22es Journées de l'Association Francophone d'Informatique Graphique, Arles, 2009

Etat de l'art des méthodes d'interpolation locale de maillages triangulaires par des facettes non-planes C^0-continues

Maria Boschiroli[1,3,†] Christoph Fünfzig[2] Lucia Romani[3] Gudrun Albrecht[1]

[1]Univ Lille Nord de France, UVHC, LAMAV-CGAO, FR no. 2956, F-59313 Valenciennes, France
[2]LE2I (UMR CNRS 5158), Université de Bourgogne, 9 Avenue Alain Savary, F-21078 Dijon, France
[3]Dip. di Matematica e Applicazioni, Università di Milano-Bicocca, Via Cozzi 53, 20125 Milano, Italia

Abstract

Depuis la fin des années 1980, le problème de l'interpolation paramétrique de points et de normales sur des maillages triangulaires a été étudié afin de modéliser des surfaces de type topologique arbitraire. En particulier, l'interpolation locale par des facettes paramétriques triangulaires et non-planes est un sujet d'actualité en raison de la difficulté pour obtenir des formes lisses et sans ondulations. La localisation des données est extrêmement importante dans certaines applications telles que l'informatique graphique utilisée pour le jeux vidéo et pour le rendu en temps réel. Des schémas produisant des surfaces interpolantes continues basés sur des facettes triangulaires non-planes sont apparus récemment afin de répondre aux conditions spécifiques d'environnements de hardware limités en ressources. Ces méthodes améliorent la qualité visuelle de la surface C^0 en utilisant uniquement des informations locales. Dans cet article, nous présentons un survol des méthodes de ce type récemment développées en discutant leur construction originale ainsi qu'en donnant leur formulation Bézier. Nous comparons les différentes surfaces construites par les méthodes considérées pour des maillages grossiers qui caractérisent en fait leurs utilisations concrètes.

Since the end of the 1980's, parametric interpolation of points and normals over triangular meshes has been investigated to answer the problem of modeling two-manifold surfaces of arbitrary topological type. In particular, local parametric curved shape interpolation is a problem of ongoing research, since it is difficult to obtain surfaces with fair and smooth shapes. Locality of data is extremely important in some applications, for instance in computer graphics used for gaming and real time rendering. Continuous interpolant curved shape schemes *recently emerged to address specific requirements of resource-limited hardware environments and to offer smooth surfaces by visually enhancing the resulting C^0 surface using only local information. In this paper we present a survey on the recently developed continuous surface schemes discussing their original construction and then comparing them from a geometric point of view in Bézier patch form. We compare the different surfaces constructed by the schemes for low triangle count meshes that actually correspond to their real-world utilization.*

1. Introduction

The easiest way of modeling free-form surfaces is to use tensor-product Bézier, BSpline or NURBS patches. For this reason a long time ago, they became a "de facto" standard in the CAD/CAM industry. But, unfortunately, tensor-product patches are able to model only a restricted type of surfaces, those which are topologically equivalent to a square. However, two-manifold surfaces with arbitrary topological type are very common in everyday life, and many different directions have been pursued to model them with the computer.

One of them consists of building a patchwork of smoothly joined parametric patches with the same topology as the control polygons. The topological information is usually spec-

† La doctorante Maria Boschiroli bénéficie du soutien de l'Université Franco Italienne sous la forme d'une bourse d'accompagnement à sa thèse dans le cadre du programme da Vinci.

ified as adjacency information relating the data points (vertices), edges, and faces. *Triangular meshes*, i.e., meshes in which the faces are triangular and any number of faces may join at a vertex, are sufficiently general to represent surfaces of arbitrary genus.

Especially in geometric modeling, the problem of passing a surface through a set of data points is a very useful and intuitive feature. In the case of triangular meshes, the given data are the triangular mesh vertices and their respective normals. The first approach is to construct the triangular mesh simply connecting the points in triangular plane faces. Obviously, with coarse meshes this leads to only continuous surfaces with low quality shape (see for example Figure 3(a)) and acceptable visually smooth shapes can be obtained only by deeply refining the triangular mesh.

Instead, a *parametric curved shape interpolant* scheme constructs a vector-valued surface, $\mathbf{s}(u,v) = (x(u,v), y(u,v), z(u,v))$ that interpolates the given points and normals and, unlike a functional method, is able to represent arbitrary topological shapes.

Besides distinguishing between parametric and functional methods, we can also classify surface fitting schemes by the locality of data used in constructing a part of the surface. A *local* scheme only considers those points near the portion of the surface it is creating, thus if a single input vertex is moved, the interpolating surface changes only in the neighborhood of the vertex. This feature is particularly attracting in most applications; however local parametric interpolation is an open problem because usually the schemes don't provide surfaces of acceptable quality or with fair and smooth shape.

Computer graphics used for gaming and realtime rendering is about shading and animating with triangle meshes. A large body of work has been devoted to creating an increasing realism of rendered surfaces. Shading techniques like phong shading, normal mapping and reflection mapping are commonly used to present cineastically looking surfaces. For animation [Col05], models are applied with a suitable skeleton structure during rigging [BP07], and then all triangle vertices can be moved according to this structure. Especially in computer games, its highly elaborate art pipeline builds upon the triangle mesh, which usually does not have stored neighborhood information. Several techniques have been leveraged for processing on programmable graphics hardware recently [Kau04, SKUP*09].

Continuous (C^0) interpolant curved shape surface schemes emerged to address requirements specific to the resource-limited hardware environment and to offer smooth surfaces by visually enhancing the resulting C^0 surface by means of local information only.

More precisely, the smallest amount of information about neighboring triangles has to be used in constructing the patch. As a consequence, continuous surface schemes point their attention towards visual smoothness. Namely, realizing that in most situations exact geometric smoothness and continuity are not critical as long as the surface appears to be smooth as a result of the shading technique. For example, interpolating per-vertex normal vectors for shading computations achieves visual smoothness, still processing triangles independently and avoiding knowledge of neighbors. The interest in continuous surface patches comes primarily from saving bus bandwidth for transfers to the graphics hardware.

The aim of this paper is to provide a unifying survey of the recently developed local parametric triangular curved shape C^0 schemes we are aware of.

The paper is organized as follows. In section 2 we present the existing continuous surface schemes both discussing their original construction and presenting a reformulation of every scheme in triangular Bézier patch form. This allows us to discuss their geometric interpretations and compare them. In section 3 we then compare the surface quality of all surveyed schemes analyzing their behavior on arbitrary triangle meshes. Finally, in section 4 we discuss our conclusions.

