

A comparison of local parametric C^0 Bézier interpolants for triangular meshes

Maria Boschioli^{a,b,*}, Christoph Fünfzig^c, Lucia Romani^b, Gudrun Albrecht^a

^a*Univ Lille Nord de France, UVHC, LAMAV-CGAO, FR no. 2956,
Le Mont Houy, F-59313 Valenciennes, France*

^b*Dipartimento di Matematica e Applicazioni, Università di Milano-Bicocca,
Via Cozzi 53, 20125 Milano, Italia*

^c*LE2I (UMR CNRS 5158), Université de Bourgogne,
9 Avenue Alain Savary, F-21078 Dijon, France*

Abstract

Parametric curved shape surface schemes interpolating vertices and normals of a given triangular mesh with arbitrary topology are widely used in computer graphics for gaming and real-time rendering due to their ability to effectively represent any surface of arbitrary genus. In this context, continuous curved shape surface schemes using only the information related to the triangle corresponding to the patch under construction, emerged as attractive solutions responding to the requirements of resource-limited hardware environments. In this paper we provide a unifying comparison of the local parametric C^0 curved shape schemes we are aware of, based on a reformulation of their original constructions in terms of polynomial Bézier triangles. With this reformulation we find a geometric interpretation of all the schemes that allows us to analyse their strengths and shortcomings from a geometrical point of view. Further, we compare the four schemes with respect to their computational costs, their reproduction capabilities of analytic surfaces and their response to different surface interrogation methods on arbitrary triangle meshes with a low triangle count that actually occur in their real-world use.

*Corresponding author: Maria Boschioli, Tel. +39 347 1566378, Fax +33 (0)3 27 51 19 00.

Email addresses: Maria.Boschioli@univ-valenciennes.fr (Maria Boschioli), Christoph.Fuenfzig@u-bourgogne.fr (Christoph Fünfzig), lucia.romani@unimib.it (Lucia Romani), Gudrun.Albrecht@univ-valenciennes.fr (Gudrun Albrecht)

Keywords:

C^0 local parametric interpolant, Bézier triangle, shape properties

1. Introduction

The easiest way of modeling free-form surfaces is to use tensor-product Bézier, B-Spline 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 of 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 in building a patchwork of smoothly joined parametric patches with the same topology as the control polygons. The topological information is usually specified 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, surface interpolation is a very useful and intuitive tool. 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 by connecting the points to create triangular planar faces. Obviously, with coarse meshes this leads to only continuous surfaces with poor shape; acceptable visually smooth shapes can be obtained only by deeply refining the triangular mesh.

In order to overcome this drawback, 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 only changes in the neighbourhood of the vertex. This feature is particularly attractive in most applications, however local parametric interpolation is an issue of ongoing research because often

the schemes do not provide surfaces of acceptable quality (i.e. 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 the rendered surfaces. Shading techniques like phong shading, normal mapping and reflection mapping are commonly used to present cineastically looking surfaces. For animation [1], models are applied with a suitable skeleton structure during rigging [2], 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 neighbourhood information. Several techniques have been leveraged for processing on programmable graphics hardware recently [3, 4].

Continuous (C^0) interpolant curved shape surface schemes emerged to address specific requirements of the resource-limited hardware environments and to offer smooth surfaces by visually enhancing the resulting C^0 surface by using as little information as possible. More precisely, the smallest amount of information about neighbouring triangles has to be used in constructing the patch, thus these schemes use only the positions and normals at the three vertices of the triangle.

The interest in continuous surface patches comes primarily from saving bus bandwidth for transfers to the graphics hardware. 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.

The aim of this paper is to provide a unifying comparison of the existing local parametric triangular curved shape C^0 schemes we are aware of. The paper is organized as follows. In section 2 we present four existing continuous surface schemes briefly describing 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 full detail. In section 3, we present additional features of two of these schemes. In section 4, we first analyse their computational costs (4.1), then we compare the schemes by looking at the reproduction of analytic surfaces like the sphere and the torus (section 4.2), and successively by looking at their response to surface interrogation methods on arbitrary triangle meshes with a low triangle count, which actually occur in real-world use of these schemes (section 4.3). Finally, in section 5 we briefly describe the use of separated normal interpolation patches in the shading process and in section 6 we give the conclusions of

our study.

2. Continuous Surface Schemes

Triangular Bézier patches have been around since the birth of Computer Aided Geometric Design. They were initially investigated by de Casteljau 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 we are going to present is that each original flat triangle of the input mesh can be replaced by a curved shape, namely a parametric cubic or quadratic polynomial triangular Bézier patch interpolating the three positions and normals of the vertices. Therefore, the patch's 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 of the triangle are used.

Let us denote the three triangle vertices by $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2$, the respective normal vectors by $\mathbf{n}_0, \mathbf{n}_1, \mathbf{n}_2$, and the edge vectors by $\mathbf{d}_1 = \mathbf{p}_1 - \mathbf{p}_0$, $\mathbf{d}_2 = \mathbf{p}_2 - \mathbf{p}_1$, $\mathbf{d}_3 = \mathbf{p}_0 - \mathbf{p}_2$, as shown in Figure 1. Additionally, we will refer to

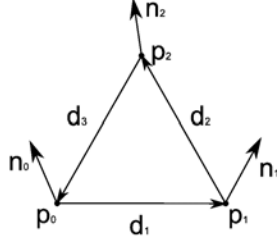


Figure 1: Notation for the vertices and respective normals of the input flat triangles.

the tangent plane in \mathbf{p}_i , which is defined by \mathbf{n}_i , by $\boldsymbol{\tau}_i$, $i = 0, 1, 2$.

Although each scheme uses its own formulation, for comparison purposes we will describe all the schemes in the form of triangular Bézier patches and we will analyse their geometrical interpretation.

Using a triangular network of control points

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

and degree- n bivariate Bernstein polynomials

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

a degree- n triangular Bézier patch is defined by

$$\mathbf{s}(u, v, w) = \sum_{i+j+k=n} \mathbf{b}_{ijk} B_{ijk}^n(u, v, w). \quad (1)$$

It maps a triangular domain $D \subset \mathbb{R}^2$ to an affine space, typically \mathbb{R}^3 , where u , v and w are the barycentric coordinates of a domain point relative to D . See [5] for details on the properties of these patches.

Together with the schemes that we are going to present in the following subsections, we would also like to cite [6, 7, 8, 9, 10] as interesting schemes related to the interpolation problem we are considering. However, we are not going to include them in our discussion because [6, 7] do not fit into the class of analytically representable curved patches, [8, 9] use neighboring triangles to construct cubic patches with approximate- G^1 continuity and, although Walton and Meek in [10] propose an interesting method leading to a G^1 surface, they make use of a blending type approach that results in a patch that is not purely polynomial.

2.1. PN Triangles

Curved PN triangles by Vlachos et al. [11], 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 on the tangent planes at the vertices.

The scheme initially places the intermediate control points $\bar{\mathbf{b}}_{ijk}$ 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 by projecting the respective intermediate control point $\bar{\mathbf{b}}_{ijk}$ 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 central control point \mathbf{b}_{111} is constructed moving the corresponding $\bar{\mathbf{b}}_{111}$ halfway in the direction $\mathbf{m} - \bar{\mathbf{b}}_{111}$, where \mathbf{m} is the average of the six control points just computed on the border.

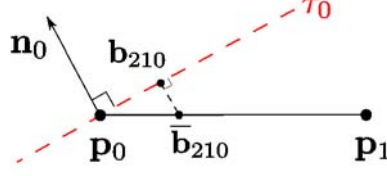


Figure 2: Construction of \mathbf{b}_{210} in PN triangle's scheme: projection of $\bar{\mathbf{b}}_{210} = (2\mathbf{p}_0 + \mathbf{p}_1)/3$ into the tangent plane at \mathbf{p}_0 .

