Reconstruction of Smooth Surface Suitable for LOD Model Construction

Jaroslav Semančík
e-mail: Jaroslav.Semancik@mff.cuni.cz

KSVI MFF UK Praha

Abstract

The paper aims (re)construction of a smooth surface over a given triangular mesh (usually scanned from a real object). The surface composed of parametric patches is intended to be a subject of future reduction (similar to mesh decimation) and other level-of-detail (LOD) techniques. The patches forming the surface must be flexible enough to catch arbitrary topology of input mesh and reasonably compact as well, since meshes scanned from real objects often consist of huge number of triangles. We discuss appropriate kind of patches, explain triangular Coons patches as a result and propose their construction to obtain "fair" surface interpolating mesh vertices smoothly across the patch boundaries. Images from an experimental implementation illustrate our approach.

Keywords: surface reconstruction, smooth surface, parametric patch, triangular Coons patch, reduction, level of detail

1 Introduction

The most of objects in the real world is rounded. In spite of this, the current de facto industry standard in boundary representation of geometric models is a triangular mesh consisting of flat triangles and sharp edges. It is pretty simple and universal, but now there are several good reasons why to use mathematically described smooth surfaces to represent geometric models:

- **accuracy and efficiency** (in the data-size sense).
  Triangular mesh for a scanned real object commonly presents a very large data set reaching up to gigabyte. Even if the model is not especially complex, a high number of small triangles is needed to preserve its curved shape accurately. It is a kind of inevitable oversampling of the surface; though a single smooth patch could do the job well. Its another advantage is smoothness at any scale level while the triangular mesh has always sharp edges in a close look.

- **advancements in theory of geometric modeling**.
  After the basic rectangular patches have been developed (and hence suitable mainly) for car and aircraft industry in 60- and 70-ties, the recent research resulted in triangular and multisided patches [3], triangular NURBS

surfaces [10] or subdivision surfaces and it is closely related to approximation theory.

- **hardware acceleration of parametric patches rendering** became a reality in commercially available graphics cards.
  It can be implemented as a kind of ray casting or an automatic adaptive tesselation of the patch. The automatic adaptive tesselation on the graphics board helps to prevent the huge number of triangles streaming into the graphics card through the bus – the bottleneck of current systems.

- **analogy with low-level → high level changeover** in programming: triangular mesh → smooth surfaces (with possible sharp edges).
  The situation is very similar as in the programming several decades ago. The change from low level assembler programming to higher level languages made possible to develop more complex software on the higher abstract level whose implementation wouldn't be on the lowest level feasible. The higher level of abstraction allows to formulate and solve more sophisticated problems without distraction by 'technical details'. In computer graphics is the role of higher programming languages taken over by mathematically sophisticated geometric elements – the building stones of the models e.g. parametric patches, while triangular meshes remains as an underlying technical layer handled by hardware. In practice – an application should work with smooth surfaces, while the tesselation in the necessary quality and the rendering are provided by a graphics card.

A number of techniques for triangular mesh modeling has been developed through the years, including level-of-detail (LOD) techniques. LOD idea resides in rendering of distant objects as coarser models, while the near ones finer. Thus, each object has several models with decreasing quality (called *levels of detail*) and the one to render is selected according to the distance of the object in a virtual world. A better method proposed by Hoppe in [4] as a *progressive mesh* represenation eliminates so-called "object popping" when the level is changed and allows smooth transition between the full-detailed model and the coarest one.

Similar techniques could be used for smooth surface models. The LOD representation is built from full model by progressive reduction preserving its overall shape. So the full smooth model must be constructed first.

2 The Goal and Choices

Obtaining a model of a real object is a kind of *reverse engineering* problem. The object is scanned by a 3D scanner first (usually mechanic or laser), producing a cloud of discrete points in space (so-called scattered data) lying on the object surface. A triangular mesh is constructed on the points then; a number of methods has been developed to do that (see [11] for an overview of them). Thus, output of traditional surface reconstruction process is a triangular mesh (usually very dense to catch all the details).

We assume triangular mesh as our input (which provides also information about topology of the surface against plain scattered data) and we construct a smooth surface passing through its vertices and having the same topology.

