

Number 699



**UNIVERSITY OF
CAMBRIDGE**

Computer Laboratory

A smooth manifold based construction of approximating lofted surfaces

Richard Southern, Neil A. Dodgson

October 2007

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500
<http://www.cl.cam.ac.uk/>

© 2007 Richard Southern, Neil A. Dodgson

Technical reports published by the University of Cambridge
Computer Laboratory are freely available via the Internet:

<http://www.cl.cam.ac.uk/techreports/>

ISSN 1476-2986

A smooth manifold based construction of approximating lofted surfaces

Richard Southern and Neil A. Dodgson

Abstract

We present a new method for constructing a smooth (theoretically C^∞) manifold approximating a curve network or control mesh. In our two-step method, smooth vertex patches are initially defined by extrapolating and then blending a univariate or bivariate surface representation. Each face is then constructed by blending together the segments of each vertex patch corresponding to the face corners. By approximating the input curve network, rather than strictly interpolating it, we have greater flexibility in controlling surface behaviour and have local control. Additionally no initial control mesh fitting or fairing needs to be performed, and no derivative information is needed to ensure continuity at patch boundaries.

1 Introduction

Curves are extensively used in CAD packages to specify surface behaviour. Coons patches [Coons, 1967] can be used for bilinear blending between four boundary curves, although adjacent patches may not join smoothly. The more general n -sided Gregory patches [Plowman and Charrot, 1996] are used in the ACIS modelling kernel. Lofting, a method by which a smooth surface is constructed to fit a network of curves, is a well established technique for surface specification. We present a method which, given a network of curves and topological information (vertices and faces), produces a smooth surface *approximating* the curve network.

Our approach defines the surface in two steps. First we define surface functions which meet at each vertex based on either the curves meeting at the vertex or a bivariate patch representation for each face incident on that vertex. These functions are extrapolated in a conformal space, and the results are blended together to define a smooth *vertex patch*.

Then, for each face in the surface, the *slices* from the vertex patches associated with that face are blended together. In this way, the surface is constructed from a smooth blend of a smoothly blended function — the resulting smoothness is limited only by the smoothness of the function used to describe the surface.

Our approximative approach has several advantages:

- Most importantly, curves which are not necessarily compatible (i.e. would not lie on the same surface) can be approximated with a smooth vertex patch. This accounts for all possible curve configurations — *curves do not need to meet at the same position in space*. The function describing the surface (either univariate or bivariate) does however require some form of computable extrapolation.

- The continuity of the resulting surface is theoretically only limited by the functions describing the initial surface.
- No control mesh fitting or optimisation step is required. The method works directly on the input curve network (assuming that each face is topologically a quad).
- The overlap of the vertex patches can be defined locally, allowing the designer greater flexibility in defining properties of the surface.
- The results are visually smooth.
- The method is simple to implement.

We do not know of any other method which increases surface smoothness by approximating rather than interpolating the input curves.

2 Background

Curves have been used extensively in surface design and solid modelling applications. All modelling packages provide an interface for the specification of surfaces from the definition of spline patch boundaries. For a thorough background in the numerous established methods which exist for producing lofted surfaces, the reader is referred to Piegl [1993].

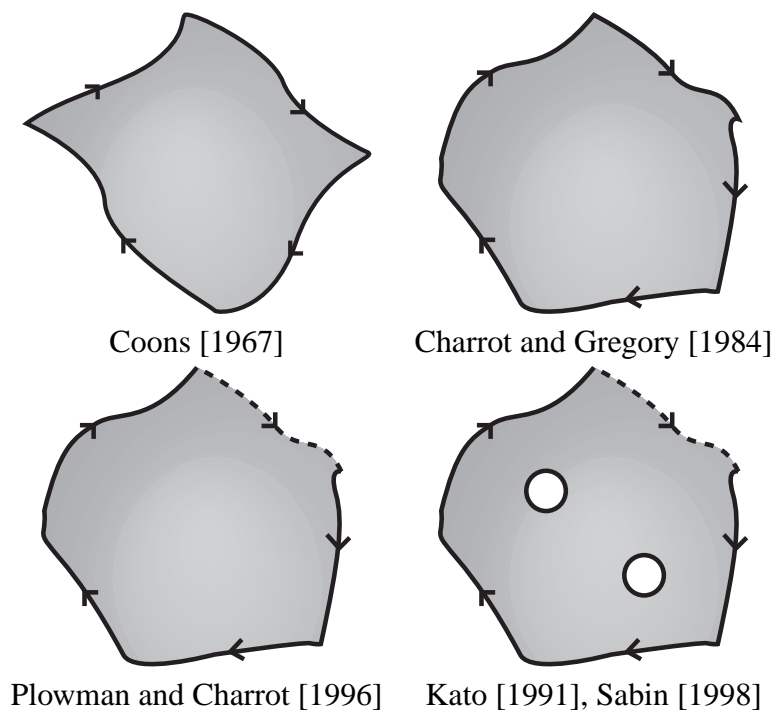


Figure 1: The evolution of the supported curve configurations in transfinite surface interpolation techniques. The 4-sided method of Coons [1967] was extended by Charrot and Gregory [1984] to 5-sided polygons, and later to arbitrary n -gons by Plowman and Charrot [1996]. Kato [1991] and Sabin [1998] further extended these approaches to support surface holes.

The problem of constructing a surface from a set of n bounding curves is well studied, originating with Coons' method [Coons, 1967]. Since its invention, it has been developed further and applied in various areas of geometric modelling and Finite Element methods by Sabin [1996]. A summary of the development of these techniques, sometimes called *transfinite interpolation*, is given in Figure 1. Numerous recent methods have been presented to fit a surface to a network of input curves, ranging from subdivision [Nasri and Abbas, 2002, Levin, 1999, Schaefer et al., 2004] to geometric diffusion using PDEs [Xu et al., 2003].