2. Continuous Surface Schemes

Triangular Bézier patches have been around since the birth of Computer Aided Geometric Design. De Casteljau investigated them as extensions of Bézier curves to surfaces. They are a simple geometric primitive that can be used to interpolate scattered data while offering interactive manipulation by its control points and local control of a surface.

The key idea behind the C^0 curved shape schemes that we are going to present is that each original flat triangle of the input mesh can be replaced by a curved shape, namely a cubic or quadratic triangular Bézier patch. The patch control net is constructed only by means of the point and normal information at the vertices of the input mesh. According to requirements explained above, no additional data beyond the positions and normals are used. Let us denote the three triangle vertices by $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{p}_2$, the respective normals with $\mathbf{n}_0$, $\mathbf{n}_1$, $\mathbf{n}_2$, and the edges with $\mathbf{d}_1 = \mathbf{p}_1 - \mathbf{p}_0$, $\mathbf{d}_2 = \mathbf{p}_2 - \mathbf{p}_1$, $\mathbf{d}_3 = \mathbf{p}_2 - \mathbf{p}_0$, as shown in Figure 1. Additionally, we will refer to the tangent plane in $\mathbf{p}_i$ (defined by $\mathbf{n}_i$) by τ_i, $i = 1,2,3$.

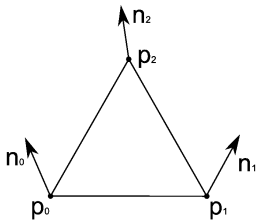


Figure 1: *Notation for the vertices and respective normals of the input flat triangles.*

For comparison purposes, although each scheme uses its own formulation, we decided to describe all the schemes using the same notation in terms of triangular Bézier patches and to analyze their geometrical interpretation.

Using a triangular network of control points

$$\mathbf{b}_{ijk}: \quad i+j+k=n,\ i,j,k \geqslant 0$$

and degree-n bivariate Bernstein polynomials

$$B^n_{ijk}(u,v,w) = \frac{n!}{i!\,j!\,k!} u^i v^j w^k, \quad u+v+w=1,$$

a degree-n triangular Bézier patch is defined by

$$\mathbf{s}(u,v,w) = \sum_{i+j+k=n} \mathbf{b}_{ijk} B^n_{ijk}(u,v,w).$$

It maps a triangular domain $D \in \mathbb{R}^2$ to an affine space, typically $\mathbb{R}^3$, where u, v and w are the barycentric coordinates of a domain point relative to D. For our purposes the main features of a triangular Bézier patch that turn out to be more interesting are the corner point interpolation, the convex hull property and the fact that the images of the three edges of the domain triangle are Bézier curves defined by the boundary control points of the patch (see [Far02] for details).

Together with the schemes that we are going to present, we would also like to cite [VMT98, VOW97a] as interesting related to the interpolation problem we are considering. However, we decided not to include them in our discussion because they do not fit into the class of analytically representable curved patches.

2.1. PN Triangles

Curved PN triangles by Vlachos et al. [VPBM01], in a certain sense, are the pioneers in the study of parametric curved patches for C^0 interpolation of triangle meshes. The geometry of a PN triangle is defined by a cubic triangular Bézier patch and the construction of its control points is based on projections over the tangent planes in the vertices.

The scheme initially places the intermediate control points $\overline{\mathbf{b}}_{ijk}$ in the positions $(i\mathbf{p}_0 + j\mathbf{p}_1 + k\mathbf{p}_2)/3$, leaving the three corner points unchanged. Then, each $\mathbf{b}_{ijk}$ on the border is constructed projecting the respective intermediate control point $\overline{\mathbf{b}}_{ijk}$ into the plane defined by the nearest corner point and the normal in that corner. For example, Figure 2 shows the construction of $\mathbf{b}_{210}$.

Finally, the center control point $\mathbf{b}_{111}$ is constructed moving the corresponding $\overline{\mathbf{b}}_{111}$ halfway in the direction $\mathbf{m} - \overline{\mathbf{b}}_{111}$ where $\mathbf{m}$ is the average of the six control points just computed on the border.

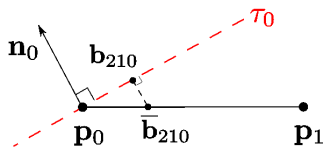


Figure 2: *Construction of* $\mathbf{b}_{210}$ *in PN triangle's scheme: projection of* $\overline{\mathbf{b}}_{210} = (2\mathbf{p}_0 + \mathbf{p}_1)/3$ *into the tangent plane at* $\mathbf{p}_0$.

In formulas:

$$\mathbf{b}_{300} = \mathbf{p}_0, \quad \mathbf{b}_{030} = \mathbf{p}_1, \quad \mathbf{b}_{003} = \mathbf{p}_2,$$
$$w_{ij} = (\mathbf{p}_j - \mathbf{p}_i) \cdot \mathbf{n}_i,$$
$$\mathbf{b}_{210} = \frac{1}{3}(2\mathbf{p}_0 + \mathbf{p}_1 - w_{01}\mathbf{n}_0), \quad \mathbf{b}_{120} = \frac{1}{3}(2\mathbf{p}_1 + \mathbf{p}_0 - w_{10}\mathbf{n}_1),$$
$$\mathbf{b}_{021} = \frac{1}{3}(2\mathbf{p}_1 + \mathbf{p}_2 - w_{12}\mathbf{n}_1), \quad \mathbf{b}_{012} = \frac{1}{3}(2\mathbf{p}_2 + \mathbf{p}_1 - w_{21}\mathbf{n}_2),$$
$$\mathbf{b}_{102} = \frac{1}{3}(2\mathbf{p}_2 + \mathbf{p}_0 - w_{20}\mathbf{n}_2), \quad \mathbf{b}_{201} = \frac{1}{3}(2\mathbf{p}_0 + \mathbf{p}_2 - w_{02}\mathbf{n}_0),$$
$$\mathbf{m} = \frac{1}{6}(\mathbf{b}_{210} + \mathbf{b}_{120} + \mathbf{b}_{021} + \mathbf{b}_{012} + \mathbf{b}_{102} + \mathbf{b}_{201}),$$
$$\overline{\mathbf{b}}_{111} = \frac{1}{3}(\mathbf{p}_0 + \mathbf{p}_1 + \mathbf{p}_2),$$
$$\mathbf{b}_{111} = \mathbf{m} + \frac{1}{2}(\mathbf{m} - \overline{\mathbf{b}}_{111}).$$

As shown in [VPBM01], this particular choice for the central control point $\mathbf{b}_{111}$ is based on exact reproduction of quadratic polynomials and has the merit of keeping the curved patch provably close to the flat triangle, preserving the shape and avoiding interference with other curved triangles.