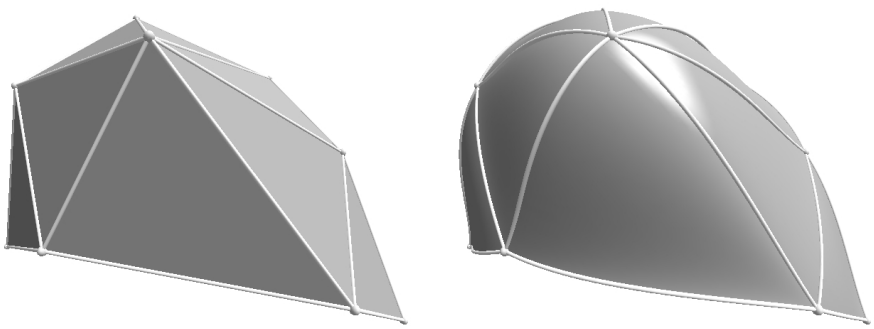

Figure 1: Illustration of a smooth surface reconstruction from a mesh portion. The patch boundaries are the blends of circular arcs (see section 3.2).

2.1 Requirements

There is a wide variety of smooth surface kinds in computer graphics and geometric modeling. Criteria for our choice are mainly:

- **flexibilty**, i.e. ability to catch both smooth curved surfaces and sharp edges, open and closed models, model composed of several independent components, possibility to define texture or material properties of the surface, etc.
- **efficiency** in two senses: (1) size of data needed to describe the surface, and (2) computational time for the basic operations related to the surface (e.g. surface point and derivatives evaluation → rendering, amount of calculations needed for approximation by that kind of surface, inside/outside test, etc.).

Besides the above two we should consider connection to technical norms and standards and difficulty of conversions from/to the commonly used representations

2.2 Choices

There are several alternatives to try to satisfy the above requirements (or at least the most of them):

- **single prametric patch**
  Shape of the most common parametric patches, e.g. Bezier, Gregory, B-spline or NURBS is determined by a grid of control points. Coons patches are suitable to interpolate given boundary curves and to fill the patch interior. They are given by arbitrary boundary curves sharing the vertices.

  Regardless of the particular type of parametric patch they all are limited to a very simple topology, equivalent (homeomorphic) to a rectangle, cylinder or torus in case of a rectangular patch and a triangle or cone in case of triangular patch. That is by far not sufficient for real models.

- **net of parametric patches**
  Model with arbitrary topology can be represented in this manner – as a mesh of parametric patches sharing the boundaries. The smoothness across the boundaries between the patches is usually desired and the individual patches must be constructed in such way to satisfy it.

- **subdivision surface**
  Subdivision surface is a modern type of smooth surface representation defined as a limit surface of the base (or control) mesh refinement steps. The refinement steps are performed according to a particular scheme (e.g. Doo-Sabin, Catmull-Clark, Loop), resulting in a mesh with higher number of smaller faces. Subdivision surface is a piecewise B-spline surface of arbitrary topology given by the topology of the base mesh. Subdivision surfaces were extended by sharp features by Hoppe in [5].

and finally implicit and volume models. We can further consider triangular vs. quadrangular and regular vs. irregular structure of the net.

Quadrangular patches are rapid to evaluate and the underlying curve algorithms can be employed. The convenient properties are, however, only in the two isoparametric directions. The triangular-domain patches are mathematically more natural generalization of the curves and though more expensive to evaluate, their properties are equally distributed in every direction. They are based on barycentric coordinates.

Regular meshes are usually result of a simulation processes and their handling is more efficient, but the real objects are most often scanned to a more flexible irregular mesh.

With all above relevant aspects in mind and considering irregular triangular mesh as input, the irregular net of triangular patches was chosen for smooth surface representation.

There are several works about reconstruction of smooth surface (see [2], [6], [9] and [7]). Our task is specific among them by intention to apply LOD techniques to created surface and assumption of very large input models (up to hundreds of thousands triangles). Hence, we must lay a particular stress on efficient, compact patches, suitable for progressive model reduction and avoid usual global surface fairing processes which often have quadratic complexity.

3 Smooth Surface Reconstrucion

To find a smooth surface passing through given vertices is an *interpolation* problem. Surface interpolation of arbitrary triangulations by parametric patches is not a simple task (see [1]). The main difficulty is to achieve the desired smoothness between adjacent patches. The shape of individual patches is determined from the tangent planes (i.e. from normals) in the mesh vertices, either given or estimated.

Construction of the surface usually runs in two steps:

1. A network of curves – the future patch boundaries is created first, over the mesh edges. A curve is constructed for each mesh edge and a tangent plane field along the boundary is defined, besides the curve itself. It assignes a plane to each point of the curve.