Several approaches have been pursued recently to construct subdivision surfaces interpolating a network of curves [Nasri and Abbas, 2002, Levin, 1999, Schaefer et al., 2004]. These methods typically rely on a dual coarse grid defining surface topology matching the curve network (which may be constructed during the course of the algorithm [Schaefer et al., 2004]). We too require a topological description of the surface in order to produce the surface. Our method differs inherently from subdivision in that the resulting surface has a parametric definition.

Xu et al. [2003] use a geometric diffusion model which can perform surface blends, n -sided hole filling and free form surface fitting with non-linear blends. Although this method does have attractive properties and is particularly applicable to lofted surfaces, alterations to the surface will have a global effect as there is no localised topological structure.

The process of blending together overlapping charts to construct a manifold was first proposed to the graphics community by Grimm and Hughes [1995]. In their approach an input medium, consisting of a topological structure of any genus, is converted into a manifold by blending overlapping vertex, face and edge charts with transition functions.

Grimm and Hughes' manifold construction was extended recently by Ying and Zorin [2004] to allow the construction of C^∞ manifolds from control meshes. Their method uses a similar principle to ours, but their transition function can be thought of as blending together patches defined by the edges of the grid rather than the faces as they are with our method. The proof of C^∞ continuity of transition maps and partition of unity functions provided in their paper is directly relevant to this work, as our method uses an equivalent construction for both the vertex patches and the final manifold blend.

Our method is also similar to the Moving Least Squares (MLS) surface approximation pioneered by David Levin and later developed for constructing and rendering point set surfaces by Alexa et al. [2003]. MLS uses partitions of unity to blend together local surface approximations in a similar manner to ours.

3 Terminology

In general, i and j are used to represent the indices of vertices and faces respectively. \mathcal{I}_j is used to represent the set of vertex indices at the corners of face j (in a mesh consisting of quadrilaterals, $|\mathcal{I}_j| = 4$) and \mathcal{J}_i is the set of faces incident on the vertex i (so $|\mathcal{J}_i|$ is the valence of the vertex i). Both sets are cyclic, ordered, finite and non-empty.

We define a set indexing operator $[]$ such that $\mathcal{I}_j[n]$ returns the n_{th} element of a set \mathcal{I}_j , modulo the size of the set $|\mathcal{I}_j|$. An example is given in Figure 2. We will refer to $k = |\mathcal{J}_i|$ as the valence of vertex i , often denoted as a subscript as many of the functions we use can be precomputed in the implementation for expected values of k .

For clarity, we define three separate vector spaces:

- $\mathbf{u} := (u, v) \in \mathbb{U} \equiv \mathbb{R}^2$ where \mathbb{U} is the parameter space of each patch \mathcal{S}_j . If $\mathbf{u} \in [0, 1]^2$ then it

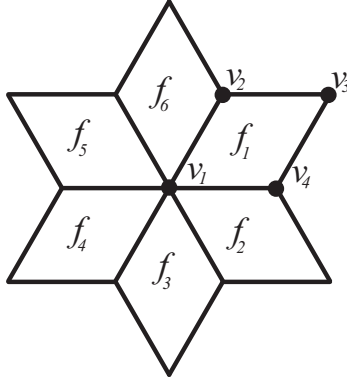


Figure 2: An example of vertex sets and face sets. Each face which borders v_1 is included in the ordered set $\mathcal{J}_{v_1} = \{f_1, \dots, f_6\}$, while each corner vertex of face f_1 is included in the ordered set $\mathcal{I}_{f_1} = \{v_1, \dots, v_4\}$.

describes a point within patch \mathcal{S}_j . These coordinates are used to describe how the surface is traversed in parameter space.

- $\mathbf{x} \in \mathbb{X} \equiv \mathbb{R}^2$ where \mathbb{X} is the “conformal space” (described in Section 4.1), and $\mathbf{x} := (x, y) \in \mathbb{R}^2$. Conformal space is used to define smooth blending between neighbouring patches.
- \mathbb{R}^3 , the Euclidean space, where the final surface lives.

The curve network is specified by three components:

- An ordered set of n_c curves $\mathcal{C} = \{c_1, \dots, c_{n_c}\}$ where $c_i : \mathbb{R}^1 \mapsto \mathbb{R}^3$ is any parametric representation.
- An ordered set of n_v vertices $\mathcal{V} = \{v_1, \dots, v_{n_v}\}$ where each vertex is a set of tuples each consisting of the index of an incident curve and a parameter value along the curve where the vertex lies. This definition allows the physical location of connected vertices to differ.
- An ordered set of n_f faces $\mathcal{F} = \{f_1, \dots, f_{n_f}\}$. Each face entry contains the indices of the vertices which bound the face.

Note that the above structure does not indicate the location of the vertices in \mathbb{R}^3 , but they are easily deduced as the average of the points they correspond to on the incident curves. Another observation is that this method will obviously work on ordinary control meshes (such as those used in subdivision) as the edges of the coarse grid could simply be defined as parametric curves.

4 Method overview

We construct a surface from a network of curves, which form the boundaries of n surface patches $S_j, j \in \{1, \dots, n\}$. The final surface takes the form of surface patches $S'_j : \mathbb{U} \mapsto \mathbb{R}^3$ which approximate the curve network, but has smooth cross border derivatives with its neighbour patches.

- A smoothly blended *vertex patch* $P_i(\mathbf{u}, j)$ is defined for each vertex i . This patch consists of separate “slices”, one for each face $j \in \mathcal{J}_i$ (as in Figure 4).

