The Visual Computer manuscript No.
(will be inserted by the editor)

Partition of Unity Parametrics: A framework for meta-modeling

Adam Runions · Faramarz F. Samavati

the date of receipt and acceptance should be inserted later

Abstract We propose Partition of Unity Parametrics (PUPs), a natural extension of NURBS that maintains affine invariance. PUPs replace the weighted basis functions of NURBS with arbitrary weight-functions (WFs). By choosing appropriate WFs, PUPs yield a comprehensive geometric modeling framework, accounting for a variety of beneficial properties, such as local-support, specified smoothness, arbitrary sharp features and approximating or interpolating curves. Additionally, we consider interactive specification of WFs to fine-tune the character of curves and generate non-trivial effects. This serves as a basis for a system where users model the tools used for modeling, here weight-functions, in tandem with the model itself, which we dub a meta-modeling system. PUP curves and surfaces are considered in detail. Curves illustrate basic concepts that apply directly to surfaces. For surfaces, the advantages of PUPs are more pronounced; permitting non-tensor WFs and direct parameter space manipulations. These features allow us to address two difficult geometric modeling problems (sketching features onto surfaces and converting planar meshes into parametric surfaces) in a conceptually and computationally simple way.

Keywords Meta-modeling, Parametric curves and surfaces, Sketch-based modeling, Geometric modeling

1 Introduction

Parametric curves and surfaces are ubiquitous geometric primitives. These primitives are typically defined using a set of control points. The resulting parametric then takes the form of weighted sums of the control-points, with the contribution of each point determined by an associated weighted basis function. Many properties follow from the choice of weight functions. NURBS inherit many nice properties from B-spline basis functions, and extend B-splines by allowing a scalar weight to be associated with each control point, indicating its *relative importance* to the curve. For these reasons NURBS have emerged as the predominant choice for geometric modeling.

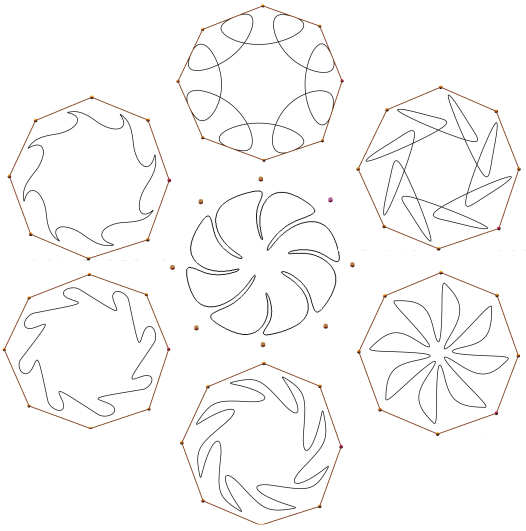

Fig. 1 PUPs allow curves with various characteristics to be generated from simple control polygons.

Despite their widespread use, it is difficult to modify the characteristics of NURBS models. In practice, it is complex to toggle between sharp and smooth features, and interpolation and approximation of control points. Likewise, it is difficult to introduce arbitrarily oriented

A. Runions F.F. Samavati
Department of Computer Science, University of Calgary
2500 University Dr. NW Calgary AB, Canada T2N 1N4
E-mail: runionsa@cpsc.ucalgary.ca

features for surfaces or to control the local character of curves (to produce the loops and other effects depicted in Fig. 1, for example). In part, this is due to NURBS rigid control net structure, which necessitates a rectangular control net for surfaces. Consequently, previous extensions have focused on extending the range of control net topologies [5,21,20]. However, it is not enough to only support a wider range of control net topologies. It is also necessary to control the contribution of control points to the final curve or surface.

Our generalization of NURBS places no topological restrictions on the control net and permits fine-grained control over model characteristics. The key insight underlying Partition of Unity Parametrics (PUPs) is that we can preserve the affine invariance of NURBS while allowing the relative importance of control points to be specified at a finer scale (for each parameter value as opposed to each basis function). To accommodate this, we replace the basis functions of NURBS with arbitrary weight functions. By choosing weight-functions appropriately, it is possible to retain many of the nice properties of NURBS. For this, we rely primarily on linear combinations of B-spline basis functions.

PUPs allow common geometric requirements and operations to be phrased succinctly, including: the addition of control points, arbitrary sharp features, increasing smoothness without increasing support, approximation and interpolation. Additionally, it is possible to fine-tune the character of the curve by editing weight-functions directly. This facilitates a *meta-modeling* framework where weight functions can be designed once with a particular modeling task in mind and then used as a template for subsequent applications. For surfaces, PUPs permit non-tensor weight functions and allow control points to be added anywhere (without introducing other control points). This facilitates simple methods for sketching features and converting a planar mesh into a parametric surface of arbitrary smoothness.

2 Related Work

There have been many works that relax the topological restrictions of NURBS. These include T-Splines, which allow T-junctions in the control mesh [21]; the many n-sided generalizations of B-spline basis functions, which allow for the inclusion of n-sided patches in otherwise regular meshes [20]; and the recent work by Cashman et al. [5], incorporating extraordinary points into NURBS. These extensions allow a wider range of control net topologies, but still require a control net structure. The algebraic-implicit splines, proposed by Li et al. [13], offer an alternative solution, by employing integral convolution to create polyhedral B-spline-like basis functions. However, they do not consider parametric curves and surfaces. Although these works allow a wider range of control-net topologies to be realized, they do not consider how fine-grained control over the character of models can be achieved.

The association of a weight function with each point, a key element of our approach, is also central to interpolation techniques relying on radial basis functions [6,4]. For these interpolation techniques a radial basis function localizing the contribution of a point to the surface is associated with each point. Based on the radial basis functions a system of equations must then be solved [4,23] to obtain a function [6,4] or implicit surface [23] that interpolates the sample points.