In formulas:

$$\begin{aligned}
\mathbf{b}_{300} &= \mathbf{p}_0, \quad \mathbf{b}_{030} = \mathbf{p}_1, \quad \mathbf{b}_{003} = \mathbf{p}_2, \\
w_{ij} &= (\mathbf{p}_j - \mathbf{p}_i) \cdot \mathbf{n}_i, \quad i, j \in \{0, 1, 2\}, \quad i \neq j, \\
\mathbf{b}_{210} &= \frac{1}{3}(2\mathbf{p}_0 + \mathbf{p}_1 - w_{01}\mathbf{n}_0), \quad \mathbf{b}_{120} = \frac{1}{3}(2\mathbf{p}_1 + \mathbf{p}_0 - w_{10}\mathbf{n}_1), \\
\mathbf{b}_{021} &= \frac{1}{3}(2\mathbf{p}_1 + \mathbf{p}_2 - w_{12}\mathbf{n}_1), \quad \mathbf{b}_{012} = \frac{1}{3}(2\mathbf{p}_2 + \mathbf{p}_1 - w_{21}\mathbf{n}_2), \\
\mathbf{b}_{102} &= \frac{1}{3}(2\mathbf{p}_2 + \mathbf{p}_0 - w_{20}\mathbf{n}_2), \quad \mathbf{b}_{201} = \frac{1}{3}(2\mathbf{p}_0 + \mathbf{p}_2 - w_{02}\mathbf{n}_0), \\
\mathbf{m} &= \frac{1}{6}(\mathbf{b}_{210} + \mathbf{b}_{120} + \mathbf{b}_{021} + \mathbf{b}_{012} + \mathbf{b}_{102} + \mathbf{b}_{201}), \\
\bar{\mathbf{b}}_{111} &= \frac{1}{3}(\mathbf{p}_0 + \mathbf{p}_1 + \mathbf{p}_2), \\
\mathbf{b}_{111} &= \mathbf{m} + \frac{1}{2}(\mathbf{m} - \bar{\mathbf{b}}_{111}).
\end{aligned}$$

As shown in [11], each boundary curve stays close to its edge, because its control points constructed as above stay within a radius of $l/6$ of the edge, where l is the length of the longest triangle edge. Further, the particular choice for the central control point \mathbf{b}_{111} is, among other things, based on symmetry, see [12]. In this way, the curved patch provably remains close to the flat triangle, preserving the shape and avoiding interference with other curved triangles.

In [13] a reformulation of PN Triangles as a modified Nielson's side-vertex triangular mesh interpolation scheme [14] can be found. Furthermore, the two recent works [15, 16] propose, respectively, a new application of PN triangles and add some improvements to the original construction. In particular,

in the former, the key idea is to assign to each mesh vertex a set of three scalar tags that act as shape controllers, improving the surface geometry and shading. The latter proposes the use of PN triangles on silhouettes.

2.2. Phong Tessellation

Phong tessellation [17] 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 around each vertex the tangent plane defined by the vertex normal is the appropriate local geometry.

The patch is constructed by direct evaluation of the point of barycentric coordinates (u, v, w) in three simple steps. First, the barycentric combination between the three triangle vertices is computed. Then, this point is projected on the three tangent planes defined by the vertices and normals in input. Finally, the final evaluation point $\mathbf{s}^*(u, v, w)$ is obtained by the barycentric combination of these three projections.

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

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

where $\mathbf{p}(u, v, w) = u\mathbf{p}_0 + v\mathbf{p}_1 + w\mathbf{p}_2$ is simply the linear tessellation of the triangle and $\mathbf{s}^*(u, v, w)$ is detailed below. In [17], $\alpha = 3/4$ is proposed because this value experimentally provides convincing results in most of the situations. In section 3 details on the role of the parameter α will be analysed.

Writing out the definition of Phong tessellation, we obtain that $\mathbf{s}^*(u, v, w)$ is a quadratic patch with control points

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

where $\pi_i(\mathbf{p}_j)$ is the projection of \mathbf{p}_j on the tangent plane τ_i defined by \mathbf{n}_i and \mathbf{p}_i (see Figure 3).

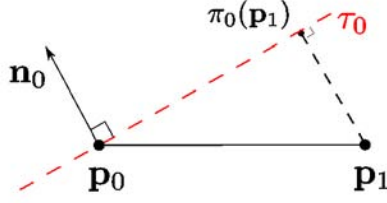


Figure 3: Projection of \mathbf{p}_1 into the tangent plane at \mathbf{p}_0 defined by \mathbf{n}_0 .

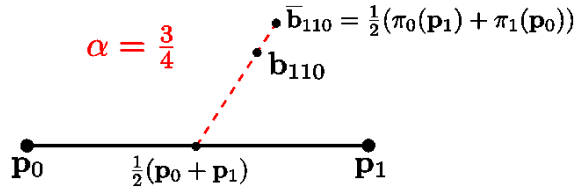


Figure 4: Geometric interpretation of the control point \mathbf{b}_{110} in Phong tessellation.

This allows us to compute a reformulation of the Phong tessellation patch $\mathbf{s}_\alpha(u, v, w)$ in quadratic Bézier triangle form with control points:

$$\begin{aligned}
 \mathbf{b}_{200} &= \mathbf{p}_0, & \mathbf{b}_{020} &= \mathbf{p}_1, & \mathbf{b}_{002} &= \mathbf{p}_2, \\
 \mathbf{b}_{110} &= \frac{1}{2}(\mathbf{p}_0 + \mathbf{p}_1) + \alpha \left[\bar{\mathbf{b}}_{110} - \frac{1}{2}(\mathbf{p}_0 + \mathbf{p}_1) \right], \\
 \mathbf{b}_{011} &= \frac{1}{2}(\mathbf{p}_1 + \mathbf{p}_2) + \alpha \left[\bar{\mathbf{b}}_{011} - \frac{1}{2}(\mathbf{p}_1 + \mathbf{p}_2) \right], \\
 \mathbf{b}_{101} &= \frac{1}{2}(\mathbf{p}_0 + \mathbf{p}_2) + \alpha \left[\bar{\mathbf{b}}_{101} - \frac{1}{2}(\mathbf{p}_0 + \mathbf{p}_2) \right].
 \end{aligned} \tag{2}$$

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 obtained by moving the middle-edge point in the direction given by the average of the projections of the edge corners, scaled by α . In Figure 4 the geometric interpretation of \mathbf{b}_{110} is shown.

Thanks to this clear geometric interpretation we found an interesting relation between PN triangles and Phong Tessellation. Looking at the edge \mathbf{d}_1 , for example, we can compute the corresponding control point \mathbf{b}_{110} for

Phong Tessellation with $\alpha = 1/3$ and with simple algebraic calculus we prove that

$$\begin{aligned}
\mathbf{b}_{110}^{phong} &= \frac{1}{2}(\mathbf{p}_0 + \mathbf{p}_1) + \frac{1}{3} \left[\frac{1}{2}(\pi_0(\mathbf{p}_1) + \pi_1(\mathbf{p}_0)) - \frac{1}{2}(\mathbf{p}_0 + \mathbf{p}_1) \right] = \\
&= \frac{1}{2}(\mathbf{p}_0 + \mathbf{p}_1) + \frac{1}{3} \left[\frac{1}{2}((\mathbf{p}_1 - \mathbf{p}_0) \cdot \mathbf{n}_0)\mathbf{n}_0 - \frac{1}{2}((\mathbf{p}_1 - \mathbf{p}_0) \cdot \mathbf{n}_1)\mathbf{n}_1 \right] = \\
&= \frac{1}{2}(\mathbf{p}_0 + \mathbf{p}_1) + \frac{1}{6}(\mathbf{p}_0 \cdot \mathbf{n}_0 - \mathbf{p}_1 \cdot \mathbf{n}_0)\mathbf{n}_0 - \frac{1}{6}(\mathbf{p}_0 \cdot \mathbf{n}_1 - \mathbf{p}_1 \cdot \mathbf{n}_1)\mathbf{n}_1 = \\
&= \frac{1}{2}(\mathbf{b}_{210}^{PN} + \mathbf{b}_{120}^{PN}).
\end{aligned}$$

The control point for the quadratic Bézier triangle of Phong Tessellation with $\alpha = 1/3$ is the average of the two control points of PN triangles relative to the same edge.