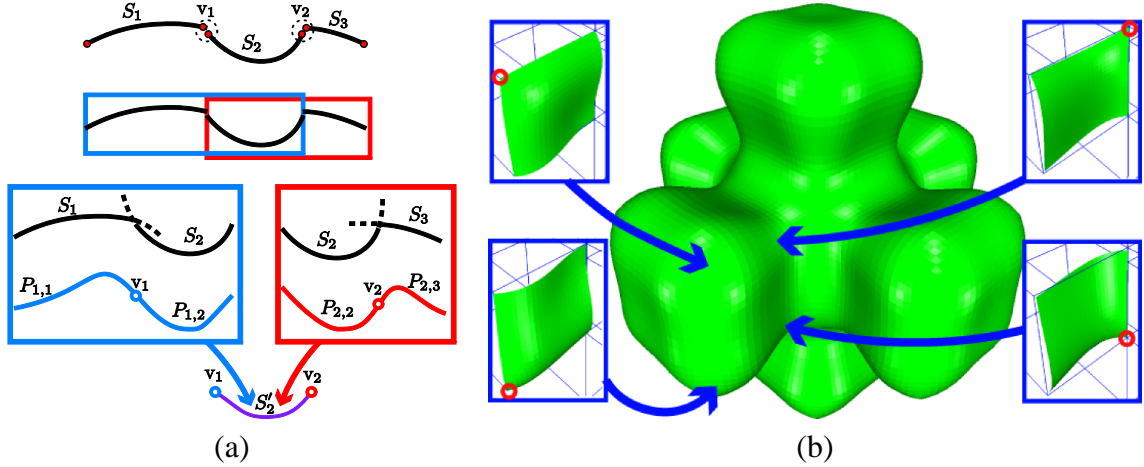


Figure 3: In (a) is a two-dimensional depiction of our method. Three parametric curve segments are defined, along with information about how they are to be connected (note that this information could be deduced geometrically). Each pair of curves associated with a vertex is used to construct the associated vertex patch. The patch is formed by extrapolating the curves and blending these together. The curve in the final surface between both these vertices is defined by blending together the vertex patch segments (in this case there are only two, one on either side of the vertex) associated with that particular curve. In (b) we show the equivalent process in \mathbb{R}^3 .

- The approximative surface construction $S'_j(\mathbf{u})$ for face j is constructed by blending together all vertex patches $P_i(\mathbf{u}, j)$ such that $i \in \mathcal{I}_j$.

This is demonstrated in the two-dimensional case in Figure 3.

4.1 Vertex patches

The smoothness of the vertex patch determines the smoothness of the final surface. The smoothness of the region is achieved by using local coordinate transforms to define how each vertex patch segment **extrapolates** into neighbouring patches. This extrapolation is achieved by a conformal space transformation for each $l \in \mathcal{J}_i$.

Each sample point \mathbf{u} is mapped into conformal (angle preserving) space using a conformal mapping function $\phi : \mathbb{U} \mapsto \mathbb{X}$. We use one of the standard conformal complex transform $\phi_k(z) = z^{A/k}$ [Feynman et al., 1989], where k is the valence of the vertex. A rotation R_l is then applied to determine the extrapolated location of the (x, y) coordinate in the local patch. The rotation is dependent on the method being used — further discussed in Section 6. The transformed point is then remapped into parameter space with an inverse conformal mapping $\phi_k^{-1}(z)$. The concatenation of functions $\phi_k^{-1} \circ R_l \circ \phi_k : \mathbb{U} \mapsto \mathbb{U}$ we call the *extrapolation transform*. The choice of the initial curve (or patch in the bivariate case) to orient with the y -axis is unimportant, although it will affect the parameterisation (which may modify the behaviour of the function λ_m introduced in Equation 2). This process is illustrated in Figure 4.

We define each vertex patch by

$$P_i(\mathbf{u}, j) = \frac{1}{\Omega} \sum_{l=1}^k \omega(\delta_k(\mathbf{u}, l)) \cdot S_{\mathcal{J}_i[l]}(\phi_k^{-1} \circ R_l \circ \phi_k(\mathbf{u})), \quad (1)$$

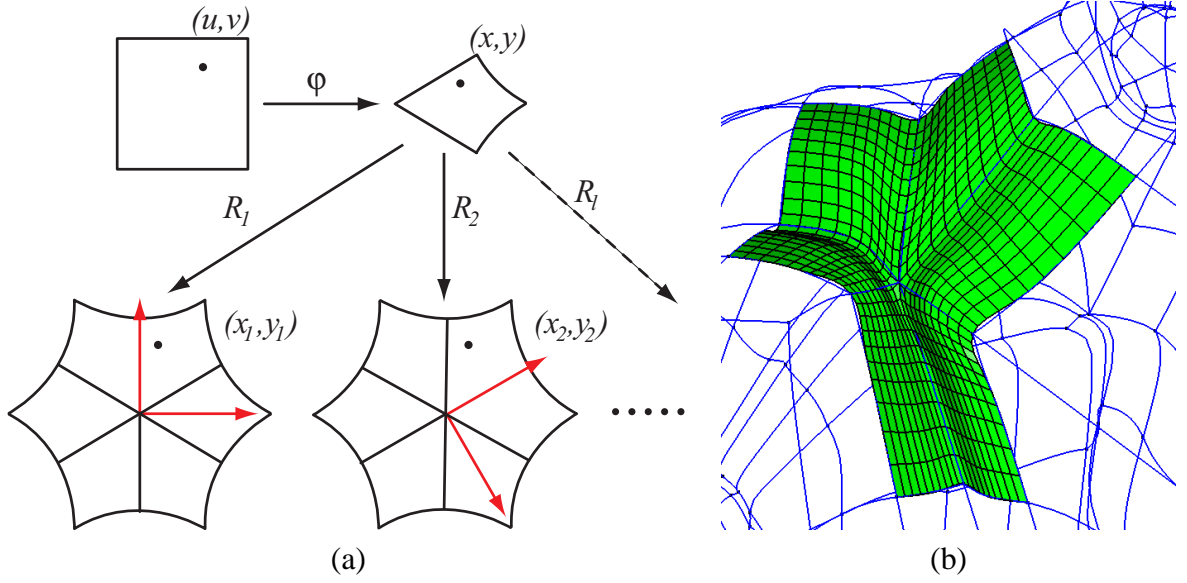


Figure 4: Extrapolating into local patch coordinates. In (a) a point in parameter space $\mathbf{u} \in \mathbb{U}$ is transformed into “conformal space” $\mathbf{x} \in \mathbb{X}$ using the conformal mapping ϕ_k . A rotation R_l is then applied to the conformal space coordinate to determine (x_l, y_l) . The rotation used here corresponds with the univariate (curve based) scheme discussed in Section 6.1. In (b) we give an example of a complex 6 vertex patch on the haunch of the bunny model in Figure 9.