Our generalization of NURBS exploits the idea that given a set of control points and associated weight-functions, we can guarantee affine invariance by normalizing the weight-functions to provide partition-of-unity. This insight was previously explored by Shepard [22] to generate interpolating functions from scattered data and underlies recent generalizations of barycentric coordinates [9,14]. Similar ideas were proposed for parametric surfaces by Sederburg [21] in the form of point-based splines and Wang et al. [24] in the form of G-NURBS. However, in these works, the variation of weighted basis functions as a framework for geometric modeling has not been explored or used for meta-modeling (e.g. to control the general characteristics of curves and surfaces). Additionally, restrictions on the forms of weight-funtions in these works (e.g. monotonically decreasing, non-negative functions) limits the range of possible parametrics (precluding, for example, the curves shown in Fig. 1 and 4).

Franke and Nielson [10] modified Shepard's method by computing algebraic approximations of the surface at each point, which were blended using weight functions and normalized to guarantee affine invariance. The basic idea developed by Franke and Nielson has been used extensively, as a basis for constructing implicit fields [16], and manifold surfaces [11,25]. In contrast, we employ Shepard's basic idea, with appropriately chosen weight-functions, towards modeling with parametric surfaces. We do not rely on local algebraic surface approximations, which simplifies our method and permits a flexible meta-modeling environment.

3 Moving from NURBS to PUP Curves

Before introducing our proposed generalization, let us first consider the standard definition of a NURBS curve. A NURBS curve is defined by a set of control-points $P = \{P_0, P_1, \cdots, P_n\}$, a set of knot values (required for the B-spline basis functions used below)

$U = \{u_0, u_1, \cdots, u_{n+k}\}$, and a set of weights indicating the relative importance of each control point to the curve $W = \{w_0, w_1, \cdots, w_n\} \subset \mathbb{R}^+$. The NURBS curve of order k, is then defined over the interval $[u_{k-1}, u_{n+1}]$ and has the form

$$Q(u) = \sum_{i=0}^{n} R_i(u) P_i \tag{1}$$

where the basis functions $R_i(u)$ are defined as

$$R_i(u) = \frac{w_i N_i^k(u)}{\sum_{j=0}^{n} w_j N_j^k(u)}, \tag{2}$$

where $N_i^k(u)$ is a B-spline basis function of order k. It is important to note the sum in the denominator of Eq. 2, which *normalizes* the weighted basis function. This guarantees that the basis functions $R_i(u)$ always sum to one, which makes the curve affine invariant [19].

We generalize NURBS by replacing the weighted basis-functions in Eq. 2 with arbitrary *Weight Functions* (WF) and preserve the normalization that provides affine invariance. Consequently, it is possible to explicitly specify weights for every parameter value. A PUP curve $Q(u)$ is defined on the interval $[a, b]$, and specified by a set of control points $P = \{P_0, P_1, \cdots, P_n\}$, with associated weight-functions $W = \{W_0(u), W_1(u), \cdots, W_n(u)\}$, where each $W_i : [a, b] \to \mathbb{R}$, is a scalar function. The general form of Eq. 1 still applies in this setting. However, the definition of $R_i(u)$ (Eq. 2) becomes

$$R_i(u) = \frac{W_i(u)}{\sum_{j=0}^{n} W_j(u)}, \tag{3}$$

guaranteeing that the $R_i(u)$s always sum to one. We refer to the $R_i(u)$s as *normalized weight-functions*. Finally, to avoid indeterminate forms in Eq. 3, we require

$$\sum_{i=0}^{n} W_i(u) \neq 0, u \in [a, b]. \tag{4}$$

We note that by setting $W_i(u) = w_i N_i^k(u)$ (and $[a, b]$ to $[u_{k-1}, u_{n+1}]$), we obtain the NURBS curve defined by Eq.1-2. Hence, NURBS curves are a special case of PUP curves.

For curves, we assume that each W_i has local support on the interval *centered* on i with radius c_i, making

$$W_i(u) = 0, u \notin [i - c_i, i + c_i] \tag{5}$$

In general, any W_i with local support can be transformed into a function with the properties listed above. These characteristics permit the curve to be evaluated efficiently and facilitate interpolation. $Q(u)$ is then defined on the interval $[0, n]$ for open curves, and $[0, n+1]$ for closed curves (extending the parameter range to connect the first and last points of the curve).

4 Determining appropriate weight functions

The definition of a PUP curve is very general. In fact, any curve that is representable as an affine, weighted-combination of points can be formulated as a PUP curve. Consequently, it is important to establish some additional constraints that can be used in selecting the WFs when defining a curve. PUPs are affine-invariant by construction. However, to offer a comprehensive geometric modeling tool, it is necessary to support curves of arbitrary smoothness (C^k continuity), local-support, the convex-hull property, the strong convex-hull property, approximation, interpolation and sharp features (although not simultaneously). To this end, we consider what geometric properties our curves should have and select our WFs appropriately. For example, when all W_i are C^k the resulting curve is also C^k, and when the W_i are non-negative the convex hull property is satisfied.

For weight functions we employ uniform B-spline basis functions as basic WFs and use one-dimensional B-spline functions to specify more complex WFs. Thus each W_i has the form

$$W_i(u) = s_i B_i^k((u - i)/c_i), \tag{6}$$

where B_i^k is a k^{th} order B-spline basis function or one-dimensional B-spline function with support on the interval $[-1, 1]$, with $s_i, c_i \in \mathbb{R}$. Consequently, each WF has three default parameters: the radius of support c_i, a uniform scaling factor s_i, and its order k. To evaluate PUP curves, we have implemented an interactive editor. The editor supports typical interactions, such as the addition and manipulation of control points. More importantly, it supports operations exceeding what is currently feasible for NURBS. In particular, WFs can be specified interactively: either locally (at a single control point) or globally (for all control points simultaneously). This allows users to fine-tune the character of the curve and serves as the basis for generating sharp features (Section 4.1) as well as converting between approximating and interpolating curves (Section 4.2).

4.1 Examining PUP curves

The PUPs definition is very general, which may make it unclear how this generality can be leveraged in a practical setting. Here, we illustrate the flexibility of our definition by examining the range of curves it can generate.