2.3. Nagata Triangles

The central idea in the scheme proposed by T. Nagata in [18] is quadratic interpolation of a curved segment from the position and normal vectors at the end-points, with the aid of generalized inverses (or pseudo-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 the interior of the patch with a parametric quadratic surface reproducing the modified boundaries. Central in the patch construction is a coefficient \mathbf{c}_1 (respectively \mathbf{c}_2 , \mathbf{c}_3) that defines the boundary curve in the monomial form $\mathbf{x}_1(t) = \mathbf{p}_0 + (\mathbf{d}_1 - \mathbf{c}_1)t + \mathbf{c}_1 t^2$ as concerns the edge given by \mathbf{d}_1 , for example. The analytical formula used for the generalized inverse \mathbf{A}^+ is

$$\mathbf{A}^+ = \lim_{\alpha \rightarrow 0^+} (\mathbf{A}^* \mathbf{A} + \alpha \mathbf{E})^{-1} \mathbf{A}^*,$$

where \mathbf{A} and \mathbf{A}^* are an arbitrary matrix and its transposed conjugate, respectively, and \mathbf{E} is the identity matrix of consistent dimension. This allows to solve the system of equations with unknown \mathbf{c}_1 .

The solution for the coefficient \mathbf{c}_1 related to the edge \mathbf{d}_1 results in

$$\mathbf{c}_1 = \begin{cases} \frac{\Delta d}{1-\Delta c} \boldsymbol{\nu} + \frac{d}{\Delta c} \Delta \boldsymbol{\nu}, & c \neq \pm 1 \\ \mathbf{0}, & c = \pm 1 \end{cases} \quad (3)$$

where $\boldsymbol{\nu} = \frac{\mathbf{n}_0 + \mathbf{n}_1}{2}$ and $\Delta\boldsymbol{\nu} = \frac{\mathbf{n}_0 - \mathbf{n}_1}{2}$ are the average and the deviation of the unit normals, $d = \mathbf{d}_1 \cdot \boldsymbol{\nu}$ and $\Delta d = \mathbf{d}_1 \cdot \Delta\boldsymbol{\nu}$ their inner products with \mathbf{d}_1 and

$$\Delta c = \mathbf{n}_0 \cdot \Delta\boldsymbol{\nu}, \quad c = \mathbf{n}_0 \cdot \mathbf{n}_1 = 1 - 2\Delta c. \quad (4)$$

In this definition two denominators obviously should not become zero and for this reason when $\mathbf{n}_0 \cdot \mathbf{n}_1 = \pm 1$ the coefficient is set to zero. Unfortunately this leads to stability problems when these two denominators are nearly zero. More details will be given in section 3.2. Analogous formulas hold for \mathbf{c}_2 and \mathbf{c}_3 related, respectively, to \mathbf{d}_2 and \mathbf{d}_3 .

Although the monomial form allows an easy and fast coefficients computation (see computational costs in section 4), a triangular Bézier formulation of the patch, which can be obtained by means of a change of parametrization, allows a better geometrical insight. The three control points \mathbf{b}_{110} , \mathbf{b}_{101} and \mathbf{b}_{011} are defined by moving the average of the vertices on an edge halfway the direction given by the curvature coefficient related to that edge:

$$\begin{aligned} \mathbf{b}_{200} &= \mathbf{p}_0, & \mathbf{b}_{110} &= \frac{1}{2}(\mathbf{p}_0 + \mathbf{p}_1) - \frac{1}{2}\mathbf{c}_1, \\ \mathbf{b}_{020} &= \mathbf{p}_1, & \mathbf{b}_{011} &= \frac{1}{2}(\mathbf{p}_1 + \mathbf{p}_2) - \frac{1}{2}\mathbf{c}_2, \\ \mathbf{b}_{002} &= \mathbf{p}_2, & \mathbf{b}_{101} &= \frac{1}{2}(\mathbf{p}_0 + \mathbf{p}_2) - \frac{1}{2}\mathbf{c}_3. \end{aligned}$$

In this formulation some easy calculations allow to show that the central control point on one edge, for instance \mathbf{b}_{110} , is on the intersection line L between the two tangent planes $\boldsymbol{\tau}_0$ and $\boldsymbol{\tau}_1$, i.e. $L = \boldsymbol{\tau}_0 \cap \boldsymbol{\tau}_1$. Moreover, the control point \mathbf{b}_{110} is the point on that line L that minimizes the distance between the middle-edge point $\frac{1}{2}(\mathbf{p}_0 + \mathbf{p}_1)$ and the line (see Figures 5 and 6).

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 [19]. 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 with direction \mathbf{d}_1 . The authors' approach, after computation of tangent vectors $\mathbf{t}_0 \in \tau_0$ and $\mathbf{t}_1 \in \tau_1$ from the normals \mathbf{n}_0 and

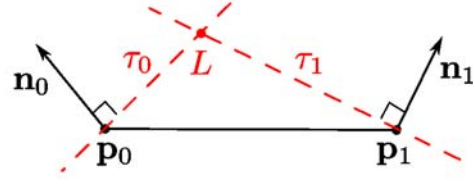


Figure 5: The intersection line L between the two tangent planes τ_0 and τ_1 (schematic view).

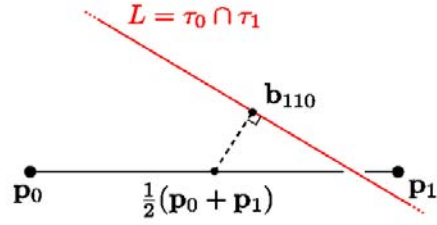


Figure 6: Geometric interpretation of the middle-edge control point in Nagata's scheme. \mathbf{b}_{110} is the point on the line L that minimizes the distance between the middle-edge point $\frac{1}{2}(\mathbf{p}_0 + \mathbf{p}_1)$ and the line L .

\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}_1 is as close to the tangent \mathbf{t}_1 as the least square minimization allows. Then, they compute a 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.

The curve $\mathbf{q}_1(t)$ is computed by solving a system of two equations in the unknowns α_1 and β_1 whose solution is

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

and analogously the curve $\mathbf{q}_2(t)$ is computed by solving a system whose solution is

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

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 $\alpha_2, \beta_2, \bar{\alpha}_2, \bar{\beta}_2$ and $\alpha_3, \beta_3, \bar{\alpha}_3, \bar{\beta}_3$.

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:

$$\begin{aligned} \mathbf{b}_{200} &= \mathbf{p}_0, & \mathbf{b}_{110} &= \frac{1}{2}(\mathbf{p}_0 + \mathbf{p}_1) + \frac{1}{4}(\alpha_1\beta_1\mathbf{t}_0 - \bar{\alpha}_1\bar{\beta}_1\mathbf{t}_1), \\ \mathbf{b}_{020} &= \mathbf{p}_1, & \mathbf{b}_{011} &= \frac{1}{2}(\mathbf{p}_1 + \mathbf{p}_2) + \frac{1}{4}(\alpha_2\beta_2\mathbf{t}_1 - \bar{\alpha}_2\bar{\beta}_2\mathbf{t}_2), \\ \mathbf{b}_{002} &= \mathbf{p}_2, & \mathbf{b}_{101} &= \frac{1}{2}(\mathbf{p}_0 + \mathbf{p}_2) + \frac{1}{4}(\alpha_3\beta_3\mathbf{t}_0 - \bar{\alpha}_3\bar{\beta}_3\mathbf{t}_2). \end{aligned}$$

Note that $\alpha_1\beta_1\mathbf{t}_0$ is just the projection of the edge vector \mathbf{d}_1 on \mathbf{t}_0 . Similarly, $\bar{\alpha}_1\bar{\beta}_1\mathbf{t}_1$ is the projection of \mathbf{d}_1 on \mathbf{t}_1 (see Figure 7).

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 subtraction of the projections of the edge on the tangents at the two vertices.