where $k = |\mathcal{J}_i|$ (the valence of vertex i) and $S_j : \mathbb{U} \mapsto \mathbb{R}^3$ is some parametric sampling function to determine the position in \mathbb{R}^3 (discussed in Section 6). The weighting ω is either determined in conformal space ($\omega : \mathbb{U} \mapsto \mathbb{R}$, see Section 6.1) or in parameter space ($\omega : \mathbb{X} \mapsto \mathbb{R}$, see Section 6.2) depending on the shape of the desired blending function. We define

$$\delta_k(\mathbf{u}, l) = \begin{cases} \phi_k^{-1} \circ R_l \circ \phi_k(\mathbf{u}), & : \omega : \mathbb{U} \mapsto \mathbb{R} \\ R_l \circ \phi_k(\mathbf{u}), & : \omega : \mathbb{X} \mapsto \mathbb{R} \end{cases}$$

which gives us greater flexibility in defining how our surfaces S_j interact with each other. We set

$$\Omega = \sum_{l=1}^k \omega(\delta_k(\mathbf{u}, l)),$$

so the weighting functions form a partition of unity. An important observation is that if $\phi_k^{-1} \circ R_l \circ \phi_k$ returns a $\mathbf{u} \notin [0, 1]^2$ then the function S_j returns an extrapolated value.

4.2 Manifold construction

The goal of this method is to fit smooth surfaces to the network which best approximates the input curves. In Section 4.1 we described a method to construct smooth vertex patches $P_i(\mathbf{u}, j)$ for a vertex i , and all $j \in \mathcal{J}_i$. The surface $S'_j(\mathbf{u})$ is constructed by blending together all vertex patches with $i \in \mathcal{I}_j$. This is performed using a blending function similar to that defined in Section 6. So

$$S'_j(\mathbf{u}) = \frac{1}{\Xi} \sum_{m=1}^{|\mathcal{I}_j|} \xi(\lambda_m(\mathbf{u})) P_{\mathcal{I}_j[m]}(\lambda_m(\mathbf{u}), j), \quad (2)$$

where λ_m is a simple coordinate transformation function to ensure that the coordinate frame of consecutive patches is aligned with the first patch ($m = 1$), $\xi : \mathbb{U} \mapsto \mathbb{R}$ is a blending function based only on the parameter space coordinates of the point, and as before the partition of unity is ensured by dividing through by the total blend weight value

$$\Xi = \sum_{m=1}^{|\mathcal{I}_j|} \xi(\lambda_m(\mathbf{u})).$$

There is obviously a great deal of flexibility in the design of the bivariate blending function ξ . We have experimented with a range of variations in its construction but empirical evidence shows that altering this function produces little effect on the resultant surface except in extreme cases. For simplicity we use $\xi(u, v) = \beta(u)\beta(v)$, using a smooth blending function $\beta(t)$ defined in Section 4.3.

4.3 A smooth blending function

The quality of the resultant surface is greatly influenced by the choice of blending function for the vertex patch region. This function is used to construct the blend weights ω and ξ mentioned in Equations 1 and 2. In order to preserve the smoothness of the overall region, we use the infinitely differentiable blending function

$$\beta(t) = \exp\left(4 + \frac{4}{t^2 - 1}\right), \quad (3)$$

defined over the open interval $t \in (-1, 1)$ (see Figure 5).

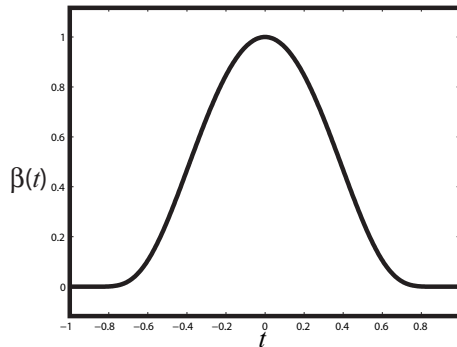


Figure 5: The blending function defined in Equation 3.

5 Continuity

In this section we show how the continuity of the surface is derived. A set M has a manifold structure if there exists a collection of open domains C_n and associate mapping functions $\alpha_n : C_n \mapsto M$ (together called *charts*), where α_n is one-to-one and the images $\alpha_n(C_n)$ cover M . M is a C^∞ manifold if the transition maps from chart to chart $t_{mn} = \alpha_m^{-1} \circ \alpha_n$ defined for pairs of

chart for which $\alpha_n(C_n)$ and $\alpha_m(C_m)$ intersect are C^∞ . In our case $S'_j(\mathbf{u})$ represents the transition map.

The two stage process consists of Equation 1 and Equation 2. The components R_l , ϕ_k , λ_m and the partitions of unity [Ying and Zorin, 2004] are infinitely smooth and invertible. The continuity therefore depends on the behaviour of the surface description function S_j . Two options are presented for describing the surface behaviour as it is extrapolated over the region:

- a univariate approach which samples values in \mathbb{R}^3 from the input curves meeting at the given vertex, and
- a bivariate approach which extrapolates the patches from each face incident on the vertex over the entire domain with a smooth and invertible “flattening” function $\tilde{\phi}^{-1}$ (defined in Section 6.2).