To start, consider a PUP curve with uniform B-spline basis functions for WFs. Superficially, it may appear that a B-spline curve is produced. However, PUP curves permit an additional degree of freedom compared to NURBS curves, as the radius of support (c_i)

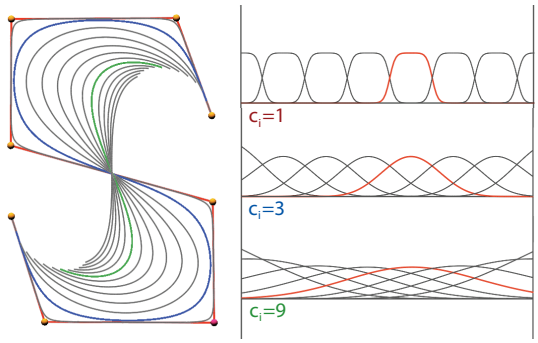

Fig. 2 PUP curves created using 4th degree uniform B-spline basis functions as WFs. The control polygon is fixed while the support (c_i) is varied from 1 (which interpolates the control polygon) to 14. Normalized weight-functions for three curves with different intervals of support are shown (right).

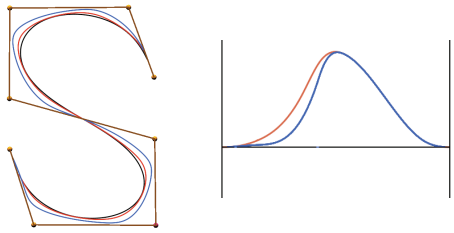

Fig. 3 The control-polygon from Fig. 2 with asymmetric WFs (with $c_i = 3$). The black curve has no bias and serves as a reference. The curves (left) were generated using the WF (right) of the same color.

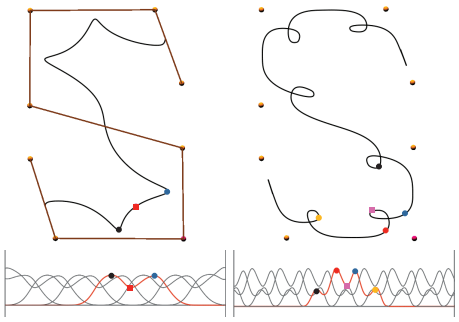

Fig. 4 Two PUP curves, created using one dimensional B-spline WFs. The corresponding normalized WFs are shown below each curve. Pulls and pushes are illustrated by the colored circles and squares respectively.

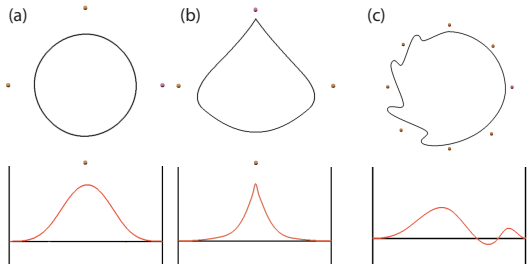

Fig. 5 Examples of local WF specification. The introduction of a sharp feature into an otherwise smooth curve (a-b). (a) The curve is created using the smooth WF below. By introducing a sharp feature into the WF of the top-most control point, a corresponding sharp feature is created in the curve (b). (c) A PUP curve with different WFs for the control points on the left and right portions of the curve. The WF used to generate the left portion of the curve is shown below.

can still be varied. Examining the progression in Fig. 2, we see that decreasing the support of the WF causes the curve to pull towards the control points. Simultaneously, curvature becomes focused in the portions of the curve closest to the control points.

This behavior corresponds to the shape parameter *tension* as described by Barsky for Beta-splines [1] (the smoothing parameter in [13] has a similar effect). Barsky describes a second shape parameter, *bias*, as an asymmetry in the pull of the curve towards the control polygon before and after a control point. For tension, as the support of the WF decreases, the curvature of the WF is focused in a decreasing parameter range, causing the curvature of the curve to change similarly. Bias can be introduced by creating an asymmetry in the WF about its centre as illustrated in Fig. 3. Increasing the asymmetry in the WF likewise increases the bias exhibited by the curve (compare the red and blue curves).

Using arbitrary B-spline functions as WFs allows us to move beyond the simple shape parameters described above (Fig. 4). Here, the relation between WFs and the resulting curve is more complex, but can be characterized by the number of times a point's WF causes the curve to *pull* towards a control point and conversely the number of times it is *pushed* away . The number of pulls toward the control point P_i is equal to the number of local maxima in the corresponding normalized WF R_i, whereas the number of pushes is equal to the number of local-minima in R_i. This is illustrated in Fig. 4 using two examples. The WF in the first example has two local maxima separated by a local minima. This results in a curve that first pulls towards each control point, then pushes away from the point as it passes, before pulling back towards the control point a final time. In the second example, a loop is introduced near each control point. Each loop results from the interplay of neighboring normalized weight functions. As the curve passes the control point it is first pulled towards the next control point, then the preceding control point, and finally towards the next control point again. In practice, loops can be generated by interleaving the pulls of consecutive control points.

All the examples presented so far were obtained by globally specifying WFs. Locally specifying WFs can be used to create localized effects, such as sharp features (Fig. 5 (b)). Unlike NURBS, the curve need not pass through the control point. In Fig. 6(left), different WFs have been used to give different portions of the curve

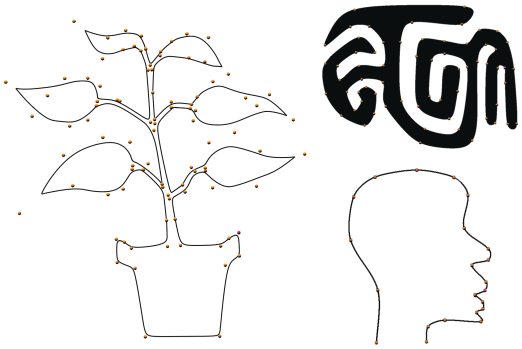

Fig. 6 Several examples of more complex PUP curves. Left: approximating WFs. Right: interpolating WFs.

different characters. This is used to specify smooth leaf margins, terminating sharply at the tip and to imbue the branching points and pot with different characters. It is also possible to piece together two PUP curves, while maintaining sufficient smoothness, by connecting the parameter ranges of two curves and allowing the WFs of each curve to extend to the neighboring curve (as illustrated in Fig. 5 (c)).