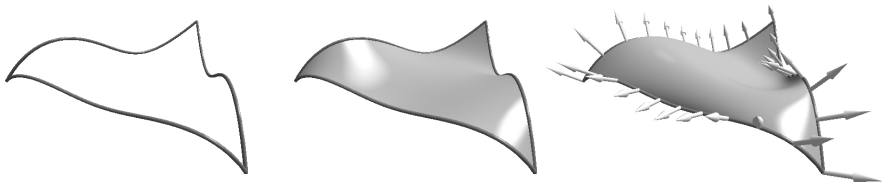

Figure 2: Three given curves sharing their endpoints on the left and corresponding Coons G^0 and G^1 interpolants respectively. Given normal fields are shown on the right image.

2. Interiors of the constructed 'curved triangles' are filled by patches afterwards. The patches must satisfy the prescribed tangent planes along the boundaries. Thus, both patches with common boundary have the same tangent plane along the entire curve and a G^1 smoothness is guaranteed. Note that tangent planes must be consistent in vertices, i.e. tangent plane fields must interpolate tangent planes in mesh vertices.

The most often used surface interpolants based on so-called *split domain schemes* divides each face to several patches. It produce too many patches for larger models. We use a different approach instead, so-called *convex combination schemes* (both schemes are explained in [7]) constructing one patch per face.

We describe step 2 – the patch construction first, to get a notion about desired properties of boundary curves. Next, we find a curve with such properties.

3.1 Triangular Coons Interpolant

Consider three curves sharing their endpoints as shown in Figure 2, thus forming a curved triangle. A triangular interpolant is a surface that fills the interior of that triangle and interpolates the given curves (i.e. they become the surface boundaries). Since the given curves are arbitrary curves, it is so-called *transfinite* (i.e. infinitely-dimensional) interpolation. Besides the three curves the input data can incorporate tangent planes (represented as normals) along those curves to achieve smooth (G^1) interpolation across the boundaries. The constructed patch $\mathcal{P}\mathbf{X}(u, v, w)$ is a parametric surface with barycentric coordinates as parameters and it is an image of a domain triangle $u+v+w = 1,\ \ u, v, w \geq 0$. We denote the vector of barycentric coordinates as $\mathbf{u} = (u, v, w)$ further.

The basic principle of the triangular Coons interpolant is a construction of three simple auxiliary patches, each of them interpolating one pair of boundary curves and blending those patches together in such a clever way, that the result interpolates all three curves.

Let the three given curves be the boundaries $\mathbf{X}(0, v, w)$, $\mathbf{X}(u, 0, w)$ and $\mathbf{X}(u, v, 0)$ of an unknown patch $\mathbf{X}(\mathbf{u})$. We construct three ruled surfaces – one for each pair of boundary curves as shown in Fig. 3 (using notation of [3]):

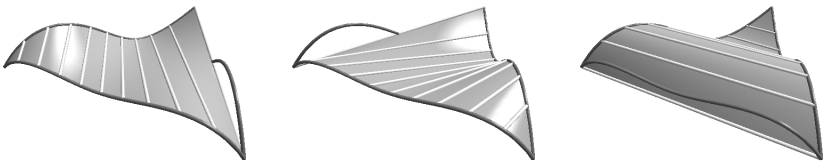

Figure 3: Three ruled surfaces $\mathcal{P}_1\mathbf{X}$, $\mathcal{P}_2\mathbf{X}$, $\mathcal{P}_3\mathbf{X}$ – the auxiliary patches of Coons interpolant.

$$\begin{aligned}
\mathcal{P}_1\mathbf{X} &= (1-r)\mathbf{X}(u,0,1-u) + r\mathbf{X}(u,1-u,0)\\
\mathcal{P}_2\mathbf{X} &= (1-s)\mathbf{X}(1-v,v,0) + s\mathbf{X}(0,v,1-v)\\
\mathcal{P}_3\mathbf{X} &= (1-t)\mathbf{X}(0,1-w,w) + t\mathbf{X}(1-w,0,w)\\
\text{denoting } r = \frac{v}{v+w} &\qquad s = \frac{w}{w+u} \qquad t = \frac{u}{u+v}.
\end{aligned} \tag{1}$$

Note that $v + w = 1 - u$, $w + u = 1 - v$ and $u + v = 1 - w$. The convex combination

$$\mathcal{P}\mathbf{X} = u\mathcal{P}_1\mathbf{X} + v\mathcal{P}_2\mathbf{X} + w\mathcal{P}_3\mathbf{X} \tag{2}$$

interpolate all three boundary curves, (see the middle of Figure 2). Actually the interpolant $\mathcal{P}\mathbf{X}$ is a blend of $\mathcal{P}_1\mathbf{X}$, $\mathcal{P}_2\mathbf{X}$ and $\mathcal{P}_3\mathbf{X}$ with some variable weight functions. Each of the three ruled surfaces interpolates two boundaries, but not the third one, thus the corresponding weight function must be zero on that boundary; u, v, w are such functions and they are shown in Figure 4.