In Figure 3(b) an example of Curved PN triangles on a triangular mesh is illustrated.

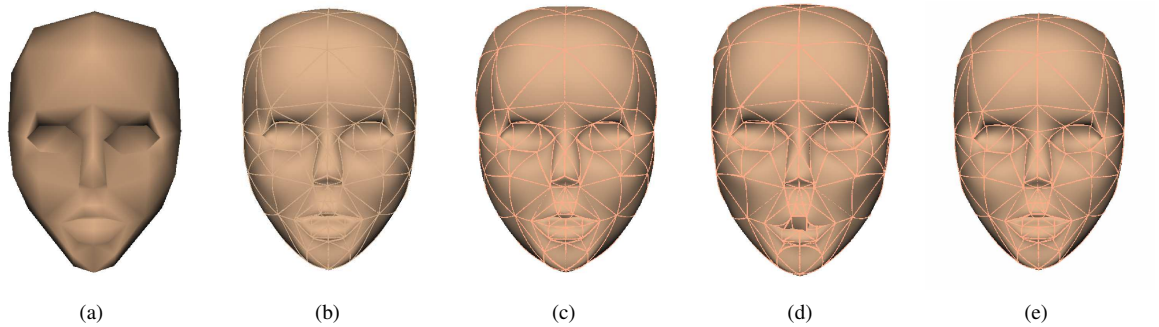
2.2. Phong tessellation

Phong tessellation [BA08] is a recent work based on the idea that a real-time mesh refinement operator should be as efficient and simple as Phong normal interpolation. The main concept behind the scheme is that the tangent plane in each vertex point defines the appropriate local surface geometry and thus has to be used in the construction of the middle-edge control points.

The patch is evaluated at the barycentric coordinates (u,v,w) in three simple steps:

- compute the point $\overline{\mathbf{p}}(u,v,w)$ as linear combination between the three triangle vertices:

$$\overline{\mathbf{p}}(u,v,w) = u\mathbf{p}_0 + v\mathbf{p}_1 + w\mathbf{p}_2;$$



(a) (b) (c) (d) (e)

Figure 3: *A triangular head model rendered with different schemes. (a) Gourand shaded model, (b) PN triangles, (c) Phong tessellation, (d) Nagata triangles and (e) NLSA triangles. Patch borders are also plotted in (b)-(e).*

- project $\overline{\mathbf{p}}(u,v,w)$ in the tangent planes defined by the input vertices and normals obtaining the three points:

$$\pi_0(\overline{\mathbf{p}}(u,v,w)) = \overline{\mathbf{p}}(u,v,w) - [(\overline{\mathbf{p}}(u,v,w) - \mathbf{p}_0)\cdot\mathbf{n}_0]\cdot\mathbf{n}_0,$$
$$\pi_1(\overline{\mathbf{p}}(u,v,w)) = \overline{\mathbf{p}}(u,v,w) - [(\overline{\mathbf{p}}(u,v,w) - \mathbf{p}_1)\cdot\mathbf{n}_1]\cdot\mathbf{n}_1,$$
$$\pi_2(\overline{\mathbf{p}}(u,v,w)) = \overline{\mathbf{p}}(u,v,w) - [(\overline{\mathbf{p}}(u,v,w) - \mathbf{p}_2)\cdot\mathbf{n}_2]\cdot\mathbf{n}_2;$$

- compute the final evaluation point as the linear interpolation of these three projections:

$$\mathbf{p}^*(u,v,w) = u\pi_0(\overline{\mathbf{p}}(u,v,w)) + v\pi_1(\overline{\mathbf{p}}(u,v,w)) + w\pi_2(\overline{\mathbf{p}}(u,v,w)). \tag{1}$$

Additionally, a shape factor α can be used to interpolate between linear (flat) and Phong tessellation, controlling the distance from the flat triangle. Hence, the final surface can be written as:

$$\mathbf{s}_\alpha(u,v,w) = (1-\alpha)\overline{\mathbf{p}}(u,v,w) + \alpha\mathbf{p}^*(u,v,w).$$

In [BA08], $\alpha = 3/4$ is proposed because this value experimentally provides convincing results in most of the situations.

Writing out the definition (1) of Phong tessellation without the shape factor, it can be easily seen that $\mathbf{p}^*(u,v)$ is a quadratic patch:

$$\begin{aligned}\mathbf{p}^*(u,v,w) = {} & u^2\mathbf{p}_0 + v^2\mathbf{p}_1 + w^2\mathbf{p}_2 + \\ & + uv[\pi_0(\mathbf{p}_1) + \pi_1(\mathbf{p}_0)] + \\ & + vw[\pi_1(\mathbf{p}_2) + \pi_2(\mathbf{p}_1)] + \\ & + wu[\pi_2(\mathbf{p}_0) + \pi_0(\mathbf{p}_2)].\end{aligned} \tag{2}$$

Eq. (2) allows us to get a formulation of Phong tessellation patch $P^*(u,v)$ in quadratic Bézier triangle form with control points:

$$\mathbf{b}_{200} = \mathbf{p}_0, \quad \mathbf{b}_{110} = \frac{1}{2}[\pi_0(\mathbf{p}_1) + \pi_1(\mathbf{p}_0)],$$
$$\mathbf{b}_{020} = \mathbf{p}_1, \quad \mathbf{b}_{011} = \frac{1}{2}[\pi_1(\mathbf{p}_2) + \pi_2(\mathbf{p}_1)],$$
$$\mathbf{b}_{002} = \mathbf{p}_2, \quad \mathbf{b}_{101} = \frac{1}{2}[\pi_2(\mathbf{p}_0) + \pi_0(\mathbf{p}_2)].$$

In this form Phong tessellation has a simple geometric interpretation: the three control points $\mathbf{b}_{110}$, $\mathbf{b}_{011}$ and $\mathbf{b}_{101}$ are the average of the projections of the edge corners into the tangent plane in the respective opposite edge corner (see Figure 4).