This subtraction results in the same direction as the one used in Phong tessellation definition (2) of the control points. More precisely, by algebraic

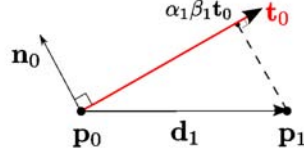


Figure 7: $\alpha_1 \beta_1 \mathbf{t}_0$ is the projection of \mathbf{d}_1 on \mathbf{t}_0 .

computations it can be proved that

$$\alpha_1 \beta_1 \mathbf{t}_0 - \bar{\alpha}_1 \bar{\beta}_1 \mathbf{t}_1 = \pi_0(\mathbf{p}_1) + \pi_1(\mathbf{p}_0) - (\mathbf{p}_0 + \mathbf{p}_1),$$

and thus

$$\begin{aligned} \mathbf{b}_{110}^{NLSA} &= \frac{1}{2}(\mathbf{p}_0 + \mathbf{p}_1) + \frac{1}{4}(\alpha_1 \beta_1 \mathbf{t}_0 - \bar{\alpha}_1 \bar{\beta}_1 \mathbf{t}_1) = \\ &= \frac{1}{2}(\mathbf{p}_0 + \mathbf{p}_1) + \frac{1}{2} \left(\frac{1}{2}(\pi_0(\mathbf{p}_1) + \pi_1(\mathbf{p}_0)) - \frac{1}{2}(\mathbf{p}_0 + \mathbf{p}_1) \right) \\ &= \mathbf{b}_{110}^{phong(\alpha=\frac{1}{2})}. \end{aligned}$$

The triangular patch obtained by the NLSA method is nothing else than the Phong tessellation patch with $\alpha = \frac{1}{2}$.

3. Additional Features

Before comparing the four schemes, we analyse two additional features of two of them. In section 3.1, we focus on the role of the shape parameter α in Phong tessellation and its relation to NLSA triangles, and in section 3.2 we analyse and try to find a solution to the stability problem of Nagata triangles.

3.1. Shape Parameter of Phong Tessellation

Phong tessellation is the only scheme with a parameter to regulate the surface flatness. Unfortunately, this is a global parameter while sometimes only some local parts of the surface need to be changed. It is also true that Phong tessellation introduces such a parameter because usually the obtained surface is too inflated, while the other schemes do not have an issue with flatness and thus do not need a parameter. In Figure 8 the surfaces constructed from a simple mesh for different values of the parameter (α) show how this parameter controls the shape of the final Phong surface.

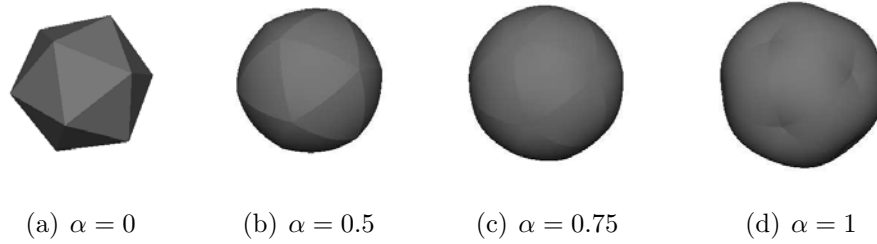


Figure 8: Surfaces constructed with Phong tessellation for different values of α .

3.2. Stability Problems of Nagata Triangles

As stated before in section 2.3, equation (4) defining the curvature coefficient \mathbf{c}_1 in Nagata's scheme has a stability problem that might strongly affect the surface.

In this formula, 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, this problem occurs on simple meshes where avoiding these two configurations it 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.

Looking at the geometric interpretation of Nagata's patch, this stability problem is even clearer. If the two tangent planes are parallel or nearly parallel, their intersection line L goes to infinity and as a consequence the middle-edge control point goes to infinity too, making the patch very inflated on that edge because this minimum distance point can be spatially very far away from the model edge.

One way to correct this problem is to use a threshold ϵ in the coefficient definition (3), setting \mathbf{c}_1 to $\mathbf{0}$ if $\Delta c \leq \epsilon$ or $1 - \Delta c \leq \epsilon$ (or equivalent conditions on c). For instance, in Figure 9 we give an example of how the final surface changes in dependence of ϵ .

In all our experiments in the next section, the threshold ϵ must be set quite high to have an acceptable result. This means that, depending on the configuration of the normals, the intersection line L can be far away from

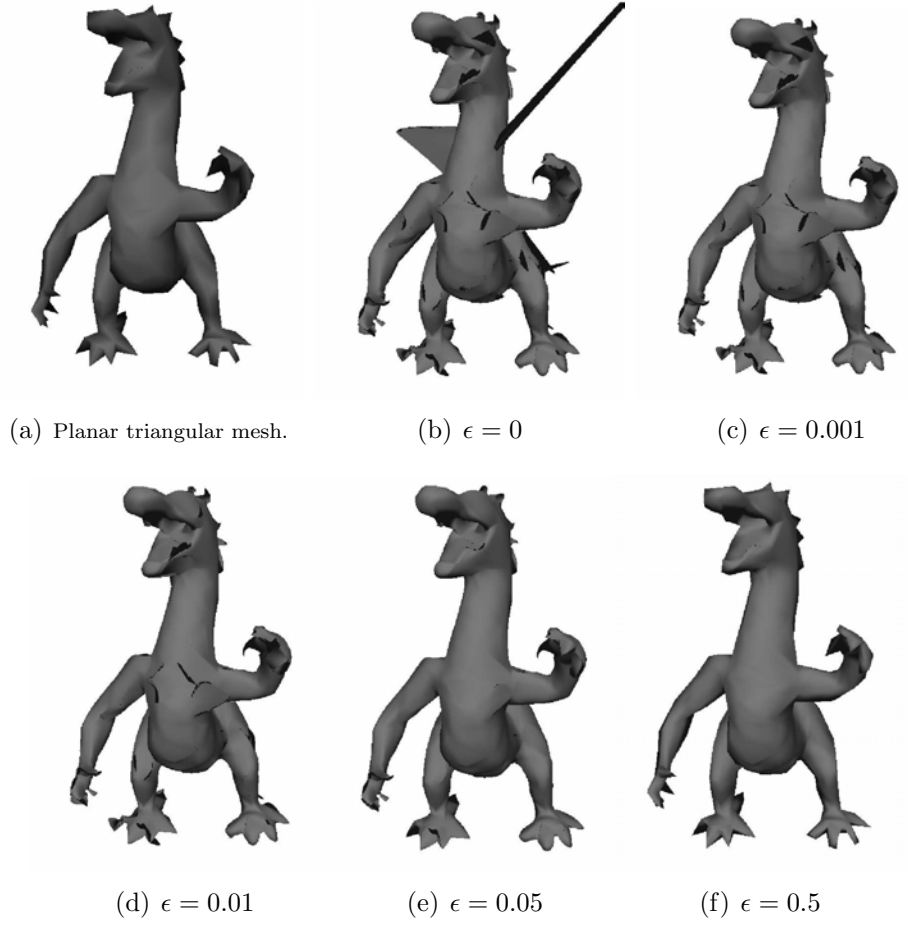


Figure 9: Stability problem of Nagata's scheme.

the plane of the triangle. Naturally, the higher the threshold, the flatter the constructed surface, because if the three curvature coefficients \mathbf{c}_1 , \mathbf{c}_2 and \mathbf{c}_3 are set to zero we obtain the linear triangular patch. On the contrary, with very regular meshes as for instance the torus and the sphere (see section 4.2), surfaces with no artifacts can be obtained with small ϵ thanks to symmetry and well-conditioned configurations.

4. Comparison

We implemented all the schemes as an Autodesk Maya® plug-in (*MPx-HwShaderNode*), based on the plug-in from [20]. The Polygons part of Autodesk Maya® is a classic polygonal modeller, and lots of low-level and high-level functions are available for surface creation.

4.1. Computational costs

Before comparing the surface quality of the four schemes, we compare their computational costs. We computed manually the number of scalar additions and scalar multiplications required for evaluation of a point on the cubic or quadratic Bézier triangle. Then, for the difference in practice, we measured the time required for the tessellation on the CPU by using a 1000 triangles Bunny mesh, tessellating every triangle patch into 210 points (tessellation factor $f = 20$). In the vertex shader on the GPU, we tessellated the patch into 210 points (tessellation factor $f = 20$) and 1830 points (tessellation factor $f = 60$), which are handled as OpenGL vertex buffer objects. As the shading is completely vertex shader-bound, we measured the time for vertex shading and fragment shading together. The total rendering time for the planar triangle mesh of the Bunny is 7.05 ms using OpenGL vertex arrays. These measurements were performed on a MS Windows Vista 32bit system with Intel P8700 (2.5 GHz) processor and NVidia Geforce 9600GT (512 MB) mobile graphics.