4.2 Interpolation

As smooth NURBS curves are approximating, interpolation is accomplished indirectly, by solving a system of equations to determine the control points required for the curve to pass through a given set of interpolants [19,2]. Using PUPs, however, interpolation can be addressed directly, by selecting an interpolation site for each control point and constructing appropriate WFs. As the WF associated with P_i is centered at i in the parameter domain, a natural choice is $Q(i) = P_i$. Interpolation of P_i at i is obtained when *only* the WF associated with P_i is active at i (this does not constrain W_i *between* integer values). This is equivalent to the following condition

$$W_i(j) = b_{ij}\delta_{ij} \tag{7}$$

where $j \in \{0, 1, \cdots, n\}$ and $b_{ij} \in \mathbb{R}$ is an arbitrary non-zero constant. To satisfy Eq. 7, we use shifted copies of the normalized sinc function (although other functions, like cubic splines, can also be used):

$$g_i(u) = sinc(u - i) = \frac{sin(\pi(u - i))}{\pi(u - i)}, \tag{8}$$

which is $C^{\inf}$ with $g_i(j) = \delta_{ij}, (j \in \mathbb{N})$. Using g_i, we can satisfy the condition given in Eq. 7 by replacing W_i with $g_i(u)W_i(u)$, yielding

$$g_i(i)W_i(i) = \delta_{ij}W_i(i) = \delta_{ij}b_{ij}.$$

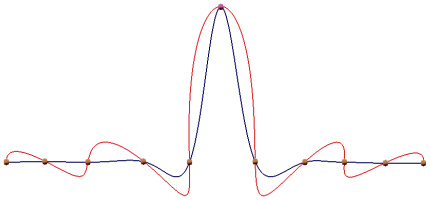

Fig. 7 Comparison of sinc interpolation (red) and sinc interpolation regulated by a WF with local support (blue).

This suffices to convert a given PUP curve into an interpolating curve. Of course, setting $W_i = g_i$ also satisfies Eq. 7, but the resulting curves suffer from the typical problems afflicting interpolating curves (i.e. ringing and over-shooting of control points, see Fig. 7(red curve)). Multiplying g_i by W_i, a WF with local support and C^k continuity, generates interpolating curves that respond predictably and minimize the problems arising when g_i is used directly (Fig. 7(blue curve)). Two examples using the method outlined above and B-spline basis functions of degree 3 with radius of support $c_i = 3$ are provided in Fig. 6.

4.3 Efficient evaluation of PUP curves

Interactive editing of PUP curves necessitates an efficient evaluation method. Our method (illustrated in Fig. 8) exploits the local support of WFs and a uniform spacing of evaluation points. The method proceeds in two phases, by: first calculating the un-normalized contribution of each control point to the curve, then normalizing these values to guarantee affine invariance.

To evaluate $Q(u)$ at uniformly spaced parameter values $\{u_0, u_1, \cdots, u_s\}$, we first allocate an array of evaluation points $E_p : \{0, 1, \cdots, s\} \to \mathbb{R}^n$ and an array to store the sum of weight values $E_w : \{0, 1, \cdots, s\} \to \mathbb{R}$, at each u_j (Fig. 8 (a)).

During the first phase each control point P_i is considered, and $W_i(u_j)P_i$ is added to each sample within it's interval of support (Fig. 8(b)-(e)). At the same time, $W_i(u_j)$ is added to $E_w(j)$. Once every point has been considered, we obtain $E_p(j) = \sum_{i=0}^{n} W_i(u_j)P_j$, the weighted sum of points at each u_j and $E_w(j) = \sum_{i=0}^{n} W_i(u_j)$, the sum of all the WFs for each u_j.In the second phase, the position of each sample point is then obtained by dividing $E_p(j)$ by $E_w(j)$ (Fig. 8(f)).

The method's efficiency depends on the support of each W_i and the complexity of evaluating W_i on this interval. Storing the sum of weight-values at each sample point permits the curve to be updated efficiently when local edits are performed (i.e. a control point's position

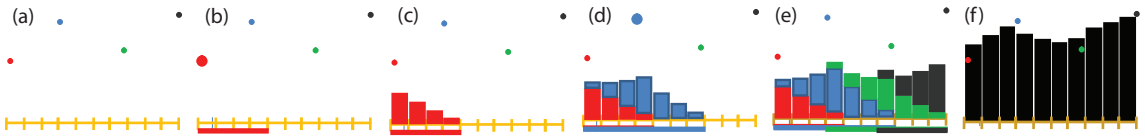

Fig. 8 An example of the evaluation method described in the text. The curve consists of four points (colored circles). The yellow intervals at the bottom of each image denote elements of E_p(E_w is not shown). E_p is first initialized to zero (a). The red-point is considered first (b), this point's support is visualized as the red-bar under E_p. The contribution of the red point is added to E_p (c). This process is repeated for the remaining three points of the curve (d-e). Finally, the value of E_p is normalized by dividing each entry by the corresponding entry of E_w ((f), black bars).

or WF is modified). For this we use a modified version of the method proposed by Barsky [1](chapter 8).

5 PUP Surfaces

PUP surfaces have a very similar definition to that provided for curves. The surface is still defined by P, a set of control points, and W, a corresponding set of WFs. However, W_i now takes the form $W_i(u,v) : D \to \mathbb{R}$ where $D \subset \mathbb{R}^2$. Thus normalized WFs are now

$$R_i(u,v) = \frac{W_i(u,v)}{\sum_{j=0}^{n} W_j(u,v)}, \tag{9}$$

making the surface

$$Q(u,v) = \sum_{i=0}^{n} R_i(u,v)P_i,\ (u,v) \in D. \tag{10}$$

Additionally, we still require

$$\sum_{i=0}^{n} W_i(u,v) \neq 0,\ (u,v) \in D \tag{11}$$

to avoid indeterminate forms.