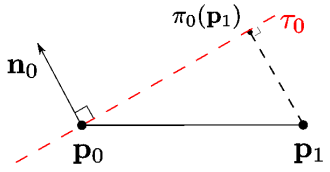


Figure 4: *Projection of* $\mathbf{p}_1$ *in the tangent plane in* $\mathbf{p}_0$ *defined by* $\mathbf{n}_0$.

In Figure 3(c) the Phong tessellation of the triangular mesh representing a head is shown.

2.3. Nagata Triangles

The central idea in the scheme proposed by T. Nagata in [Nag05] is quadratic interpolation of a curved segment from the position and normal vectors at the end-points, with the aid of generalized inverses. More precisely, this scheme first replaces each edge of the planar triangle with a curve orthogonal to the normals given at the end-points, then fills in the interior of the patch with a parametric quadratic surface reproducing the modified boundaries.

For instance, let $\mathbf{x}_1(t)$ ($t \in [0,1]$) be the quadratic polynomial curve between $\mathbf{p}_0$ and $\mathbf{p}_1$ written in monomial form

$$\mathbf{x}_1(t) = \mathbf{a}_1 + \mathbf{b}_1 t + \mathbf{c}_1 t^2. \tag{3}$$

End-points interpolation imposes that $\mathbf{c} = \mathbf{p}_0$ and $\mathbf{a}_1 + \mathbf{b}_1 + \mathbf{c}_1 = \mathbf{p}_1$. Therefore, we can write the curve as

$$\mathbf{x}_1(t) = \mathbf{p}_0 + (\mathbf{d}_1 - \mathbf{c}_1)t + \mathbf{c}_1 t^2$$

where $\mathbf{c}_1$ can be interpreted as an unknown that makes the segment curved. Orthogonality condition between the curve and the normal $\mathbf{n}_0$ in $t=0$ and $\mathbf{n}_1$ in $t=1$ leads to the system of equations

$$\begin{pmatrix} \mathbf{n}_0^T \\ \mathbf{n}_1^T \end{pmatrix} \mathbf{c}_1 = \begin{pmatrix} \mathbf{n}_0^T \mathbf{d}_1 \\ -\mathbf{n}_1^T \mathbf{d}_1 \end{pmatrix}. \tag{4}$$

Nagata suggests to find a solution for eq. (4) minimizing $\mathbf{c}_1$ through the use of generalized inverse (or pseudo inverse). This can be attained by

$$\mathbf{c}_1 = \begin{pmatrix} \mathbf{n}_0^T \\ \mathbf{n}_1^T \end{pmatrix}^+ \begin{pmatrix} \mathbf{n}_0^T \mathbf{d}_1 \\ -\mathbf{n}_1^T \mathbf{d}_1 \end{pmatrix} \tag{5}$$

where $^+$ denotes the generalized inverse. Its explicit expression is obtained by the analytical formula

$$\mathbf{A}^+ = \lim_{\alpha \to 0^+} \left(\mathbf{A}^* \mathbf{A} + \alpha \mathbf{E}\right)^{-1} \mathbf{A}^*, \tag{6}$$

where $\mathbf{A}$, respectively $\mathbf{A}^*$, are an arbitrary matrix, respectively its transposed conjugate, and $\mathbf{E}$ is the identity matrix of consistent dimension. Direct substitution of (6) in (5) gives:

$$\mathbf{c}_1 = \begin{cases} \frac{\Delta d}{1-\Delta c}\nu + \frac{d}{\Delta c}\Delta\nu, & c \neq \pm 1; \\ \mathbf{0}, & c = \pm 1; \end{cases} \tag{7}$$

where ν and $\Delta\nu$ are the average and the deviation of the unit normals

$$\nu = \frac{\mathbf{n}_0 + \mathbf{n}_1}{2}, \quad \Delta\nu = \frac{\mathbf{n}_0 - \mathbf{n}_1}{2},$$

d and Δd are their inner products with $\mathbf{d}_1$

$$d = \mathbf{d}_1^T \nu, \quad \Delta d = \mathbf{d}_1^T \Delta\nu,$$

and

$$\Delta c = \mathbf{n}_0^T \Delta\nu, \quad c = \mathbf{n}_0^T \mathbf{n}_1 = 1 - 2\Delta c.$$

The same procedure can be applied to the edges $\mathbf{d}_2$ and $\mathbf{d}_3$ resulting in the two curves

$$\mathbf{x}_2(t) = \mathbf{p}_1 + (\mathbf{d}_2 - \mathbf{c}_2)t + \mathbf{c}_2 t^2$$

and

$$\mathbf{x}_3(t) = \mathbf{p}_0 + (\mathbf{d}_3 - \mathbf{c}_3)t + \mathbf{c}_3 t^2,$$

defined by the coefficients $\mathbf{c}_2$ and $\mathbf{c}_3$, respectively.

Now, let $\mathbf{s}(t,s)$ $(0 \leqslant s \leqslant t \leqslant 1)$ be a quadratic triangular patch in monomial form

$$\mathbf{s}(t,s) = \mathbf{c}_{00} + \mathbf{c}_{10}t + \mathbf{c}_{01}s + \mathbf{c}_{11}ts + \mathbf{c}_{20}t^2 + \mathbf{c}_{02}s^2.$$

By imposing

$$\begin{aligned} \mathbf{s}(t,0) &= x_1(t), \\ \mathbf{s}(1,s) &= x_2(s), \\ \mathbf{s}(t,t) &= x_3(t), \end{aligned}$$

the surface coefficients can be easily computed as

$$\begin{aligned} \mathbf{c}_{00} &= \mathbf{p}_0, \\ \mathbf{c}_{10} &= \mathbf{d}_1 - \mathbf{c}_1, \\ \mathbf{c}_{01} &= \mathbf{d}_2 + \mathbf{c}_1 - \mathbf{c}_3, \\ \mathbf{c}_{11} &= \mathbf{c}_3 - \mathbf{c}_1 - \mathbf{c}_2, \\ \mathbf{c}_{20} &= \mathbf{c}_1, \\ \mathbf{c}_{02} &= \mathbf{c}_2. \end{aligned}$$

Once the boundary is interpolated the parametric representation of the patch is readily obtained.