To extend that scheme to G^1 and incorporate boundary tangent planes, we modify the construction of the auxiliary patches. Instead of the line segment ribs forming a ruled surface we use curved ribs interpolating also normals in the endpoints, besides the endpoints themselves (see Figure 5). Thus normal fields $\mathbf{X}_n(0,v,w)$, $\mathbf{X}_n(u,0,w)$ and $\mathbf{X}_n(u,v,0)$ are required as input besides the boundary curves.

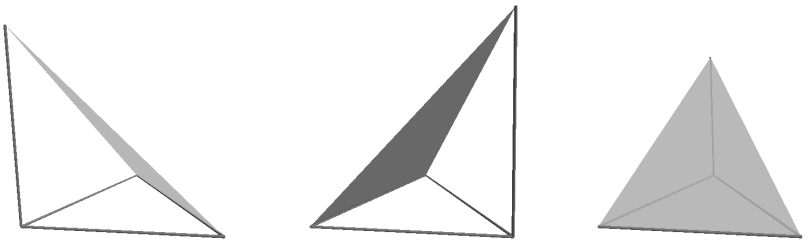

Figure 4: Linear weight functions $f(\mathbf{u}) = u$, $g(\mathbf{u}) = v$, $h(\mathbf{u}) = w$ used for G^0 interpolant. The weight of individual patches falls from 1 in a triangle vertex to 0 on the boundary across. Thus the resultant surface is almost identical to a particular patch in the vertex neighbourhood while it is absolutely unaffected by that patch on the boundary across.

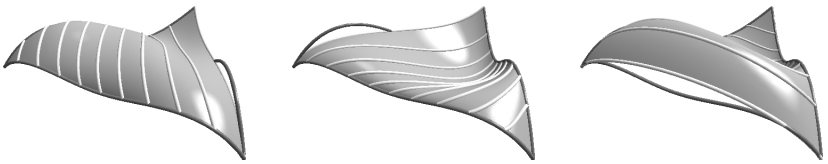

Figure 5: Three auxiliary patches $\mathcal{P}_1\mathbf{X}$, $\mathcal{P}_2\mathbf{X}$, $\mathcal{P}_3\mathbf{X}$ for G^1 interpolant construction. The curved ribs respect given boundary normal fields.

We assume a curve construction operator $\mathbf{C}$ giving for two points $\mathbf{P}_0$, $\mathbf{P}_1$ and their respective normals $\mathbf{n}_0$, $\mathbf{n}_1$ a curve $\mathbf{C}[\mathbf{P}_0, \mathbf{P}_1, \mathbf{n}_0, \mathbf{n}_1](t)$ interpolating them, i.e. $\mathbf{C}(0) = \mathbf{P}_0$, $\mathbf{C}(1) = \mathbf{P}_1$, $\mathbf{C}'(0) \cdot \mathbf{n}_0 = 0$ and $\mathbf{C}'(1) \cdot \mathbf{n}_1 = 0$.[1]

Rewriting the formula (1) to use curved ribs instead of line segments, i.e. to use a more general $\mathbf{C}$-curve interpolation instead of linear, yields:

$$\begin{aligned}
\mathcal{P}_1\mathbf{X} &= \mathbf{C}[\mathbf{X}(u, 0, 1-u), \mathbf{X}(u, 1-u, 0), \mathbf{X}_n(u, 0, 1-u), \mathbf{X}_n(u, 1-u, 0)](r) \\
\mathcal{P}_2\mathbf{X} &= \mathbf{C}[\mathbf{X}(1-v, v, 0), \mathbf{X}(0, v, 1-v), \mathbf{X}_n(1-v, v, 0), \mathbf{X}_n(0, v, 1-v)](s) \\
\mathcal{P}_3\mathbf{X} &= \mathbf{C}[\mathbf{X}(0, 1-w, w), \mathbf{X}(1-w, 0, w), \mathbf{X}_n(0, 1-w, w), \mathbf{X}_n(1-w, 0, w)](t)
\end{aligned} \tag{3}$$

where r, s, t are defined as in (1). It is not difficult to show,[2] however, that the convex combination (2) of these patches does not interpolate tangent planes. Different blending functions are needed.