Table 1 gives the required operations for the evaluation of a point on the surface in the Bézier form, splitted into control points (cp) computation and patch evaluation according to definition (1). We computed the same values for the original construction given by the respective papers of Nagata triangles and Phong tessellation (Table 2). Note that Phong tessellation in their original construction makes use of a direct per-point computation; there is no separate per-patch computation anymore.

In general, the evaluation of a surface point on the cubic patch is more expensive than on a quadratic patch, which makes a difference for the scalar CPU implementation (not using SIMD extensions). The tessellation time for the cubic PN patch is considerably larger than for any of the quadratic patches. Concerning the difference between the Bézier formulation and the original formulation on the CPU, Nagata triangles are the most efficient due to the formulation in the canonical basis in u and v . In contrast, the combination of construction and point evaluation into one step for Phong

Bézier form	Per-patch: cp construction		Per-point: evaluation		Total		CPU $f = 20$	GPU	
	add	mult	add	mult	add	mult	$f = 20$ ms	$f = 20$ ms	$f = 60$ ms
PN triangles	93	81	27	57	120	138	0.112	0.036	0.061
Nagata triangles	72	87	15	27	87	114	0.071	0.035	0.061
NLSA triangles	123	153	15	27	138	180	0.071	0.038	0.066
Phong tessellation	57	45	15	27	72	72	0.070	0.035	0.054

Table 1: Scalar additions and multiplications required by all the schemes to evaluate the patch and time required for the tessellation on the CPU and on the GPU using the Bézier form.

Original form	Per-patch		Per-point		Total		CPU $f = 20$	GPU	
	add	mult	add	mult	add	mult	$f = 20$ ms	$f = 20$ ms	$f = 60$ ms
PN triangles	93	81	27	57	120	138	0.112	0.036	0.061
Nagata triangles	54	69	21	21	75	90	0.054	0.034	0.059
NLSA triangles	123	153	15	27	138	180	0.071	0.038	0.066
Phong tessellation	—	—	30	36	30	36	0.094	0.034	0.050

Table 2: Scalar additions and multiplications required by all the schemes to evaluate the patch using the original construction proposed by their authors. Note that PN triangles and NLSA triangles are originally defined in the Bézier form; we just repeat the numbers from Table 1 for them.

tessellation is slower on the CPU as it needs to be done per surface point and requires more operations than the evaluation of a quadratic Bézier triangle (15 adds, 27 mults). This combination into one step is especially beneficial for the evaluation in the vertex shader of the GPU.

For the GPU evaluation, we compare the time differences for tessellation factor $f = 60$. Interpolation in the vertex shader requires patch construction and evaluation per surface point, so that the schemes with small total costs (Phong tessellation and Nagata triangles) are advantageous. PN triangles are slightly slower, and NLSA triangles are the slowest due to the largest total costs among all the schemes. Nagata triangles and Phong tessellation triangles require less additions/multiplications in their original construction for point evaluation compared to the Bézier form (Table 2), as argued in [17] and [18]. This results also in better timings, which become even more significant for larger tessellation factors.

4.2. Sphere and Torus Interpolation

In this section, we compare the behaviour of the four schemes with respect to a known surface. We compare the signed distance between the analytic surface (a sphere and a torus) and the piecewise parametric interpolants computed by the schemes on a sampling of points and normals from that

surface. We are especially interested in the schemes behaviour when refining the base mesh of the piecewise parametric surface.

The base mesh for the sphere is an icosahedron sampled from a sphere of radius r . At any refinement step i it is refined by means of a 4-split division of the triangles, which results in triangle meshes with $20 \cdot 4^i$ triangles, *i.e.*, 20 for $i = 0$, 80 for $i = 1$, 320 for $i = 2$, and 1280 triangles for $i = 3$.

The base mesh for the torus is generated by a subdivision of the bivariate parameter domain $[0, 2\pi) \times [0, 2\pi)$ into j^2 quadrangular regions. After the refinement, the quadrangular mesh is triangulated adding the diagonals. This results in $2 \cdot j^2$ triangles at any refinement step j ($j = 1, 2, 3, \dots$).

We measure the signed distance between the analytic surface and the piecewise parametric interpolant along the patch normal for the refinement steps $i = 0, 1, 2, 3$, in case of the sphere, and for $j = 0, \dots, 19$, in case of the torus. Iterations $i = 3$ and $j = 19$, respectively, yield acceptable average distance values.

Figure 10 shows the approximation behaviour of the average signed distance to the sphere with radius $r = 1$. For this radius, the Phong triangles bend to the exterior, whereas the NLSA triangles and PN triangles are always interior to the sphere. The Nagata triangles ($\epsilon = 0$) result in the best average distances, which is a consequence of reproducing the tangent planes at the vertices for this degree-2 scheme. PN triangles also reproduce the given tangent planes in the sphere points but, due to the general choice of the control point \mathbf{b}_{111} , are less curved than the sphere.

Figure 11 shows the approximation behaviour of the average signed distance to a torus with radii $r_1 = 1$ and $r_2 = 0.5$. The behaviour of the different schemes is qualitatively the same for the torus as for the sphere above. Again Nagata triangles ($\epsilon = 0$) result in the best average distance. Due to the general choice of the control point \mathbf{b}_{111} , the PN triangles are less curved than the original torus and consequently are always in the interior. The same is true for NLSA triangles, while the contrary occurs for Phong tessellation.

Data in Table 3 confirm the following classification of the methods with respect to their approximation behaviours: Nagata triangles perform best, followed by PN triangles and NLSA triangles, whereas Phong tessellation exhibits the worst approximation behaviour.

4.3. Arbitrary Triangle Meshes

In this section, we want to compare the surfaces constructed by the four schemes on arbitrary triangle meshes with a low triangle count (1000 – 3000

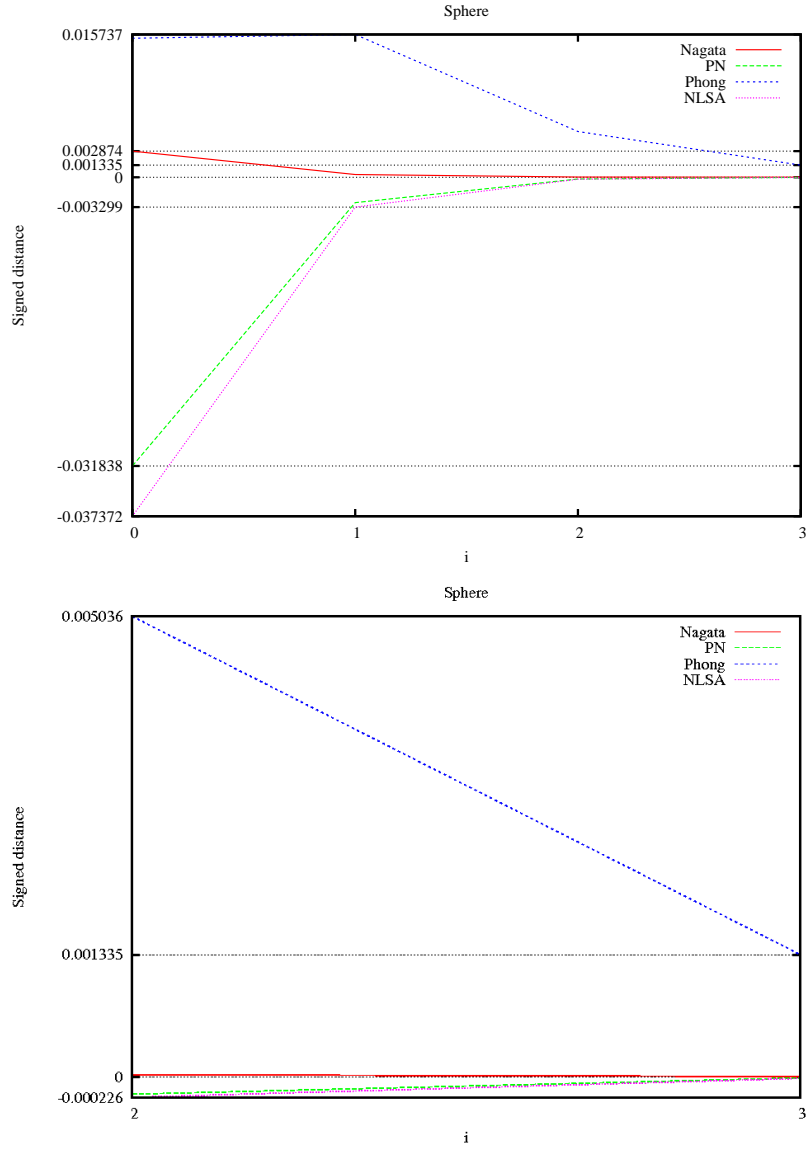


Figure 10: Average signed distance of sphere interpolation depending on the refinement step i . Top: approximation behaviour for $i = 0, \dots, 3$. Bottom: zoom on $i = 2, 3$.