Although the monomial form allows an easy and fast coefficients computation, a triangular Bézier formulation of the patch can be obtained by means of a change of parametrization and has a simple geometric interpretation. The three control points $\mathbf{b}_{110}$, $\mathbf{b}_{101}$ and $\mathbf{b}_{011}$ are defined moving the average of the vertices on an edge halfway the direction given by the computed coefficient related to that edge:

$$\begin{aligned} \mathbf{b}_{110} &= \frac{1}{2}(\mathbf{p}_0 + \mathbf{p}_1) - \frac{1}{2}\mathbf{c}_1, \\ \mathbf{b}_{011} &= \frac{1}{2}(\mathbf{p}_1 + \mathbf{p}_2) - \frac{1}{2}\mathbf{c}_2, \\ \mathbf{b}_{101} &= \frac{1}{2}(\mathbf{p}_0 + \mathbf{p}_2) - \frac{1}{2}\mathbf{c}_3, \end{aligned}$$

where explicitly

$$\mathbf{c}_1 = \begin{cases} \frac{\mathbf{d}_1 \cdot \frac{\mathbf{n}_0 - \mathbf{n}_1}{2}}{1 - \mathbf{n}_0 \cdot \frac{\mathbf{n}_0 - \mathbf{n}_1}{2}} \frac{\mathbf{n}_0 + \mathbf{n}_1}{2} + \frac{\mathbf{d}_1 \cdot \frac{\mathbf{n}_0 + \mathbf{n}_1}{2}}{\mathbf{n}_0 \cdot \frac{\mathbf{n}_0 - \mathbf{n}_1}{2}} \frac{\mathbf{n}_0 - \mathbf{n}_1}{2}, & \mathbf{n}_0 \cdot \mathbf{n}_1 \neq \pm 1; \\ \mathbf{0}, & \mathbf{n}_0 \cdot \mathbf{n}_1 = \pm 1. \end{cases} \tag{8}$$

and analogous formulas hold for $\mathbf{c}_2$ and $\mathbf{c}_3$.

Equations (7) and (8) defining the coefficient $\mathbf{c}_1$ have a stability problem that might strongly engrave on the surface. In fact, there are two denominators that obviously should not become zero. This happens when $c = \mathbf{n}_0 \cdot \mathbf{n}_1 = \pm 1$ or equivalently when $\Delta c = 0$ or $\Delta c = 1$, and in these cases the curvature coefficient is set to be zero. This means that when the angle between the two normals in the vertices of one edge is 0° or 180°, the curvature coefficient cannot be calculated and it is set to zero, leading the surface to be linear on that edge.

Unfortunately, avoiding these two configurations is often not sufficient. In fact, when the angle between the two normals is not 0° but very small (or analogously when it is near 180°) we have the same stability problem because we divide by a number close to zero. This can lead to visible artifacts on the surface.

We corrected this problem by using a threshold ε in the coefficient definitions (7) and (8), where $\mathbf{c}_1$ is set to $\mathbf{0}$ if $\Delta c \leqslant \varepsilon$ or $1 - \Delta c \leqslant \varepsilon$ (or equivalent conditions on c).

In Figure 3(d) Nagata's surface of the head model is shown.

2.4. NLSA Triangles

The last scheme we describe presents the construction of a curvilinear mesh using quadratic curves with near least square acceleration (NLSA). This scheme was proposed by Barrera et al. in [BHB02]. It constructs a quadratic surface using quadratic curves which are derived using vertex normals and vertex points only, as Nagata's scheme above, but with different minimizations.

Consider the edge $\mathbf{d}_1$, and a quadratic end-points interpolating curve in monomial form as in (3). Their approach, after computation of the tangents $\mathbf{t}_0$ and $\mathbf{t}_1$ from the normals $\mathbf{n}_0$ and $\mathbf{n}_1$, first computes a curve $\mathbf{q}_1(t)$ such that its derivative in $\mathbf{p}_0$ is equal to the tangent $\mathbf{t}_0$ and the derivative in $\mathbf{p}_2$ is as close to the tangent $\mathbf{t}_1$ as the least square minimization allows. Then, they compute the curve $\mathbf{q}_2(t)$ with the derivative equal to the tangent in $\mathbf{p}_1$ and optimized at $\mathbf{p}_0$. Finally, by taking the average of $\mathbf{q}_1(t)$ and $\mathbf{q}_2(t)$, they get the near least square acceleration second degree curve $\mathbf{x}_1(t)$ which is close to optimal in both ends.

Defining $\mathbf{q}_1(t)$ as required implies that the conditions $\mathbf{q}_1'(0) = \alpha_1\beta_1\mathbf{t}_0$ and $\mathbf{q}_1'(1) = \beta_1\mathbf{t}_1$ have to be satisfied together with the end-points interpolation condition. This forces the coefficient $\mathbf{b}_1$ to be

$$\mathbf{b}_1 = \alpha_1\beta_1\mathbf{t}_0,$$

and the curve takes now the form

$$\mathbf{q}_1(t) = \mathbf{p}_0 + \mathbf{b}_1 t + (\mathbf{d}_1 - \mathbf{b}_1)t^2. \tag{9}$$

Hence, the definition of $\mathbf{q}_1(t)$ consists in the determination of the constants α_1 and β_1.

First, the difference between the derivative and the tangent in $\mathbf{p}_1$ is minimized. Thus, by imposing:

$$\frac{\partial}{\partial\alpha_1}\left\{\|\mathbf{q}_1'(1) - \beta_1\mathbf{t}_1\|^2\right\} = 0,$$

the following first condition is obtained:

$$2\mathbf{d}_1\cdot\mathbf{t}_0 - \alpha_1\beta_1\mathbf{t}_0\cdot\mathbf{t}_0 - \beta_1\mathbf{t}_0\cdot\mathbf{t}_1 = 0.$$

Next, a second condition is derived imposing that the integral of the acceleration is the least-square minimum in the interval $[0,1]$:

$$\frac{\partial}{\partial\beta_1}\int_0^1 \|\mathbf{q}_1''(t)\|^2 = 0,$$

leading to

$$\mathbf{d}_1\cdot\mathbf{t}_0 - \alpha_1\beta_1\mathbf{t}_0\cdot\mathbf{t}_0 = 0.$$

We therefore end up with a system of two equations in the unknowns α_1 and β_1 with solution

$$\alpha_1 = \frac{\mathbf{t}_0\cdot\mathbf{t}_1}{\mathbf{t}_0\cdot\mathbf{t}_0} \text{ and } \beta_1 = \frac{\mathbf{d}_1\cdot\mathbf{t}_0}{\mathbf{t}_0\cdot\mathbf{t}_1}.$$