Let us write the blend (2) in a general fashion:

$$\mathcal{P}\mathbf{X} = f\mathcal{P}_1\mathbf{X}(\mathbf{u}) + g\mathcal{P}_2\mathbf{X}(\mathbf{u}) + h\mathcal{P}_3\mathbf{X}(\mathbf{u}). \tag{4}$$

Consider what are the properties required for the blending functions f, g and h:

$$(\forall \mathbf{u} = (u, v, w), \quad u + v + w = 1) \qquad f(\mathbf{u}) + g(\mathbf{u}) + h(\mathbf{u}) = 1, \tag{I}$$

$$f(\mathbf{u})\big|_{u=0} = 0, \qquad g(\mathbf{u})\big|_{v=0} = 0, \qquad h(\mathbf{u})\big|_{w=0} = 0. \tag{II}$$

The first requirement expresses affine correctness of the blend and the second one boundary interpolation. Example of blending functions satisfying (I) and (II) are $f(\mathbf{u}) = u$, $g(\mathbf{u}) = v$, $h(\mathbf{u}) = w$ used in (2). Wee need to add one more property to achieve interpolation of tangent plane besides, i.e. to make the derivatives on a boundary independent from the opposite auxiliary patch. Derivatives of each blending function must be zero in every direction $\mathbf{d}$ on the non-interpolated boundary:

$$\mathrm{D}_{\mathbf{d}} f(\mathbf{u})\big|_{u=0} = 0, \quad \mathrm{D}_{\mathbf{d}} g(\mathbf{u})\big|_{v=0} = 0, \quad \mathrm{D}_{\mathbf{d}} h(\mathbf{u})\big|_{w=0} = 0. \tag{III}$$

The blending functions from (2) do not satisfy that. We show different ones based on Bernstein polynomials and satisfying all (I), (II) and (III). Triangular

[1] We propose some curve constructions in next section.

[2] Calculate derivatives on the patch boundaries.

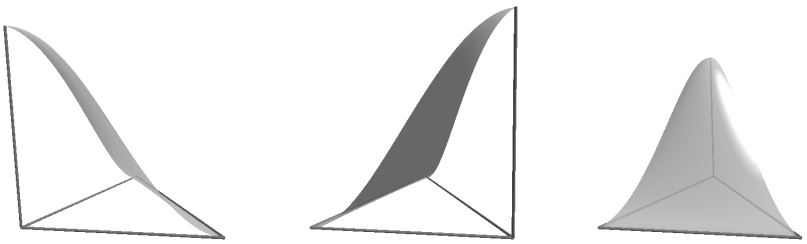

Figure 6: Weight functions f, g, h given by (7). Derivatives on the non-interpolated boundary are zero.

Bernstein polynomials are

$$B^n_{ijk}(u,v,w) = \frac{n!}{i!j!k!}u^i v^j w^k, \quad i+j+k=n. \tag{5}$$

and their known property is $(\forall \mathbf{u})\big(\sum_{i+j+k=n} B^n_{ijk}(\mathbf{u}) = 1\big)$. The following blending functions

$$\begin{aligned} f &= B^4_{400} + B^4_{310} + B^4_{301} + B^4_{211} + \frac{1}{2}(B^4_{220} + B^4_{202}) \\ g &= B^4_{040} + B^4_{031} + B^4_{130} + B^4_{121} + \frac{1}{2}(B^4_{022} + B^4_{220}) \\ h &= B^4_{004} + B^4_{103} + B^4_{013} + B^4_{112} + \frac{1}{2}(B^4_{202} + B^4_{022}) \end{aligned} \tag{6}$$

or expanded

$$\begin{aligned} f(\mathbf{u}) &= u^4 + 4u^3v + 4u^3w + 12u^2vw + 3u^2v^2 + 3u^2w^2 \\ g(\mathbf{u}) &= v^4 + 4v^3w + 4uv^3 + 12uv^2w + 3v^2w^2 + 3u^2v^2 \\ h(\mathbf{u}) &= w^4 + 4uw^3 + 4vw^3 + 12uvw^2 + 3u^2w^2 + 3v^2w^2 \end{aligned} \tag{7}$$

satisfy all (I), (II) and (III), as one can easily verify. They are shown in Fig. 6.

We have explained construction of triangular Coons patch for arbitrary given boundary curves and their tangent plane fields. In our case the boundary curves are defined by us from three vertices of a triangle and their normals. If we use the same curve construction operator $\mathbf{C}$ for both boundary curves and the 'ribs' of the three auxiliary patches, each of these patches will interpolate besides two boudaries and their tangent plane also the third one (but not its tangent plane!). Welcomed surprise is then, that the first introduced blending functions u, v and w are sufficient for smooth interpolation in this case. It is easy to prove it.