In both cases these methods are constrained by the continuity of the representation of the extrapolated surface. In the case of the univariate extrapolation the continuity is constrained by the designer, who could use higher order curves to specify higher order continuity. Similarly, the bivariate case would be constrained by the type of patch used. Therefore the continuity of the resulting surface is limited only by the continuity of the functions which are extrapolated.

6 Vertex sampling

The function $S_j : \mathbb{U} \mapsto \mathbb{R}^3$ returns a point which best defines the extrapolation of a surface over the vertex patch region. We define two possible surface functions $S_j(\mathbf{u})$: one based on the curves meeting at the vertex, and one based on a bivariate patch extrapolation.

6.1 Univariate vertex patches

In this section we present a simple univariate scheme to define a surface function for $S_j(\mathbf{u})$ which extrapolates the *curves* into the blended region. If the curve $c_l(t)$ is the curve corresponding with the u axis of face $l \in \mathcal{J}_i[l]$ we use

$$S_{\mathcal{J}_i[l]}(\mathbf{u}) = c_l(u). \quad (4)$$

We use a conformal space blend $\omega_{curve} : \mathbb{X} \mapsto \mathbb{R}$ and define a blending shape as follows:

$$\omega_{curve}(\mathbf{x}) = \begin{cases} 1 & : x = 0, y \geq 0 \\ \beta(x) & : x \neq 0, y > 0 \\ \beta(x)\beta(y) & : -1 < x < 1, -1 < y < 0 \\ 0 & : \text{otherwise} \end{cases} \quad (5)$$

In order to orient the blending function shape we define the rotation $R_l = 2l\pi/k + \pi/2$, and we normalise the resulting values of (x, y) to ensure that the blending function reduces to zero at the corners of the conformal region (i.e. the location $\phi(1, 1)$, rotated by R_l). The resulting shape is a blend that falls off evenly to the left and right of the vertical, and falls away radially when $y < 0$.

This method causes bunching due to the distribution of curves in parameter space, and the results are not generally visually pleasing (see Figure 6). However, the smoothness of this

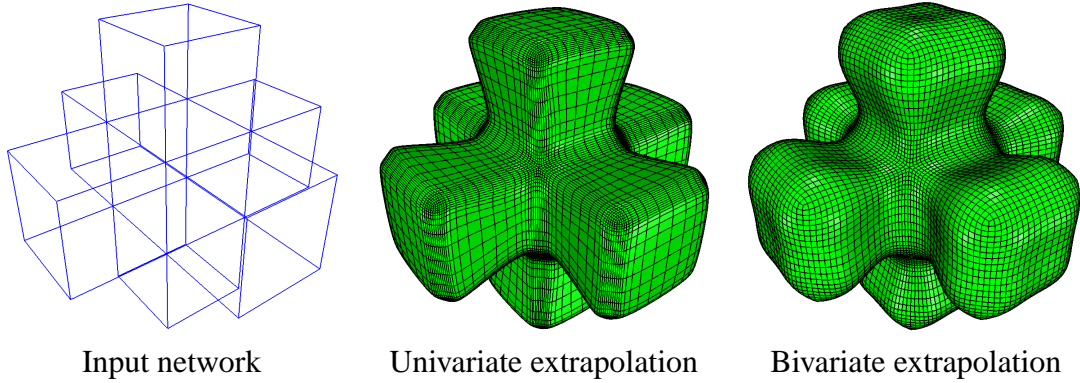


Figure 6: A comparison of the univariate and bivariate ($s = 0.5$) extrapolation methods (see Section 6). Note in particular the parameter line bunching of the univariate (curve) extrapolation method.

method is only limited by the smoothness of the input curves. Should a method be found to evenly distribute the samples this would lead to a better behaved surface. This is an area for future work.

6.2 Bivariate extrapolation

For better visual results, we instead define $S_j(\mathbf{u})$ to return a point on a bivariate patch representation bounded by the four input curves associated with the patch. Any bivariate patch can be used for this operation, such as a bivariate b-spline patch network or even simple flat quadrilateral patches, yielding surfaces which are C^∞ . For simplicity we have used a Coons patch Coons [1967], limiting the continuity of the resulting scheme to at most C^1 .

Such a bivariate patch representation provides a mapping $C(\mathcal{K}, \mathbf{u}') : \mathbb{U} \mapsto \mathbb{R}^3$, where \mathcal{K} is the set of four curves bounding the current face. We require some function $\phi_k^{-1} : \mathbb{X} \mapsto \mathbb{U}$ to provide the extrapolated parameter space coordinates for C . The function $\phi_k^{-1} = z^{k/4}$ would be an obvious choice, but unfortunately the extrapolation information would be lost. This is difficult to visualise, but it is the equivalent of converting a higher valency vertex back to a 4 valency vertex, causing one or more faces to be folded *underneath* an existing face.

We use an alternative “flattening” approach to define $\tilde{\phi}_k^{-1}$, shown in Figure 7, using a complex transform using a smooth mapping between the angles of the form $\theta + \alpha \sin \theta$. We define the function in complex notation as follows:

$$\tilde{\phi}_k^{-1}(z) = |z|e^{i(\theta + \alpha_k \sin \theta)} \quad (6)$$

where $\alpha_k = \frac{\pi(k-4)}{4k \sin(\pi/k)}$, and θ is the angle between the point \mathbf{x} and the start curve in conformal space. In the case of the bivariate construction, we apply the rotation $R_l = 2l\pi/k$.

We define the bivariate blending function in parameter space $\omega_{coons} : \mathbb{U} \mapsto \mathbb{R}$ to be

$$\omega_{coons}(\mathbf{u}) = \beta(s(0.5 - u)) \beta(s(0.5 - v)).$$

The value of s is a user controlled parameter which can be used to control the level of overlap between neighbouring patches in the vertex patch. This is demonstrated in Figure 8. We have found through experimentation that $s = 0.8$ produces surfaces of a respectable visual quality (see Figure 10). However this value can be specified by the designer for each vertex in the surface. Adaptively determining vertex overlap weights is a topic for future work.