The next step is to repeat the same procedure for the other tangent $\mathbf{t}_1$ at the other end of the curve, in order to obtain $\mathbf{q}_2(t)$. This requires to impose the conditions $\mathbf{q}_2'(0) = \overline{\beta}_1\mathbf{t}_0$ and $\mathbf{q}_2'(1) = \overline{\alpha}_1\overline{\beta}_1\mathbf{t}_1$, leading to

$$\mathbf{c}_2 = \overline{\alpha}_1\overline{\beta}_1\mathbf{t}_1 - \mathbf{d}_1,$$
$$\mathbf{b}_2 = 2\mathbf{d}_1 - \overline{\alpha}_1\overline{\beta}_1\mathbf{t}_1.$$

Finally, the quadratic curve becomes

$$\mathbf{q}_2(t) = (\overline{\alpha}_1\overline{\beta}_1\mathbf{t}_1 - \mathbf{d}_1)t^2 + (2\mathbf{d}_1 - \overline{\alpha}_1\overline{\beta}_1\mathbf{t}_1)t + \mathbf{p}_0, \tag{10}$$

where $\overline{\alpha}_1$ and $\overline{\beta}_1$ are defined by repeating the same minimization process as before on $q_2(t)$, thus yielding

$$\overline{\alpha}_1 = \frac{\mathbf{t}_0\cdot\mathbf{t}_1}{\mathbf{t}_1\cdot\mathbf{t}_1} \text{ and } \overline{\beta}_1 = \frac{\mathbf{d}_1\cdot\mathbf{t}_1}{\mathbf{t}_0\cdot\mathbf{t}_1}.$$

By taking the mean value of (9) and (10) we get the final symmetric curve with near least square acceleration:

$$\mathbf{x}_1(t) = \left(\frac{\overline{\alpha}_1\overline{\beta}_1\mathbf{t}_1 - \alpha_1\beta_1\mathbf{t}_0}{2}\right)t^2 + \left(\mathbf{d}_1 + \frac{\alpha_1\beta_1\mathbf{t}_0 - \overline{\alpha}_1\overline{\beta}_1\mathbf{t}_1}{2}\right)t + \mathbf{p}_0.$$

The same procedure can be repeated for the edges $\mathbf{d}_2$ and $\mathbf{d}_3$ to obtain the curves $\mathbf{x}_2(t)$ and $\mathbf{x}_3(t)$ defined respectively by the parameters α_2, β_2, $\overline{\alpha}_2$, $\overline{\beta}_2$ and α_3, β_3, $\overline{\alpha}_3$, $\overline{\beta}_3$.

Having the three border curves in monomial form, it is easy to compute their Bézier formulation. The three central control points together with the vertices are then used as control net for a quadratic Bézier triangle. In formulas:

$$\mathbf{b}_{110} = \frac{1}{2}(\mathbf{p}_0 + \mathbf{p}_1) + \frac{1}{4}\left(\alpha_1\beta_1\mathbf{t}_0 - \overline{\alpha}_1\overline{\beta}_1\mathbf{t}_1\right),$$
$$\mathbf{b}_{011} = \frac{1}{2}(\mathbf{p}_1 + \mathbf{p}_2) + \frac{1}{4}\left(\alpha_2\beta_2\mathbf{t}_1 - \overline{\alpha}_2\overline{\beta}_2\mathbf{t}_2\right),$$
$$\mathbf{b}_{101} = \frac{1}{2}(\mathbf{p}_0 + \mathbf{p}_2) + \frac{1}{4}\left(\alpha_3\beta_3\mathbf{t}_0 - \overline{\alpha}_3\overline{\beta}_3\mathbf{t}_2\right).$$

Note that $\alpha_1\beta_1\mathbf{t}_0$ is nothing else that the projection of $\mathbf{d}_1$ on $\mathbf{t}_0$. Similarly, $\overline{\alpha}_1\overline{\beta}_1\mathbf{t}_1$ is the projection of $\mathbf{d}_1$ on $\mathbf{t}_1$.

It means that NLSA triangles in Bézier formulation have a simple geometric interpretation. The central control point on one edge is defined moving the average of the two edge vertices in the direction given by the sum of the projections of the edge on the tangents of the two vertices.

In Figure 3(e) the triangle mesh of a head model is rendered by NLSA triangles.

3. Comparison

We implemented all the schemes as an Autodesk Maya® plug-in (*MPxHwShaderNode*). The Polygons part of Autodesk Maya® is a classic polygonal modeler, and lots of low-level and high-level functions are available for surface creation. The normals in input for every mesh are computed

as the average of the incident face normals with Maya's *soften edges* function.

In section 3.1, we compare the schemes on an arbitrary triangle mesh with a low triangle count ($1000-3000$ triangles). Similar meshes will occur in real-world uses of the schemes. Phong, Nagata and NLSA triangles use surfaces of degree at most 2. PN triangles is the only scheme using a surface of degree 3.

3.1. Arbitrary Triangle Meshes

In this section we want to compare the surfaces constructed by the four schemes on an arbitrary triangle mesh in Figure 5. It has a low triangle count that actually corresponds to the real-world utilization of these schemes.

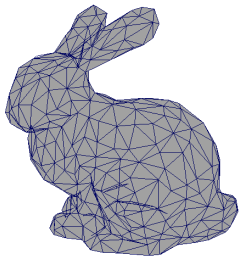


Figure 5: *Triangular mesh of a Bunny.*

In Figure 6 the four schemes are compared using three different normal patches in the shading process. In the first row analytic normals are used, directly computed from the patch as

$$\mathbf{n}(u,v) = \frac{\frac{\partial \mathbf{s}}{\partial u}(u,v) \times \frac{\partial \mathbf{s}}{\partial v}(u,v)}{\left\| \frac{\partial \mathbf{s}}{\partial u}(u,v) \times \frac{\partial \mathbf{s}}{\partial v}(u,v) \right\|}.$$

With analytic normals we can analyze the real surface geometry. The PN triangle's surface appears smoother than the other three quadratic surfaces due to its higher degree. Nonetheless, it also has to deal with the discontinuity of normals. The three quadratic schemes show more or less the same behavior, and normals discontinuity is clearly visible.

The normals that can be computed analytically from the constructed patch in general do not vary continuously from triangle to triangle. In particular, the schemes in this survey use only the three vertices and normals in the input. As a consequence, they do not control the patch normals. Dealing with this problem, in fact, would mean to impose more constraints on the control points defining the patch.