3.2 Boundary Curves Construction

Construction of boundary curves for Coons patches from the mesh vertices and normals is the same problem as defining $\mathbf{C}$ operator for patch ribs. All we know about the desired curve over an edge are its endpoints $\mathbf{P}_0$, $\mathbf{P}_1$ – the edge

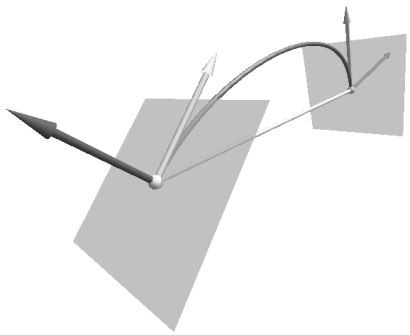

Figure 7: Calculation of tangent vectors from normals as an orthogonal projection of the edge vector into the tangent plane given by the normal.

endpoints and normals $\mathbf{n}_0$, $\mathbf{n}_1$ in the endpoints. The first idea is to use a cubic Ferguson curve given by two endpoints and the tangent vectors in these endpoints as

$$\mathbf{C}(t) = \mathbf{P}_0 H_0^3(t) + \mathbf{P}_1 H_1^3(t) + \mathbf{t}_0 H_2^3(t) + \mathbf{t}_1 H_3^3(t) \tag{8}$$

where $H_i^3(t)$ are cubic Hermit polynomials

$$\begin{aligned} H_0^3(t) &= 2t^3 - 3t^2 + 1 \\ H_1^3(t) &= -2t^3 + 3t^2 \\ H_2^3(t) &= t^3 - 2t^2 + t \\ H_3^3(t) &= t^3 - t^2. \end{aligned} \tag{9}$$

The tangent vectors $\mathbf{t}_0$, $\mathbf{t}_1$ of the boundary curves can be derived from the normals $\mathbf{n}_0$, $\mathbf{n}_1$ as follows. The normals determine a tangent plane of the surface in the edge endpoints. Tangent vectors $\mathbf{t}_0$ and $\mathbf{t}_1$ must lie in these tangent planes and we define their directions as an orthogonal projection of the edge into the respective tangent plane as shown in Figure 7. The directions are given by

$$\begin{aligned} \mathbf{t}_0 &= \big(\mathbf{n}_0 \times (\mathbf{P}_1 - \mathbf{P}_0)\big) \times \mathbf{n}_0 \\ \mathbf{t}_1 &= \big(\mathbf{n}_1 \times (\mathbf{P}_1 - \mathbf{P}_0)\big) \times \mathbf{n}_1. \end{aligned} \tag{10}$$

Length of the tangent vectors is basically set to the length of the edge. An additional "tension" parameter η may be introduced and the tangent vector length defined as the edge length divided by η. Thus, the constructed curve will approach to the edge as η raises and hence the surface will approach to the faceted mesh. $\eta = 1$ does not affect the curve and when η is dropping to zero, the curve is more and more attracted to the tangents of the endpoints. The tension parameter η needs to be set according to the surface character.

The surface interpolation with Ferguson boundary cubics, however, yields "too flat" interiors of the patches while the curvature is concentrated near the patch boundaries. It is concluded in [7], that the flatness of the patches is caused by the flatness of the Ferguson boundary curves, i.e. the curvature is

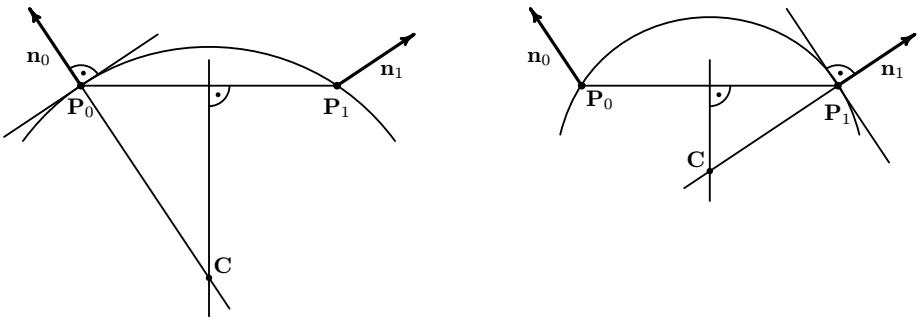

Figure 8: Construction of circular arcs interpolating $(\mathbf{P}_0, \mathbf{P}_1, \mathbf{n}_0)$ and $(\mathbf{P}_0, \mathbf{P}_1, \mathbf{n}_1)$ respectively.

concentrated around the endpoints, while the middle part of the curve is more flat. This flatness is then propagated to the patch interior. A way to obtain a better quality surface is to construct the boundary curves with a more uniform curvature distribution. We propose such curve construction.