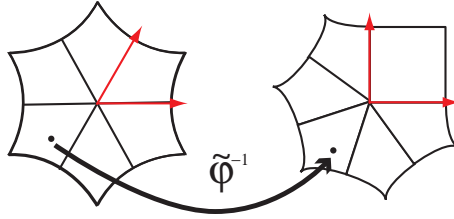


Figure 7: Inverting the conformal map using $\tilde{\phi}^{-1}$. The arrows indicate the coordinate frame of the patch from which we are extrapolating. This patch is “flattened”, causing other patches to be squashed proportionally.

7 Implementation

The method is simple to implement — the vertex patch generation code is only a few hundred lines of code. Key to the continuity of the resulting surface is the parametric representation of the input curves. We have allowed for both algebraic curves (C^∞) to evaluate the smoothness of the method, and Catmull-Rom splines (C^1) for compatibility with existing models.

As the method requires a topology consisting only of quad faces, surfaces consisting of other polygons must first be topologically subdivided with one iteration of an interpolative subdivision. We use a topological alteration analogous to the Catmull Clark scheme [Catmull and Clark, 1978] but use differing rules to compute the locations of new vertices. We evaluate edge vertices directly from the input curves, and face vertices from the midpoints of a n -sided patch using the method of Plowman and Charrot [1996]. New curves can be constructed by sampling points in the parametric coordinates of the n -sided patch and fitting a spline of the desired order. One stage of this subdivision is shown in Figure 9.

Sharp edges — either specified per vertex or detected by topological discontinuities — are handled by using a simple bivariate patch interpolating the four input curves. This would blend into the remaining vertex patches corresponding to the same face in Equation 1.

Geometry is defined by regularly sampling coordinates within faces. This will obviously produce unpleasant parameter bunching within small faces, but could be rectified by using the Least-Squares approach of Ying and Zorin [2004], although such an approach would require a redefinition of the final blending step in Equation 2.

8 Conclusion and future work

We have presented a method of constructing a smooth surface approximating a network of input curves by first constructing smooth vertex patches and then blending together the segments corresponding to the corners of each face. Each stage uses a manifold based construction. Our main contributions have been:

- a two step manifold based approach for producing smooth surfaces by extrapolation (Section 4),
- a univariate surface function approach to the method which blends together the curves meeting at each vertex to form a surface (Section 6.1), and

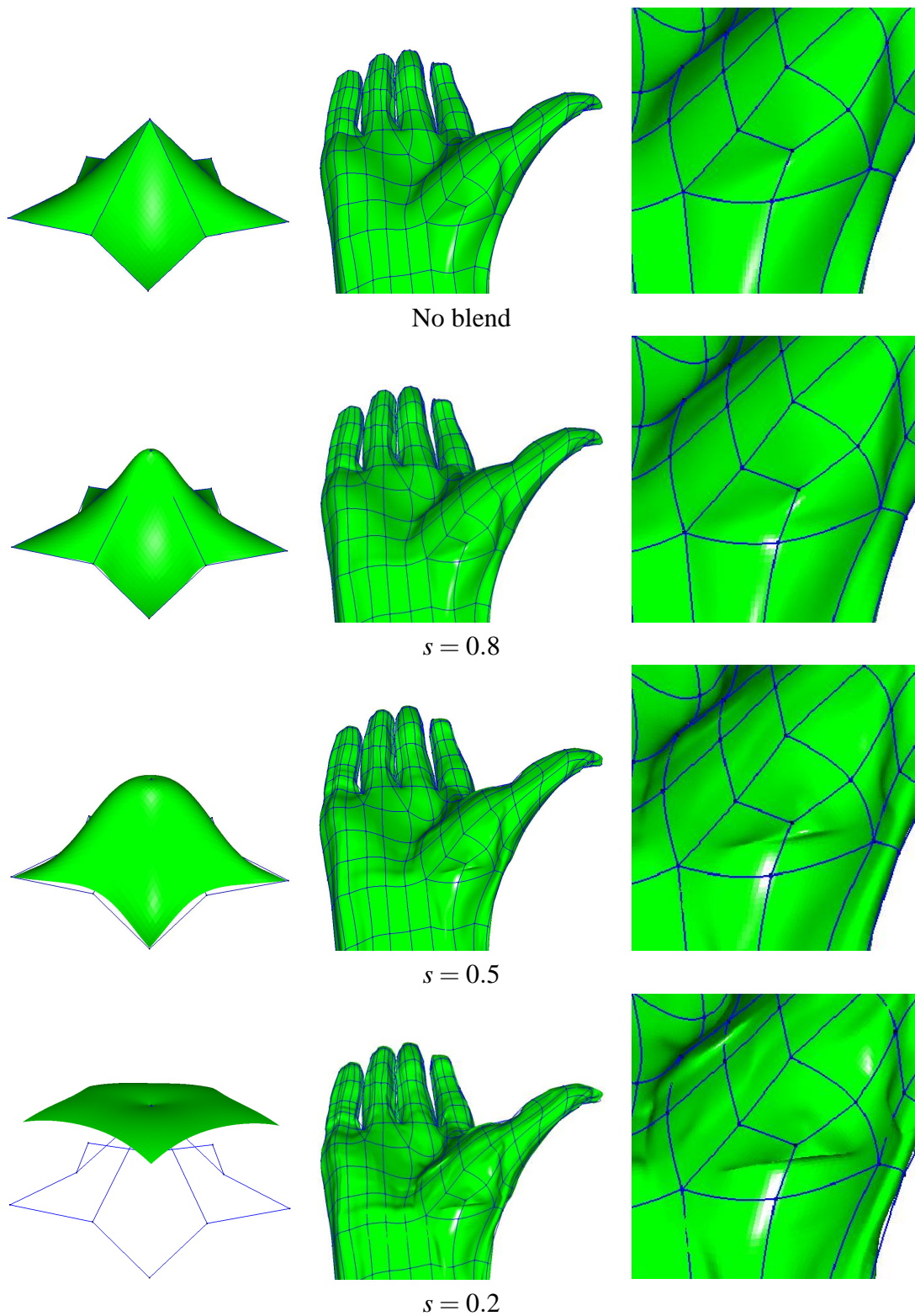


Figure 8: The result of varying the parameter s (the overlap) on a 5 valency vertex patch and the *hand* model. Note that a crease at the base of the thumb (right column) becomes progressively more prominent as the overlap increases.