Thus, as solution, an independent normal patch (usually linear or quadratic) has been proposed together with the surface to improve the surface visualization as a sort of normal smoothing. Using these normals in the shading process, the surface appearance gives us the impression that it is smoother because curvature discontinuities are alleviated. It has to be pointed out that in this way the surface is simply enhanced in its visualization and not at all smoothed in its geometry. More details can be found in [VOW97b].

The second and the third row in Figure 6 show the difference between the use of linear and quadratic normals. Although quadratic models shaded using quadratic normal patches may sometimes show oscillations, they give the impression of more real surfaces. Linear normal patches, instead, result in more flattened surfaces.

In Figure 7, the same surfaces in Figure 6 are shaded with highlight lines ([HBB*08]).

In the first row, the four surfaces with analytic normals are compared. First, the difference in the case of analytic normals between each of the quadratic patches and the cubic one are clearly visible. Furthermore, there are large differences between the three quadratic patches. Nagata's surface seems to be the worst although it has to be noticed that dealing with this mesh Nagata's scheme suffers from stability problems and needs to keep a threshold ε quite high ($\varepsilon = 0.03$). Phong tessellation and NLSA triangles do not present a lot of differences, but their lines are very broken. Again PN triangles result in the best surface compared to the other three, nonetheless they show important discontinuities and breaks.

The situation changes when linear and quadratic independent normal patches are used. In the second and third row, the four surfaces do not present remarkable differences due to the fact that the same independent normal patch is used for each scheme. Rather, the difference between the linear normal patch and the quadratic normal patch is clearly visible in the surface visualization.

4. Conclusions

This survey covers parametric curved patches for C^0 interpolation of triangle meshes as used in computer graphics to save bandwidth when rendering curved models or sections of them. The main emphasis of the survey is to compare the surface quality of the different schemes available. Related techniques from computer graphics to improve the silhouette only ([GGH*99, SGG*00, DRS08]) were not taken into account. The comparison includes four different schemes based on Bézier triangles: PN triangles that are of degree 3; Phong triangles, NLSA triangles and Nagata triangles, all of degree 2.

For real-world models with moderately fine faces, the differences between all the schemes are quite small, as can be seen in Figures 3 and 6. Nagata triangles impose the additional restriction on the angles between the vertex normals to avoid artifacts.

Depending on the application, the user should decide whether the degree 3 scheme is required or one of the degree 2 schemes could be sufficient for his/her task.

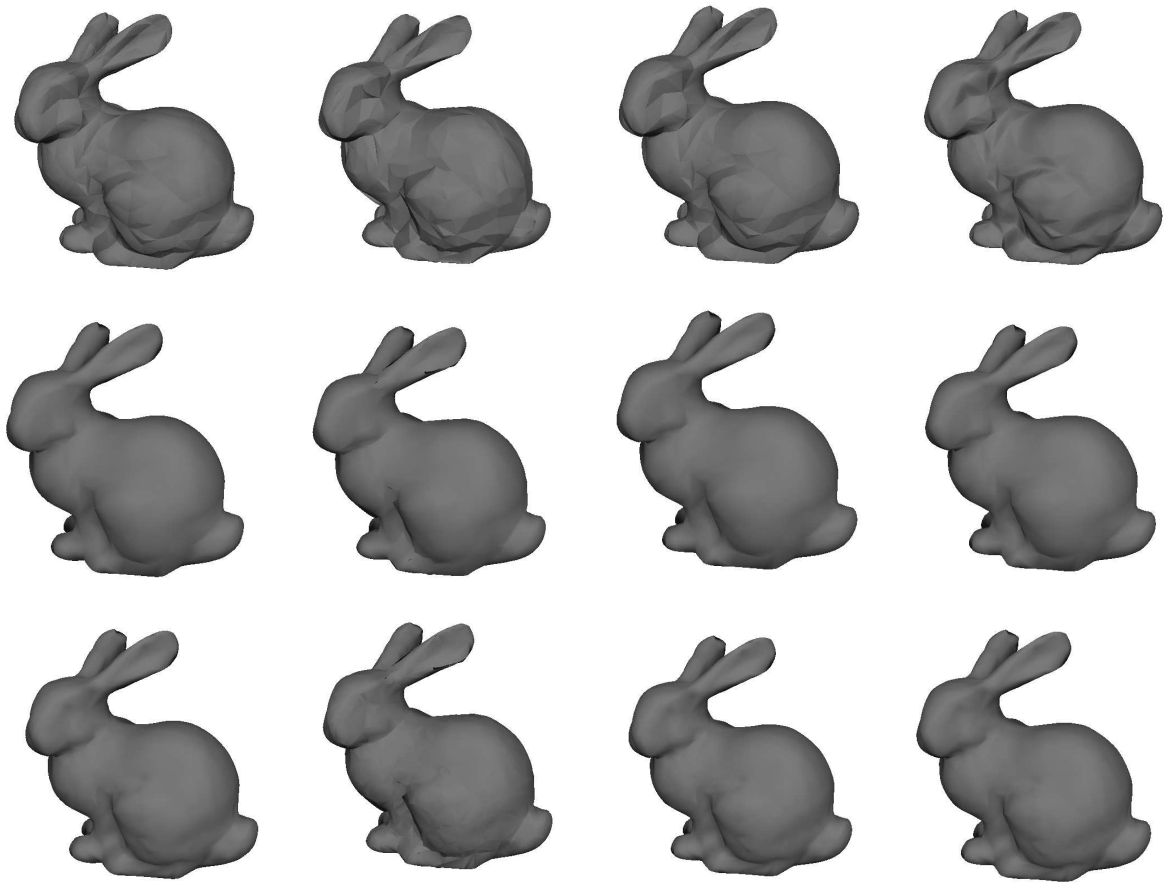


Figure 6: *Gouraud-shaded surfaces in columns from left to right: Phong, Nagata ($\varepsilon = 0.027$), NLSA, PN triangles. First row: Analytic normals. Second row: Linear normals. Third row: Quadratic normals.*

References

[BA08] BOUBEKEUR T., ALEXA M.: Phong tessellation. In *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers* (New York, NY, USA, 2008), pp. 1–5.

[BHB02] BARRERA T., HAST A., BENGTSSON E.: Surface construction with near least square acceleration based on vertex normals on triangular meshes. In *SIGRAD 2002 Conference Proceedings, Norrköping* (2002), pp. 17–22.