Thus, the surface is simply computed as a sum of weighted points over a given domain D. As no additional structure is needed, this frees us from the constraints of knot-values and the rigid control-net structure required for NURBS. This permits a wider-range of WFs and more natural parameter space manipulation. This flexibility is illustrated by addressing the problem of sketching details onto surfaces and converting planar meshes into PUP surfaces. Extending PUPs to volumes and higher dimensions follows similarly.

As for curves, we consider WFs of a specific form. Each W_i is centered at (u_i, v_i) in D with radii of support (c_i^u, c_i^v) in the u and v directions, yielding

$$W_i(U) = 0, U \notin [u_i - c_i^u, u_i + c_i^u] \times [v_i - c_i^v, v_i + c_i^v]. \tag{12}$$

We also restrict the parameter domain to the rectangle $[a,b] \times [c,d]$. Using a rectangular parameter domain and requiring local support allows the surface to be evaluated efficiently by extending the method in Section 4.3 to 2D domains. When interpolation is required, we arrange the WF centers in a uniform grid, permitting the scheme for curves to be extended to surfaces.

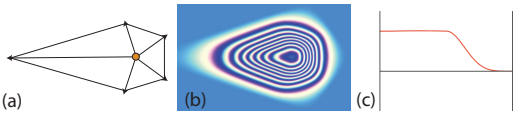

Fig. 9 An axial WF is constructed using a set of axis (a, black arrows) and falls-off from a maximum at (u_i, v_i) (the orange circle) to zero at the edges of the polygon. The falloff along an axis is determined by a function with the form shown in (c). The resulting WF for the axes and function shown in (a) and (c) is depicted in (b).

5.1 Surface Weight Functions

In contrast to NURBS, which use tensor product WFs exclusively, we consider three additional WFs: rotated-tensor, radial, and axial; defined using the one-dimensional WFs used for curves. Each has several parameters, including the radius of support in the u and v directions (c_i^u and c_i^v) and a uniform scaling factor s_i.

Tensor product WFs have the usual form, resulting from the multiplication of two one-dimensional weight-functions. However, the PUP surface definition allows for tensor product WFs that are not aligned with the u,v directions. We exploit this by using rotated-tensor WFs, created by rotating tensor WFs by θ in the parameter domain to obtain

$$W_i(u,v) = W(u_\theta, v_\theta),$$

where W is a tensor WF centered at (u_i, v_i) and (u_θ, v_θ) is obtained by rotating (u,v) by θ about (u_i, v_i) in the parameter domain. Thus $(u_\theta, v_\theta) = R_{-\theta}(u,v)$ where $R_{-\theta}$ is a two-dimensional rotation by $-\theta$. It is important to note that to rotate the *value* of W by θ in the parameter domain, we must rotate (u,v) by $-\theta$.

It is also possible to consider radial WFs (i.e. radial basis functions [4]) defined as

$$W_i(u,v) = W(d),\ (d = \|(u,v) - (u_i, v_i)\|),$$

where W is a uniform B-spline WF. Given that W decreases monotonically as d increases, the radial weight-functions fall off continuously as the distance between (u,v) and (u_i, v_i) increases.

Axial weight functions are defined by a set of axes $A = \{\alpha_0, \alpha_1, \cdots, \alpha_p\} \subset \mathbb{R}^2$ in the parameter domain, which demarcate the boundaries of a polygon (Fig. 9(a)). We then construct W_i to fall-off smoothly along each axis, from a maximum value at (u_i, v_i) to zero on the polygon's boundary. For this, we use WF of the form

$$W_i(u,v) = \prod_{r=0}^{p} W_{\alpha_r}(proj_{\alpha_r}(u,v)),$$

where α_r is an axis, $proj_{\alpha_r}(u,v)$ is the projection of $(u-u_i, v-v_i)$ on α_r and each W_{α_r} is a B-spline curve of sufficient smoothness, decreasing from one to zero as the magnitude of the projection increases (Fig. 9(c)). This generates WF of the form shown in Fig. 9(b). We note that the weight-function will have compact support provided that the axes A delimits a valid polygon.

5.2 Examining PUP Surfaces

Much of the exploration performed for curves in Section 4.1 applies in the context of surfaces, including: the relation between support and tension, and asymmetries in the WF and bias. However, there are two avenues of exploration that cannot be inferred directly from curves.

First, we will illustrate the impact of the WFs outlined in Section 5.1 using a simple surface (Fig. 10(a-e)). From the figure, we can see that each WF can be used to define a smooth surface that has a predictable correspondence to the control points defining the surface. The bottom row illustrates the parametric structure of each example. Starting from a tensor-product WF (a) (or radial-WF (d)), we can extend the support in one parametric direction to obtain anisotropic WFs (b). Rotating the WF (c) causes the features generated by displaced control points to change orientation similarly. For the axial WF used in (e), the impact of the WF falls off as depicted in Fig. 9, evident in the asymmetry of the features of the surface. As with curves, we can obtain controllable interpolation for surfaces by modulating the normalized sinc functions with one of the preceding weight-functions (f).

Extending PUP to surfaces provides flexibility in the specification of WFs. For example, the rotated tensor WF permits WFs that are not aligned with the u,v directions of the parameter space. The advantages of this is especially apparent in the scenario depicted in Fig. 11, where a user has introduced a feature into the surface (a diagonal line) that does not follow the control net. For NURBS, this can create undesirable effects (a) that can only be addressed by modifying the position or number of control points. However, by rotating our WFs, it is possible to follow the feature without modifying the position or number of control points (b).

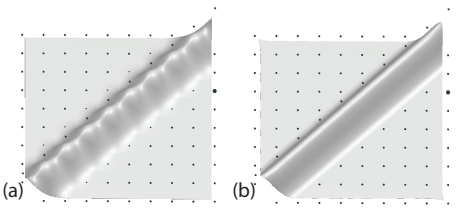

Fig. 11 A PUP surface created from the initial control grid by elevating control points in a diagonal line. When WFs are aligned with the control net, the feature is not respected by the surface (a). Using rotated tensor WFs allows the surface to represent the feature more precisely (b).

