates, which is called a free-form deformation [9] (a special case of an S-Patch).

Surface deformation follows a similar route. In this case, we are typically given

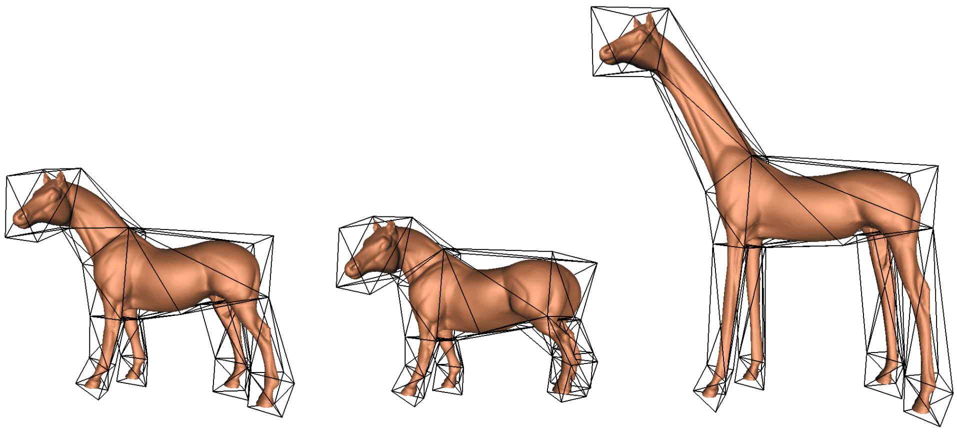

Figure 1.11 An example of surface deformation using linear S-Patches. The initial surface of a horse is surrounded by a low resolution approximation called a cage (left). The right two images show different deformations where the head and torso are kept the same size but the neck and legs are compressed or stretched.

a triangulated surface that we wish to deform with vertices $\boldsymbol{x}_j$. Figure 1.11 (left) shows an example of a 3D model of a horse containing 48485 such vertices. First, we construct a polytope to surround the horse typically with fewer vertices $v_{\boldsymbol{\ell}}$ than the actual horse. In this example, the cage, shown on the left of the figure, contains only 51 control points. Next, we compute each $\boldsymbol{x}_j$ as a weighted combination of the $v_{\boldsymbol{\ell}}$. As the user manipulates the control points, we apply the constant weights for each vertex to the deformed control point locations to create the location of the deformed vertex of the surface as shown on the middle and right of Figure 1.11.

Unlike the original definition of free-form deformations that relies on tensor-product Bézier functions of various degrees with cube domains, cage-based deformations can conform to the shape of the object. And while cage-based deformations are not typically thought of this way, these deformations are obviously S-Patch deformations of degree 1. This insight means that we can construct volumetric cage-based deformations of higher degree as well. Unfortunately, the number of control points for these deformations grows quite rapidly. For example, the cage in Figure 1.11 has 51 control points as a linear S-Patch. A quadratic S-Patch would contain 1326 control points. Moreover, the majority of those control points exist in the interior of the cage, which would make it difficult for a user to manipulate those points. Hence, higher degree cage-based deformations are not practical from a user-interface perspective. One alternative would be to use selective degree elevation [10] to avoid inserting more control points than desired as the degree of the patch is elevated. However, such a method is beyond the scope of this chapter, and we refer the interested reader to the corresponding reference.

Bibliography

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

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

[3] R. Goldman. *Pyramid Algorithms : A Dynamic Programming Approach to Curves and Surfaces for Geometric Modeling*. Morgan Kaufmann, San Francisco (Calif.), 2003.

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

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

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

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

[8] L. Ramshaw. *Blossoming: A Connect-the-dots Approach to Splines*. Digital Systems Research Center, 1987.

[9] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. In *Proceedings of SIGGRAPH*, pages 151–160, 1986.

[10] J. Smith and S. Schaefer. Selective degree elevation for multi-sided Bézier patches. *Computer Graphics Forum*, 34(2):609–615, 2015.

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