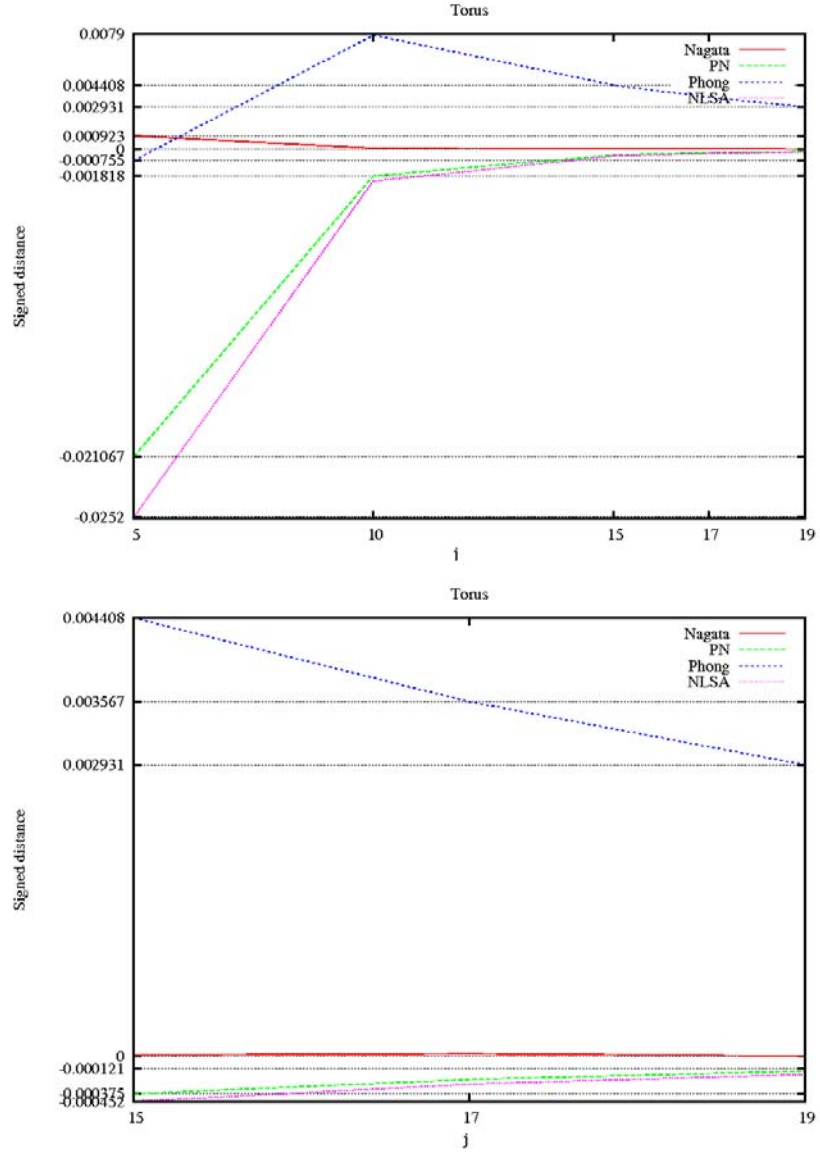


Figure 11: Average signed distance of torus interpolation depending on the refinement step j . Top: approximation behaviour for $j = 5, 10, 15, 17, 19$. Bottom: zoom on $j = 15, 17, 19$.

Name	Step	Mean Distance \pm std dev		Min,Max Distance
Sphere	$i = 0$	Phong	0.0153196 ± 0.00940232	[0, 0.0269864]
		Nagata $\epsilon = 0$	$0.00287491 \pm 0.00437559$	[-0.00290352, 0.0131106]
		NLSA	-0.0373717 ± 0.0231368	[-0.0586981, 0]
		PN	-0.0318387 ± 0.0225525	[-0.0586009, 0]
	$i = 1$	Phong	0.0157366 ± 0.00930732	[0, 0.0233265]
		Nagata $\epsilon = 0$	$0.000301524 \pm 0.000336243$	[-7.21216e-5, 0.00125933]
		NLSA	$-0.00329917 \pm 0.00209737$	[-0.00633496, 0]
		PN	$-0.0028053 \pm 0.00203187$	[-0.00632477, 0]
	$i = 2$	Phong	$0.00503575 \pm 0.00300305$	[-5.96046e-8, 0.00815177]
		Nagata $\epsilon = 0$	$2.07119e-5 \pm 2.18946e-5$	[-1.13249e-6, 8.9407e-5]
		NLSA	$-0.000225536 \pm 0.000143714$	[-0.000468373, 0]
		PN	$-0.000191595 \pm 0.000139045$	[-0.000467658, 0]
	$i = 3$	Phong	$0.00133509 \pm 0.000797624$	[-5.96046e-8, 0.00221264]
		Nagata $\epsilon = 0$	$1.29748e-6 \pm 1.37613e-6$	[-1.78814e-7, 5.84126e-6]
		NLSA	$-1.44418e-5 \pm 9.19883e-6$	[-3.06964e-5, 0]
		PN	$-1.22342e-5 \pm 8.88072e-6$	[-3.05772e-5, 0]
Torus	$j = 5$	Phong	$-0.000754569 \pm 0.0315128$	[-0.111965, 0.0630175]
		Nagata $\epsilon = 0$	$0.000922857 \pm 0.0386565$	[-0.112992, 0.0931029]
		NLSA	-0.0252151 ± 0.0498816	[-0.195886, 0.0383056]
		PN	-0.0210668 ± 0.0457862	[-0.195886, 0.0467965]
	$j = 10$	Phong	$0.00783327 \pm 0.00929549$	[-0.00958499, 0.0287549]
		Nagata $\epsilon = 0$	$9.40043e-5 \pm 0.00211594$	[-0.00666818, 0.00601918]
		NLSA	$-0.00217635 \pm 0.0041617$	[-0.0149165, 0.00385857]
		PN	$-0.00181822 \pm 0.00374165$	[-0.0147308, 0.00398123]
	$j = 15$	Phong	$0.00440836 \pm 0.00541819$	[-0.00492978, 0.0166597]
		Nagata $\epsilon = 0$	$1.57127e-5 \pm 0.000580865$	[-0.00252217, 0.00191915]
		NLSA	$-0.000451227 \pm 0.000939187$	[-0.00340354, 0.00134289]
		PN	$-0.000376097 \pm 0.000814623$	[-0.00340357, 0.00117016]
	$j = 17$	Phong	$0.00356666 \pm 0.00441199$	[-0.00393245, 0.0138325]
		Nagata $\epsilon = 0$	$2.97925e-5 \pm 0.00039676$	[-0.00130969, 0.00172943]
		NLSA	$-0.000275727 \pm 0.000596803$	[-0.00208876, 0.000976563]
		PN	$-0.000229708 \pm 0.000509889$	[-0.00208879, 0.000768423]
	$j = 19$	Phong	$0.00293179 \pm 0.00364289$	[-0.0032014, 0.0115674]
		Nagata $\epsilon = 0$	$6.28688e-6 \pm 0.000270014$	[-0.00122377, 0.000931621]
		NLSA	$-0.000177753 \pm 0.0004006$	[-0.00135079, 0.000708401]
		PN	$-0.000148038 \pm 0.000337174$	[-0.0013507, 0.000557363]

Table 3: Statistics of signed distances to sphere and torus surfaces: mean distance with standard deviation (defined as $\sqrt{\frac{1}{n-1} \sum_{k=1}^n (x_k - \bar{x})^2}$, x_k being the distance values) and minimum and maximum distance.

triangles) because similar meshes will occur in a real-world use of the schemes.