6 Converting planar meshes to PUP surfaces

To convert a given polygon mesh M with planar topology (i.e. the mesh has a single boundary), into a PUP surface, we proceed by computing a global parameterization and then construct a WF for each vertex of M. To parameterize the mesh we use the method proposed by Floater in [8] (although, any method that produces parameterizations with a locally uniform distribution of points should work equally well).

Given a mesh with planar topology M, with vertices $V = \{P_0, P_1, \cdots, P_n\}$ and corresponding parameter values $U = \{U_0, U_1, \cdots, U_n\} \subset \mathbb{R}^2$, we construct an axial-WF for each vertex. Thus, the vertices of M serve as the control points of our surface. Accordingly, for each P_i, to construct W_i, we need to specify the center, support, and axes defining the WF. The center is given by the corresponding parameter value $U_i = (u_i, v_i)$. The set of axes A are then defined as

$$\alpha_k = U_k - U_i,\ k \in N(i),$$

where $N(i)$ are indices of P_i's neighbors in M. Finally, the support for W_i is $c_i = (c_u du_i, c_v dv_i)$ where $c_u, c_v \geq 1$ are constants and

$$du_i = \max_{k \in N(i)}\{|u_k - u_i|\},$$
$$dv_i = \max_{k \in N(i)}\{|v_k - v_i|\},$$

the maximum distance between U_i and neighboring parameter values in each parametric direction. Choosing our support in this way guarantees that Eq. 11 is satisfied. This suffices to convert M into a PUP surface. The method is illustrated in Fig. 12 using a mannequin head mesh. Once the mesh has been converted into a PUP surface, it can be edited by moving and adding control points (c-d). More complex topologies can be handled by decomposing the mesh into planar patches (as in [7]), but are not addressed here.

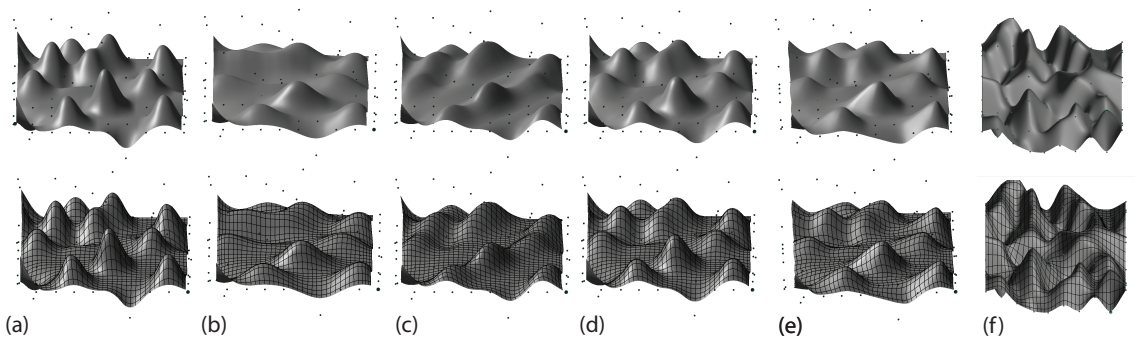

Fig. 10 PUP surfaces generated using the same control points while the WF is varied; rendered as a shaded surface (top), and with isocurves (bottom). The surfaces were generated using the following WFs: (a) tensor, (b) elongated tensor, $c_i^u > c_i^v$, (c) rotated-tensor ($\theta = 45$), (d) radial, (e) axial (the WF from Fig. 9). (f) Interpolation is demonstrated, for a different set of control points.

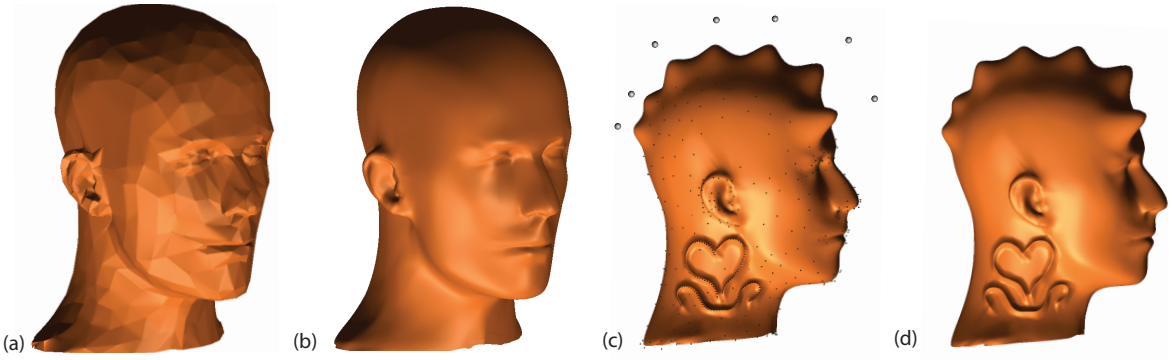

Fig. 12 A coarse mesh with 636 vertices (a) and corresponding PUP surface (b). In (c), several edits have been performed on the mesh. A horn and bumpy ridge along the middle of the head have been introduced by manipulating control points (silver spheres). Additionally, the heart motif has been introduced by adding control points and WF to the surface (using the sketching method outlined in Section 7). In (d) the mesh is rendered without control points.

7 Sketching Details on Surfaces

Sketching details onto a base surface is commonly used to refine a pre-existing model [18]. This problem has been addressed previously [17,15]. However, modifying the geometry of the surface based on a user's sketch usually involves solving a fairly complex optimization problem [15]. Furthermore, creating visually pleasing sharp features requires the addition of edges and vertices to the mesh [15,18] (or control-net for NURBS surfaces) that are aligned with the sharp feature. In contrast, PUPs permit a particularly simple method for sketching details onto a base surface.