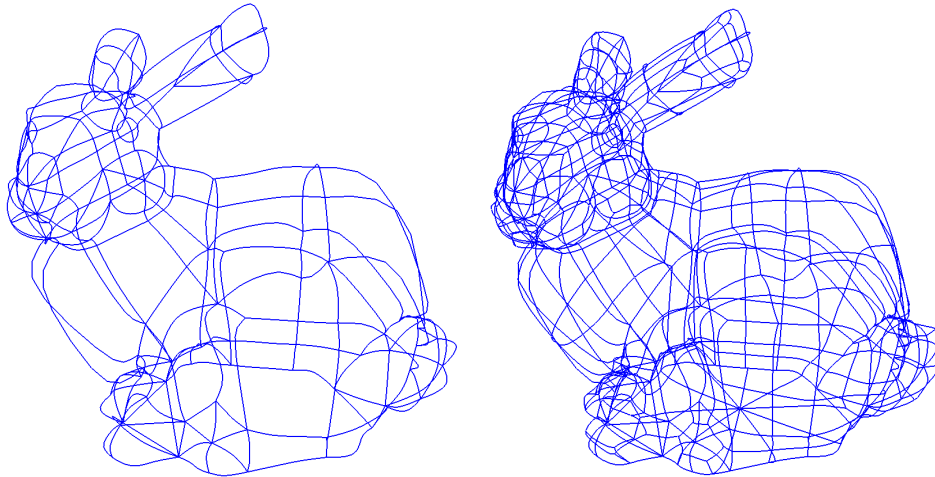


Figure 9: One subdivision step is applied to the bunny model in order to convert the n -sided patches to only 4-sided patches. This is described in Section 7.

- a bivariate surface function approach which computes the extrapolated coordinate of a standard patch representation by using a smooth invertible function $\tilde{\varphi}^{-1}$ (Section 6.2).

There are a number of shortcomings of this technique, giving rise to several avenues for future work:

- Coons patches do not extrapolate well when vertices have a valence greater than 8. We have experimented with Mean Value Coordinates (see Floater [2003] and Ju et al. [2005]) as a method to define extrapolation patches which do not overlap with high valence. Further research in this direction is required.
- The method does not prevent self intersection, which arises naturally in regions where neighbouring patches are of significantly differing sizes. This phenomenon can also cause undesirable surface artifacts (such as bumps). This could be repaired by adding geometric sensitivity in the blending process, causing parametric curves to be weighted inversely based on their length.
- At present, the behaviour of corners are limited to either being completely sharp or completely smooth, so complex blending configurations arising from realistic modelling are currently not possible (several examples are shown by Schaefer et al. [2004]).
- Topological alterations of the surface patches, such as holes or seams, are not currently supported. It is probable that these could be supported in an approximative fashion defining holes with a method like that of Kato [1991] or Sabin [1998] in each bivariate patch extrapolated over the vertex patch region.
- Performing extrapolation can be problematic (likened by [Press et al., 1992] to “turning lead into gold”). It is a simple task to construct splines which curl sharply back on themselves unpredictably. While a method of performing robust extrapolation is not a topic of this paper, developments in this field will have a strong impact on the quality and predictability of the surfaces that our method produces.