Table 4 gives detailed information about the meshes in our experiments. As the vertex normals have strong influence on the constructed surfaces, the table gives statistics on the angle cosine between the vertex normals and the triangle normals. This is a rough measure for the curvedness of the triangular patches. Additionally, the border curves are classified into convex, concave, and inflection, by the directions of the two vertex normals relative to a plane orthogonal to the edge (details in [11, 21]).

We analyse the surfaces from four meshes, Bunny, Monsterfrog, Vase and RoundedCube, by using highlight lines and Gaussian curvature plots [22]. Bunny and Monsterfrog represent two arbitrary fine triangle meshes. The remaining two meshes, Vase and RoundedCube, were chosen because they exhibit certain special effects for some of the schemes. We use the exact surface normals $\mathbf{n}(u, v) = \frac{\frac{\partial \mathbf{s}}{\partial u}(u, v) \times \frac{\partial \mathbf{s}}{\partial v}(u, v)}{\left\| \frac{\partial \mathbf{s}}{\partial u}(u, v) \times \frac{\partial \mathbf{s}}{\partial v}(u, v) \right\|}$ for computing the Gaussian curvature. Table 5 contains the minimum, maximum and mean values, with standard deviation defined as $\sqrt{\frac{1}{n-1} \sum_{k=1}^n (x_k - \bar{x})^2}$, of the Gaussian curvature computed on a dense sampling grid of 210 points per patch.

Analysing these statistics we found that the mean and maximum values of curvature for NLSA triangles are always smaller in the absolute value than the respective mean and maximum values for Phong Tessellation, meaning that the shape parameter, in a certain sense, controls also the curvature of the constructed surface. Further, NLSA triangles have the lowest standard deviation values of all the methods, meaning that the curvature is in general nearby the average value and few points have strongly deviating curvatures. Finally, in all the models there is a remarkable difference between the minimum and maximum value of the curvature for Nagata triangles, confirming that the stability problem deeply affects the resulting overall surface shape.

Additionally, in Table 6 we computed the angle cosine of the two normals in the same point of the edge between two adjacent patches. Although the C^0 patch construction does not guarantee G^1 continuity, this can be approximately the case, indicated by a mean value close to 1. It can be seen in a certain sense as a measure of “how far the patches are from having G^1 continuity”. The average cosine angle is computed by sampling 20 points along each edge.

In general, for all our examples, PN triangles give the best continuity, in the sense that the mean angle cosine between the normals is the nearest to 1 with the least standard deviations (except for the sphere). For the Bunny,







Name	#V/#E/#T	Mean Angle Normals \pm std dev	Min,Max Angle Normals	#E convex/concave/inflection
 Bunny	502/1500/1000	0.911775 ± 0.11007	[0.00118579, 0.999985]	556/71/873
 Monsterfrog	1308/3876/2584	0.867032 ± 0.131679	[-0.263223, 0.999968]	1602/233/2040
 Vase	241/690/460	0.91861 ± 0.0825731	0.428546, 0.999982]	122/115/452
 Rounded Cube	30/84/56	0.812907 ± 0.136409	[0.688079, 1]	44/0/40
 Sphere $r = 1.0$	162/480/320	0.98479 ± 0.00113563	[0.982247, 0.985606]	320/0/160
 Torus $r_1 = 1.0$ $r_2 = 0.5$	100/300/200	0.927627 ± 0.0169889	[0.901404, 0.951429]	135/25/140

Table 4: Statistics of triangle meshes: number of vertices/edges/triangles, angle cosine between vertex and triangle normals (mean \pm standard deviation, minimum and maximum), number of convex, concave, inflection edges.

Name	Mean Gaussian Curv \pm std dev	Min,Max Gaussian Curv
Bunny Phong	-276.54 ± 7782.39	$[-962065.0, 57399.7]$
Bunny Nagata $\epsilon = 0.03$	$-23256.3 \pm 2.4e+6$	$[-9.0e+8, 2.4e+8]$
Bunny NLSA	-28.1 ± 2709.6	$[-50854.5, 28265.6]$
Bunny PN	-383.9 ± 7981.8	$[-348286.0, 412429.0]$
Monsterfrog Phong	0.177 ± 3.8	$[-516.303, 186.17]$
Monsterfrog Nagata $\epsilon = 0.01$	-2.974 ± 412.0	$[-174208, 48027.7]$
Monsterfrog NLSA	0.109 ± 1.8	$[-174.445, 90.0984]$
Monsterfrog PN	0.141 ± 13.9	$[-496.404, 8768.75]$
Vase Phong	-0.00171 ± 0.0132	$[-0.350, 0.024]$
Vase Nagata $\epsilon = 0.06$	-0.02402 ± 0.7317	$[-94.715, 0.012]$
Vase NLSA	-0.00053 ± 0.0052	$[-0.049, 0.011]$
Vase PN	-0.00161 ± 0.0237	$[-0.380, 0.547]$
RoundedCube Phong	1.235 ± 0.690	$[0.123, 2.538]$
RoundedCube Nagata $\epsilon = 0$	-1.766 ± 4.84	$[-27.775, 1.653]$
RoundedCube NLSA	0.550 ± 0.314	$[0.091, 1.239]$
RoundedCube PN	0.468 ± 0.938	$[-1.093, 4.351]$
Sphere Phong	2.032 ± 0.046	$[1.847, 2.097]$
Sphere Nagata $\epsilon = 0$	0.972 ± 0.010	$[0.933, 0.985]$
Sphere NLSA	0.930 ± 0.010	$[0.887, 0.945]$
Sphere PN	0.950 ± 0.029	$[0.900, 1.047]$
Torus Phong	-1.503 ± 3.310	$[-9.476, 1.978]$
Torus Nagata $\epsilon = 0$	-0.412 ± 3.483	$[-4.027, 92.277]$
Torus NLSA	-0.672 ± 1.544	$[-3.901, 0.978]$
Torus PN	-0.638 ± 1.610	$[-5.612, 2.894]$

Table 5: Statistics on Gaussian curvature. The mean value for Gaussian curvature (mean \pm standard deviation) and the minimum and maximum value measured from the surfaces.

Monsterfrog, Vase and RoundedCube models, in terms of G^1 continuity performance, PN triangles perform best, followed by Phong tessellation (rank 2), and NLSA triangles (rank 3). For these models Nagata triangles turn out to be the worst, due to the use of the threshold ϵ to control the stability problem which produces linear edges reducing considerably the mean values of the cosine. But Nagata triangles are nearly optimal in the sphere and torus, confirming the results obtained in section 4.2. Looking at minimum and maximum values for these angles, note that almost in all situations there exist points in which adjacent patches have the same normals ($\cos = 1$), but