Starting with a base surface, the user sketches a curve above the surface (Fig. 13 (a)). The curve is arc-length parameterized and resampled to provide the set $C = \{P_0, P_1, \cdots, P_r\}$ of control points that will be added to the surface (Fig. 13 (b)). A WF, oriented along the curve, is then calculated for each control-point (Fig. 13 (c)), and added to the surface (Fig. 13(d)). Constructing WFs to support high quality features and

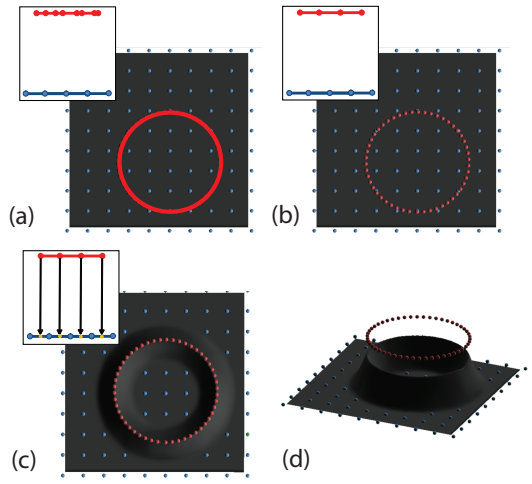

Fig. 13 Overview of the procedure for sketching details onto a base-surface, insets depict an orthogonal viewpoint. (a) First, the user sketches the curve (shown in red). (b) The curve is then resampled. (c) The resulting points are projected onto the surface (inset), and used to calculate corresponding WFs. The points and WFs are then added to the surface. (d) The resulting surface is shown from another vantage point.

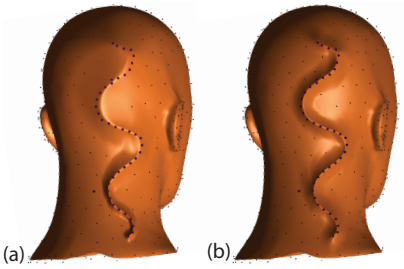

Fig. 14 A single stroke is placed on the surface, (a)without and (b)with compensation for metric distortion.

further editing requires special consideration. Our construction is outlined below.

We create a WF W_t^c for each point $P_t \in C$ according to the definition given in Section 5.1, by specifying three attributes: the centre and radius of support ($U_t = (u_t, v_t)$ and $C_t = (c_t^u, c_t^v)$ resp.), and the form of W_t. The centers of support are determined by projecting each P_t onto the surface and estimating the corresponding parameter value (Fig. 13 (c)). This produces a second curve C_D, corresponding to C, in the parameter domain, with $C_D = \{U_0, U_1, \cdots, U_r\}$, where U_t is associated with $P_t \in C$.

Next, we determine an appropriate W_t^c for each P_t. Here, we employ rotated-tensors, aligned with C_D at U_t using the tangent of the curve T_t. Finally, we must specify the radius of support $C_t = (c_t^u, c_t^v)$, measured along and perpendicular to T_t respectively. The mapping from D to $\mathbb{R}^3$ by $Q(U)$ typically distorts the angles, distances, and areas of D creating *metric distortion* [12]. As such, using the same support for all W_t^c causes the stroke width to vary along the surface (Fig. 14(a)). To compensate, we let T_t' be perpendicular to T_t and use support of the form

$$C_t = \left(\frac{c^u}{D_t}, \frac{c^v}{D_t'} \right),$$

where $c^u, c^v \in \mathbb{R}$ are constants and

$$D_t = \left|\left| \tfrac{\delta Q(U)}{\delta U} \cdot T_t \right|\right|, \; D_t' = \left|\left| \tfrac{\delta Q(U)}{\delta U} \cdot T_t' \right|\right|$$

are the magnitudes of the directional derivative of $Q(U)$ along and perpendicular to T_t respectively. This compensates for metric distortion and can dramatically change the resulting feature (Fig. 14(b)). Once the WFs have been constructed, the control points C and their corresponding WFs are added to the surface.

The proposed method permits interactive frame rates, even when several curves are added to the surface (Fig. 12c-d and 15). Additionally, details can be accurately represented using only a small number of control points (even for complex features). The character of the details introduced can be fine-tuned by modifying the

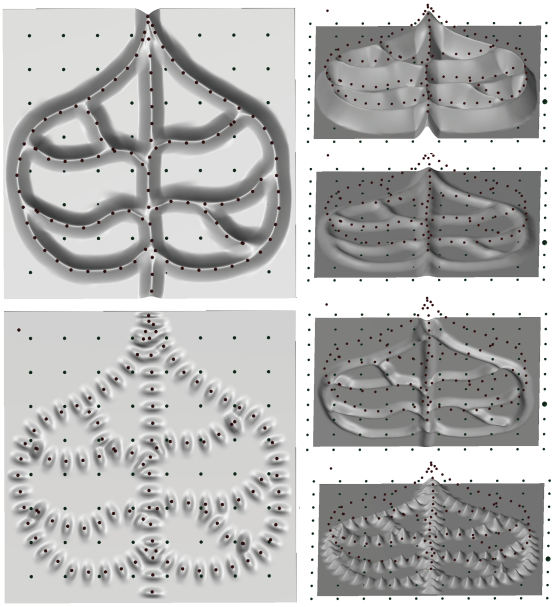

Fig. 15 A complex feature, consisting of nine strokes, has been added to the surface (left). Changing the WFs associated with the curve changes the character of the feature as described in the text (right).

WFs associated with the curve. In Fig. 15(right), this possibility is explored to obtain sharp-extrusions (top), soft-extrusions (middle-top), a soft-indentation into the surface (middle-bottom, by specifying a negative uniform scaling for the WFs), and to introduce regularly spaced features along each curve (bottom).

8 Conclusions

Our proposed generalization, PUPs, is a natural extension of NURBS, obtained by preserving affine invariance. PUPs continue the progression of ideas from Bezier curves to B-splines and NURBS. B-splines allow the degree and number of control points in a curve to be specified independently. Whereas, PUPs allows the degree and support of basis functions to be specified independently. NURBS allow the relative importance of each control point to be specified, whereas, PUPs allow the relative importance of control points to be specified arbitrarily along the curve.

Our exploration of partition of unity parametrics illustrates the utility of specifying WFs, either to fine-tune the character of a parametric or obtain interesting effects. Once a WF, or class of WFs, has been identified, the modeler can save them for use in other applications. This provides the foundations for a meta-modeling environment, where the modeler can design the tools that are used as a basis for geometric modeling.