The most uniform curvature distribution – constant, has a circular arc. But for two given points $\mathbf{P}_0$, $\mathbf{P}_1$ and their normals $\mathbf{n}_0$, $\mathbf{n}_1$ generally does not exist a circular arc from $\mathbf{P}_0$ to $\mathbf{P}_1$ with those given normals. Two circular arcs $\mathbf{A}_0(t)$ and $\mathbf{A}_1(t)$ can be constructed, however, interpolating $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{n}_0$ and $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{n}_1$ respectivelly. And as one can easily verify, a linear blend of these two arcs

$$\mathbf{C}(t) = (1-t)\mathbf{A}_0(t) + t\mathbf{A}_1(t) \tag{11}$$

interpolates all $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{n}_0$, $\mathbf{n}_1$. Centers of the arcs are found as an intersection of the line given by an endpoint with its desired normal and a plane perpendicular to the edge P_0P_1 crossing the edge midpoint (see Figure 8).

Curvature of the above circular arcs blend is more uniform, as shown in Figure 9 and hence it yields a more pleasing surface. Here we take advantage of Coons interpolation allowing arbitrary boundary curves, since the above curve is not polynomial. Even better curvature distribution and surface fairness is

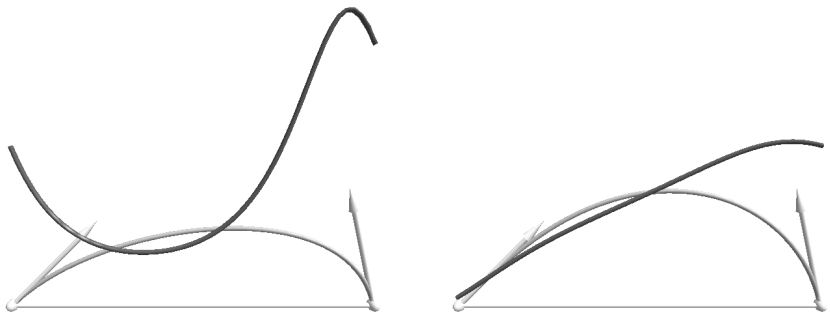

Figure 9: Curvature comparison of Ferguson cubic (left) and a blend of two circular arcs (right). The dark curve reflects curvature values along the ligth curve as a variable height.

achieved using a cubic blend

$$\begin{aligned} \mathbf{C}(t) &= f(t)\mathbf{A}_0(t) + (1 - f(t))\mathbf{A}_1(t) \\ f(t) &= 2t^3 - 3t^2 + 1 \end{aligned} \tag{12}$$

or a rational blend for

$$f(t) = \frac{t^2 - 2t + 1}{2t^2 - 2t + 1}. \tag{13}$$

The last issue, omitted so far, is the normal field construction. We define a similar operator $\mathbf{C}_n(t)$ which returns a normal to the curve $\mathbf{C}(t)$ for all $t \in \langle 0, 1\rangle$ and it interpolates the end-normals, i.e. $\mathbf{C}_n(0) = \mathbf{n}_0$ and $\mathbf{C}_n(1) = \mathbf{n}_1$. A simple interpolation of $\mathbf{n}_0$ and $\mathbf{n}_1$ is not guaranteed to be perpendicular to $\mathbf{C}'(t)$ for all t. We use a little 'trick'. 'Binormals'[3] $\mathbf{b}_0$, $\mathbf{b}_1$ in the endpoints are calculated as $\mathbf{b}_0 = \mathbf{n}_0 \times \mathbf{C}'(0)$ and $\mathbf{b}_1 = \mathbf{n}_1 \times \mathbf{C}'(1)$. To get a normal $\mathbf{C}_n(t)$ for a particular t the $\mathbf{b}_0$, $\mathbf{b}_1$ are linearly interpolated $\mathbf{b}_t = (1 - t)\mathbf{b}_0 + t\mathbf{b}_1$ and the normal is calculated then as $\mathbf{C}_n(t) = \mathbf{C}'(t) \times \mathbf{b}_t$. This normal is clearly perpendicular to the curve tangent and it interpolates $\mathbf{n}_0$, $\mathbf{n}_1$ as well due to the 'binormal' interpolation. Better results are achieved by quadratic interpolation of the 'binormal' adding one more 'binormal' sample in the curve midpoint derived from the edge midpoint normal.