[BP07] BARAN I., POPOVIĆ J.: Automatic rigging and animation of 3d characters. *ACM Trans. Graph. 26*, 3 (2007), 72.

[Col05] COLLINS S.: Kinematics, dynamics, biomechanics: Evolution of autonomy in game animation. *Computer Graphics Forum 24*, 3 (2005).

[DRS08] DYKEN C., REIMERS M., SELAND J.: Real-time gpu silhouette refinement using adaptively blended bézier patches. *Computer Graphics Forum 27*, 1 (2008), 1–12.

[Far02] FARIN G.: *Curves and surfaces for CAGD: a practical guide*, 5th ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[GGH*99] GU X., GORTLER S., HOPPE H., MCMILLAN L., BROWN B., STONE A.: Silhouette mapping. *Technical Report TR-1-99, Department of Computer Science, Harvard University* (March 1999).

[HBB*08] HAHMANN S., BELYAEV A., BUSE L., ELBER G., MOURRAIN B., ROESSL C.: Shape interrogation - a state of the art. In *Shape Analysis and Structuring*, De Floriani L., Spagnuolo M., (Eds.). Springer, 2008, pp. 1–52.

[Kau04] KAUTZ J.: Hardware lighting and shading: a survey. *Computer Graphics Forum 23*, 1 (2004), 85–112.

[Nag05] NAGATA T.: Simple local interpolation of sur-

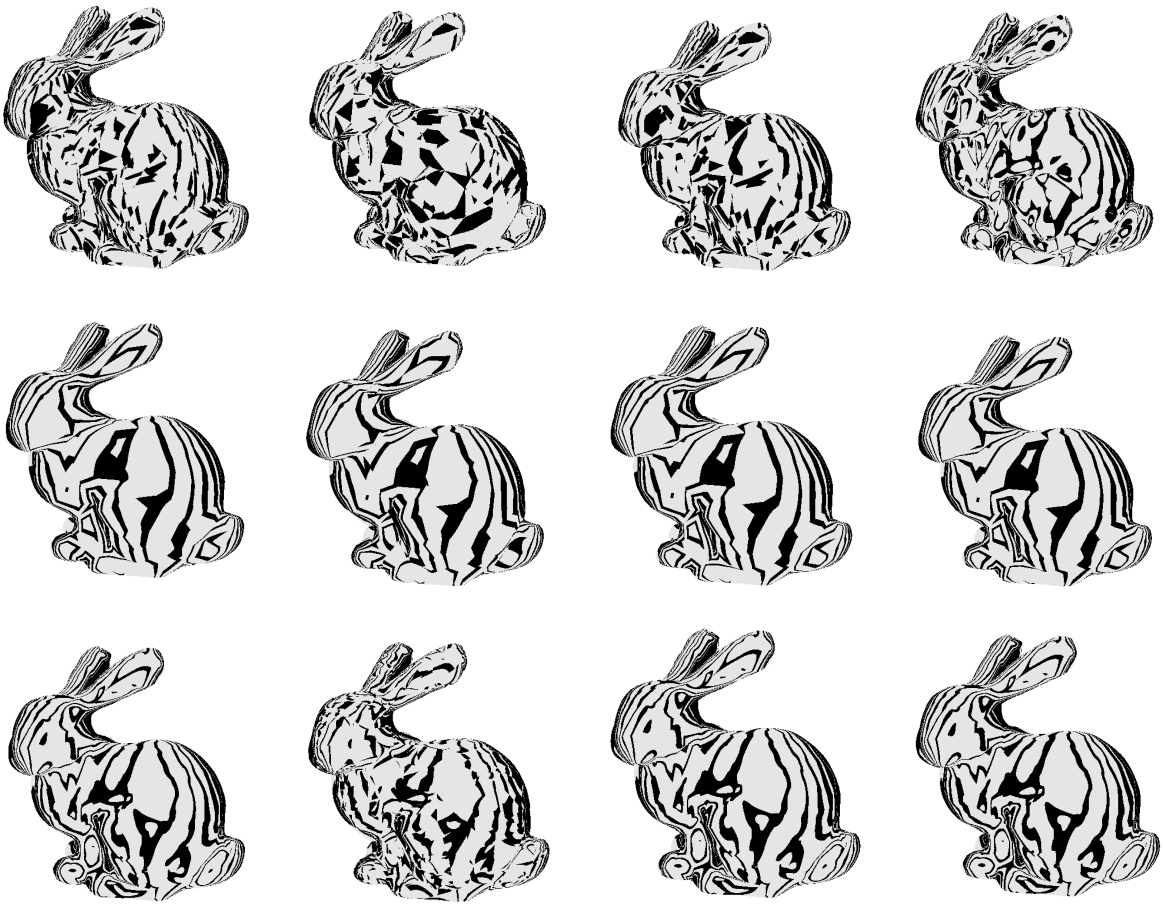


Figure 7: *Highlight lines: Phong, Nagata ($\varepsilon = 0.027$), NLSA, PN triangles. First row: Analytic normals. Second row: Linear normals. Third row: Quadratic normals.*

faces using normal vectors. *Computer Aided Geometric Design 22*, 4 (2005), 327–347.

[SGG*00] SANDER P., GU X., GORTLER S., HOPPE H., SNYDER J.: Silhouette clipping. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 327–334.

[SKUP*09] SZIRMAY-KALOS L., UMENHOFFER T., PATOW G., SZÉCSI L., SBERT M.: Specular effects on the gpu: State of the art. *Computer Graphics Forum 28*, 6 (2009), 1586–1617.

[VMT98] VOLINO P., MAGNENAT-THALMANN N.: The spherigon: A simple polygon patch for smoothing quickly your polygonal meshes. In *CA '98: Proceedings of the Computer Animation* (Washington, DC, USA, 1998), IEEE Computer Society, p. 72.

[VOW97a] VAN OVERVELD C. W. A. M., WYVILL B.: An algorithm for polygon subdivision based on vertex normals. In *CGI '97: Proceedings of the 1997 Conference on Computer Graphics International* (Washington, DC, USA, 1997), IEEE Computer Society, p. 3.

[VOW97b] VAN OVERVELD C. W. A. M., WYVILL B.: Phong normal interpolation revisited. *ACM Trans. Graph. 16*, 4 (1997), 397–419.

[VPBM01] VLACHOS A., PETERS J., BOYD C., MITCHELL J. L.: Curved pn triangles. In *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics* (New York, NY, USA, 2001), ACM, pp. 159–166.