For curves, we developed alternative definitions of the shape parameters bias and tension. Additionally, the complementary shape parameters of pushes and pulls were described and related to the WFs defining the curve. Furthermore, we illustrated that high-quality interpolants can be produced without solving a least-squares problem (as is required for NURBS). It is also possible to specify complicated curves using a coarse control polygon where the local characteristics, or details, of the curve are captured by the weight-functions (Fig. 1). This hints at a more general idea, namely using weight-functions to encode local details, such as wrinkles or artistic styles (c.f. [3]).

Conceptually, the ideas developed for curves extend directly to surfaces. However, for surfaces, the removal of NURBS rigid control-net structure provides a greater degree of freedom in specifying surfaces. This permits remarkably simple methods for sketching details onto a base surface (comparable in quality to the current state of the art [15,17]) and approximating planar meshes using PUP surfaces. Together, these illustrate the applicability of PUP curves and surfaces to difficult geometric modeling problems.

There are many ideas that we feel merit further exploration. Foremost, facilitating intuitive interaction requires an appropriate means for interactively specifying weight-functions. In this paper, we only considered interactive specification of the weight-functions directly, which sometimes only loosely correspond to their normalized form. Additionally, as NURBS have been applied to many standard problems arising in geometric modeling we feel it would be elucidating to re-examine these problems using PUPs. One particularly appealing problem is using PUPs for curve and surface fitting, by employing linear least squares or non-linear optimization. Conversely, it is important to establish techniques and conditions to convert or approximate PUP surfaces using NURBS. This would make PUPs more applicable to problems arising in CAD/CAM applications where NURBS are the surface of choice.

Acknowledgements We thank Przemyslaw Prusinkiewicz for insightful discussions and editorial comments. This research was supported in part by the National Science and Engineering Research Council of Canada and GRAND Network of Centre of Excellence of Canada.

References

1. Barsky, B.: Computer Graphics and Geometric Modeling using Beta-splines. Springer-Verlag (1988)
2. de Boor, C.: A practical guide to splines (revised edition). Springer-Verlag (1978)
3. Brunn, M., Sousa, M., Samavati, F.: Capturing and reusing artistic styles with multiresolution analysis. International Journal of Images and Graphics **7**(4), 593–615 (2007)
4. Buhmann, M.D.: Radial basis functions. Acta Numerica **9**, 1–38 (2000)
5. Cashman, T.J., Augsdörfer, U.H., Dodgson, N.A., Sabin, M.A.: NURBS with extraordinary points: high-degree, non-uniform, rational subdivision schemes. In: SIGGRAPH '09, pp. 1–9 (2009)
6. Duchon, J.: Splines minimizing rotation-invariant seminorms in sobolev spaces. In: W. Schempp, K. Zeller (eds.) Constructive Theory of Functions of Several Variables, vol. 571, pp. 85–100 (1977)
7. Eck, M., Hoppe, H.: Automatic reconstruction of b-spline surfaces of arbitrary topological type. In: proceedings of SIGGRAPH '96, pp. 325–334 (1996)
8. Floater, M.S.: Parametrization and smooth approximation of surface triangulations. Comput. Aided Geom. Des. **14**(3), 231–250 (1997)
9. Floater, M.S.: Mean value coordinates. Comput. Aided Geom. Des. **20**(1), 19–27 (2003)
10. Franke, R., Nielson, G.: Smooth interpolation of large sets of scattered data. International Journal for Numerical Methods in Engineering **15**(11), 1691–1704 (1980)
11. Grimm, C.M., Hughes, J.F.: Modeling surfaces of arbitrary topology using manifolds. In: Proceedings of SIGGRAPH '95, pp. 359–368 (1995)
12. Hormann, K., Lévy, B., Sheffer, A.: Mesh parameterization: theory and practice. In: SIGGRAPH 2007 course notes, pp. 1–87 (2007)
13. Li, Q., Tian, J.: 2d piecewise algebraic splines for implicit modeling. ACM Trans. Graph. **28**(2), 1–19 (2009)
14. Meyer, M., Barr, A., Lee, H., Desbrun, M.: Generalized barycentric coordinates on irregular polygons. J. Graph. Tools **7**(1), 13–22 (2002)
15. Nealen, A., Igarashi, T., Sorkine, O., Alexa, M.: Fibermesh: designing freeform surfaces with 3D curves. In: SIGGRAPH '07, p. 41 (2007)
16. Ohtake, Y., Belyaev, A., Alexa, M., Turk, G., Seidel, H.P.: Multi-level partition of unity implicits. ACM Trans. Graph. **22**(3), 463–470 (2003)
17. Olsen, L., Samavati, F., Sousa, M., Jorge, J.: Sketch-based mesh augmentation. In: Proceedings of the sketch-base interfaces and modeling (SBIM) workshop (2005)
18. Olsen, L., Samavati, F., Sousa, M., Jorge, J.: Sketch-based modeling: A survey. Computers and Graphics **33**(1), 85 – 103 (2009)
19. Piegl, L., Tiller, W.: The NURBS Book. Springer-Verlag (1997)
20. Piegl, L., Tiller, W.: Filling n-sided regions with NURBS patches. The Visual Computer **15**(2), 77–89 (1999)
21. Sederberg, T.W., Zheng, J., Bakenov, A., Nasri, A.: T-splines and T-NURCCs. ACM Trans. Graph. **22**(3), 477–484 (2003)
22. Shepard, D.: A two-dimensional interpolation function for irregularly-spaced data. In: Proceedings of the 23rd ACM national conference, pp. 517–524 (1968)
23. Turk, G., O' Brien, J.: Variational implicit surfaces. Tech. rep., Georgia Institute of Technology (1999)
24. Wang, Q., Hua, W., Guiqing, L., Bao, H.: Generalized NURBS curves and surfaces. In: Geometric Modeling and Processing, pp. 365–368 (2004)
25. Ying, L., Zorin, D.: A simple manifold-based construction of surfaces of arbitrary smoothness. In: SIGGRAPH '04, pp. 271–275 (2004)