4 Conclusion and Results

We have shown a smooth surface reconstruction from a triangular mesh. A simple implementation has been developed in C++ using OpenGL to test and experiment with desribed methods. Images in the paper come from our implementation.

It is quite a complex problem to achieve a really "fair" surface. Although the reconstructed surface is tangent-plane smooth, using Ferguson boundary curves the patch interiors are relatively flat and the curvature is concentrated near the patch boundaries making composition of the surfaces from individual patches visible. Even a decrease of the introduced tension parameter η does not solve the "flatness problem". In that case the flat regions appear around the mesh vertices. Better situation is with the second type of boundary curves, the blend of two circular arcs. See Figure 10 for comparison. The really pleasing surface usually needs some fairing process as used in [6], [2], typically an iterative one. We avoid of such process due to its global nature, since we expect really large models.

Another way is *surface approximation*. A more pleasing surface free of undesired undulations can be obtained by relaxing the requirement of passing exactly through the given points to minimization of a chosen error norm. Typically one patch is fitted to a whole region of triangles then.

The smooth surface reconstruction needs to be extended yet to make the representation of sharp edges and vertices possible, using multiple normals.

[3] They are not the real binormals, in fact, since the $\mathbf{n}_0$, $\mathbf{n}_1$ are generally not the main normals of the curve.

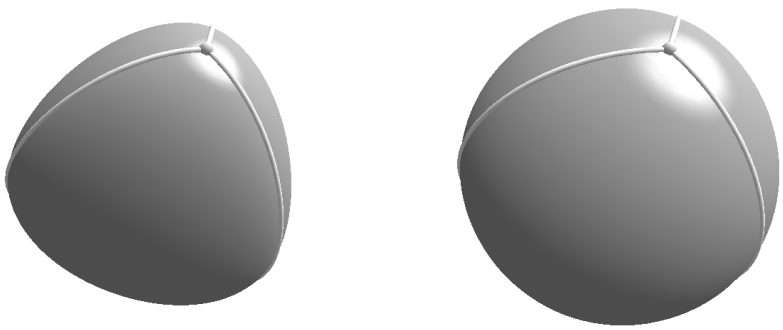

Figure 10: Comparison of reconstruction from a tetrahedron using cubic Ferguson boundary curves (left) vs. circular arcs blending curves (right). Surface on the right is closer to a sphere.

Figure 11: Example of a larger model – a cow with 5804 faces. The problems with interpolation mentioned in section 4 appears here – the surface approximates the original mesh too exactly.

References

[1] Chandrajit L. Bajaj and Susan Evans. Splines and geometric modeling. *CRC Handbook of Discrete and Computational Geometry*, 1997.

[2] Matthias Eck and Hugues Hoppe. Automatic reconstruction of B-spline surfaces of arbitrary topological type. *Computer Graphics*, 30(Annual Conference Series):325–334, 1996.

[3] Gerald Farin. *Curves and Surfaces for CAGD: A Practical Guide*. Academic Press, 4^{th} edition, 1996.

[4] Hugues Hoppe. Progressive meshes. *Computer Graphics*, 30(Annual Conference Series):99–108, 1996.

[5] Hugues Hoppe, Tony DeRose, Tom Duchamp, Mark Halstead, Hubert Jin, John McDonald, Jean Schweitzer, and Werner Stuetzle. Piecewise smooth surface reconstruction. *Computer Graphics*, 28(Annual Conference Series):295–302, 1994.

[6] Venkat Krishnamurthy and Marc Levoy. Fitting smooth surfaces to dense polygon meshes. *Computer Graphics*, 30(Annual Conference Series):313–324, 1996.

[7] S. Mann, C. Loop, M. Lounsbery, D. Meyers, J. Painter, T. DeRose, and K. Sloan. A survey of parametric scattered data fitting using triangular interpolants, 1992.

[8] G. M. Nielson. A transfinite, visually continuous, triangular interpolant. *Geometric modeling: Algorithms and new trends*, pages 235–246, 1987.

[9] In Kyu Park, Sang Uk Lee, and Il Dong Yun. Constructing NURBS surface model from scattered and unorganized range data. *Proceedings of the Second International Conference on 3-D Imaging and Modeling*, 1998.

[10] Hong Qin and Demetri Terzopoulos. Triangular NURBS and their dynamic generalizations. *Computer Aided Geometric Design*, 14(4):325–347, 1997.

[11] Inge Soderkvist. Introductory overview of surface reconstruction methods, 1999.