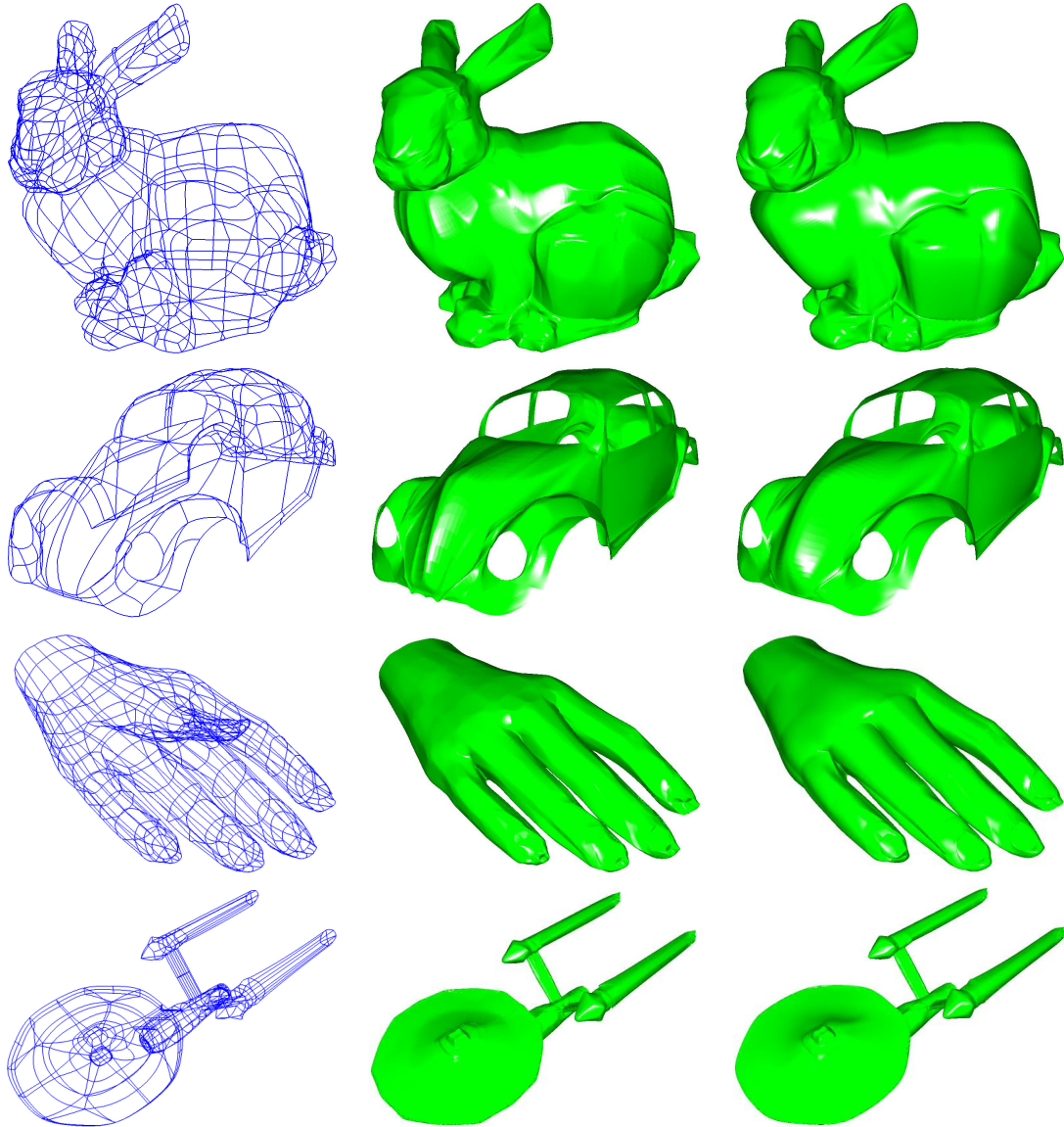


Figure 10: Some results of our technique. The first column shows the input curve network after one level of subdivision (with the method described in Section 7). The second and third columns illustrate the resultant surface with the univariate and bivariate extrapolation methods respectively. All results were generated with the overlap value $s = 0.8$. Note that some bulging artifacts arise on the surface where neighbouring patches are of significantly differing sizes (most obvious at the front grille of the *bug* model). This can be rectified by adapting the overlap between patches.

9 Acknowledgements

The authors would like to thank Malcolm Sabin for his many insightful comments and assistance which provided guidance to this project. We would also like to thank Scott Schaeffer for the use of the *bunny*, *hand*, *car* and *enterprise* models, and Cindy Grimm for assistance in finding a copy of her dissertation. This work was supported by the EPSRC Grant GR/S67173/01.

References

- Mark Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9(1), January 2003.
- E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10(6):350–355, 1978.
- P. Charrot and J. Gregory. A pentagonal surface patch for CAGD. *Computer Aided Geometric Design I*, pages 87–94, 1984.
- Steven Coons. Surfaces for computer aided design of space forms. Technical Report MAC-TR-41, Project MAC, Massachusetts Institute of Technology, 1967.
- R. P. Feynman, R. B. Leighton, and M. Sands. *The Feynman Lectures on Physics*, volume 1. Addison-Wesley, Redwood City, CA, 1989.
- Michael S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20(1):19–27, 2003.
- Cindy Grimm and John Hughes. Modeling surfaces of arbitrary topology. *Proceedings of SIGGRAPH*, pages 359–369, August 1995.
- Tao Ju, Scott Schaefer, and Joe Warren. Mean value coordinates for closed triangle meshes. In *Proceedings of SIGGRAPH*, pages 561–566, 2005.
- K. Kato. Generation of n -sided surface patches with holes. *Computer Aided Design*, 23(10): 676–683, December 1991.
- Adi Levin. Interpolating nets of curves by smooth subdivision surfaces. *Proceedings of SIGGRAPH*, pages 57–64, 1999.
- Ahmad Nasri and A. Abbas. Designing Catmull-Clark subdivision surfaces with curve interpolation constraints. *Computers and Graphics*, 26:393–400, 2002.
- Les Piegl, editor. *Fundamental Developments of Computer-Aided Geometric Modeling*. Academic Press, 1993.
- D. Plowman and P. Charrot. A practical implementation of vertex blend surfaces using an n -sided patch. In Glen Mullineux, editor, *The Mathematics of Surfaces VI*, pages 67–78. Clarendon Press, 1996.

- W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in FORTRAN: The Art of Scientific Computing*, chapter Richardson Extrapolation and the Bulirsch-Stoer Method, pages 718–725. Cambridge University Press, Cambridge, England, 2nd edition, 1992.
- M. Sabin. Transfinite surface interpolation. In Glen Mullineux, editor, *The Mathematics of Surfaces VI*, pages 517–534. Clarendon Press, 1996.
- M. Sabin. Further transfinite surface developments. In Robert Cripps, editor, *The Mathematics of Surfaces VIII*, pages 161–174. Information Geometers Ltd, 1998.
- S. Schaefer, J. Warren, and D. Zorin. Lofting curve networks using subdivision surfaces. In *Eurographics Symposium on Geometry Processing*, pages 105–116, 2004.
- Guoliang Xu, Pan Qing, and Chandrajit Bajaj. Discrete surface modeling using geometric flows. Technical Report ICM-03-013, ICES Technical Report 03-38, Institute of Computational Mathematics and Scientific/Engineering Computing, Chinese Academy of Sciences, 2003.
- Lexing Ying and Denis Zorin. A simple manifold-based construction of surfaces of arbitrary smoothness. In *Proceedings of SIGGRAPH*, pages 271–275, August 2004.