Name	Mean Angle Normals \pm std dev	Min,Max Angle Normals
Bunny Phong	0.959583 ± 0.0759962	$[-0.474661, 1]$
Bunny Nagata $\epsilon = 0.03$	0.813598 ± 0.470506	$[-1, 1]$
Bunny NLSA	0.949725 ± 0.0989627	$[0.0668511, 1]$
Bunny PN	0.983522 ± 0.0454926	$[0.0154815, 1]$
Monsterfrog Phong	0.949468 ± 0.128726	$[-0.999867, 1]$
Monsterfrog Nagata $\epsilon = 0.01$	0.822587 ± 0.950974	$[-1, 1]$
Monsterfrog NLSA	0.934331 ± 0.149359	$[-0.999826, 1]$
Monsterfrog PN	0.971483 ± 0.102529	$[-1, 1]$
Vase Phong	0.944903 ± 0.114099	$[0.0978981, 1]$
Vase Nagata $\epsilon = 0.06$	0.796925 ± 0.44401	$[-0.999999, 1]$
Vase NLSA	0.936506 ± 0.143869	$[-0.0187727, 1]$
Vase PN	0.987425 ± 0.0318718	$[0.684518, 1]$
RoundedCube Phong	0.938795 ± 0.0550764	$[0.796848, 0.980378]$
RoundedCube Nagata $\epsilon = 0$	0.67767 ± 0.457164	$[-1, 1]$
RoundedCube NLSA	0.880881 ± 0.153709	$[0.525784, 0.987624]$
RoundedCube PN	0.976669 ± 0.0390528	$[0.843874, 1]$
Sphere Phong	0.997072 ± 0.000765733	$[0.995723, 0.99874]$
Sphere Nagata $\epsilon = 0$	$0.999996 \pm 3.27602\text{e-}6$	$[0.999988, 1]$
Sphere NLSA	$0.999979 \pm 9.69038\text{e-}6$	$[0.999958, 1]$
Sphere PN	$0.999991 \pm 6.6563\text{e-}6$	$[0.999975, 1]$
Torus Phong	0.987918 ± 0.0157815	$[0.915295, 0.999999]$
Torus Nagata $\epsilon = 0$	0.997556 ± 0.0086036	$[0.938828, 1]$
Torus NLSA	0.994738 ± 0.0067909	$[0.969450, 1]$
Torus PN	0.998873 ± 0.0018468	$[0.989164, 1]$

Table 6: Statistics on the mean of angle cosines between normals of adjacent patches (mean \pm standard deviation), and the minimum and the maximum angle cosine.

also that there are unwanted situations, especially in Bunny, Monsterfrog and RoudedCube, where there are points in which the angles between the normals are π ($\cos = -1$ or nearby), meaning that there are points in which the two normals are parallel but pointing in opposite directions.

In the following we analyse each model in detail. The Bunny mesh is of moderate size (1000 triangles) with no degenerate normals, *i.e.*, all normals point into the positive halfspace defined by the triangle. It has a large fraction of inflection edges (58%), which results in patches of high curvature magnitude for the PN patches. For Nagata triangles a fairly large value

$\epsilon = 0.03$ has to be used, but still the resulting patches are exaggeratedly curved. Looking at the shaded images in the first row in Figure 12, no significative differences can be noticed between the four methods, but the highlight line plots in the second row confirm that PN triangles have less discontinuous normals. The Gaussian curvature plots in the third row, instead, show that, due to inflections introduced by the cubic patches on the inflection edges, the PN triangles curvature changes sign much more often than that of the three quadratic schemes, which therefore, have bigger regions of positive and negative curvature.

The Monsterfrog mesh is of larger size (2584 triangles) with a fraction of 52% inflection edges. Unlike the Bunny mesh, there are some degenerate normals at the frog’s teeth and in the tail. It is actually in these locations that the tangent-plane continuity is violated the most, see Table 6.

On this fine base mesh, the difference between all the schemes gets smaller with the exception of Nagata triangles, which need to cope with the stability problems elucidated above in Section 3.2.

The Vase model in Figure 14 is a modified cylinder, where the circle is deformed into a star shape in the lower part. The original quadrangle mesh was triangulated by introducing one of the diagonals. For this special configuration, all of the degree-2 schemes generate fairly planar patches. Nagata triangles require the largest threshold $\epsilon = 0.06$ among all the models considered. PN triangles generate typical curvature patterns for the two triangles adjacent to the diagonal edge: one has positive and the other one has negative Gaussian curvature in vicinity of the edge. In Figure 15, we have reproduced the different behaviors in the vicinity of a diagonal curve depending on the normals configuration. A characterization of the qualitative shape of a planar cubic Bézier curve is described in [23, 24]. Following this characterization, only the cases 1. (no inflection point) and 3. (one inflection point) can occur for PN triangles due to the way the edge control points are constructed by the projection of interior points $\bar{\mathbf{b}}_{210} = (2\mathbf{p}_0 + \mathbf{p}_1)/3$ and $\bar{\mathbf{b}}_{120} = (\mathbf{p}_0 + 2\mathbf{p}_1)/3$ into the tangent planes. When PN triangles have to deal with configurations like in Figure 15 left or center, they can produce artifacts or ondulations due to these inflections. For these configurations a solution could be the use of PN-quads [25], although switching of triangles to quads is computationally not very convenient and advantageous.

Finally, the RoundedCube mesh has 56 triangles of large size, which also results in patches of large size. The cube corners are cut and the remaining faces are triangulated into a fan with respect to an added center vertex.

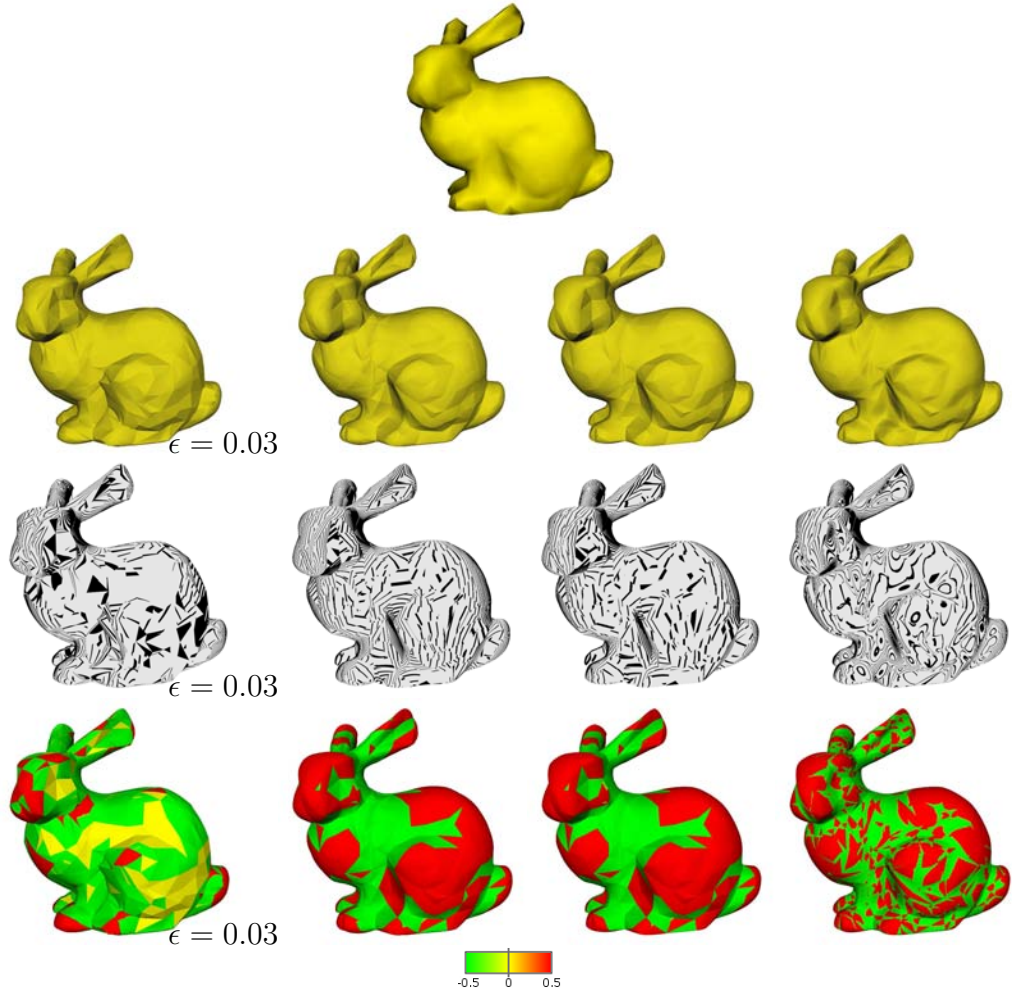


Figure 12: Bunny in columns from left to right: Nagata, Phong, NLSA, PN triangles. First row: shaded planar mesh; second row: shaded surfaces; third row: highlight lines; fourth row: Gaussian curvature.

For this highly symmetric base mesh the differences between the surfaces constructed by the four schemes are clearly visible in Figure 16. The PN triangles reproduce the tangent planes and also give good approximate tangent-plane continuity along the triangle edges. NLSA and Phong triangles are not enough curved and do not reproduce the tangent planes, which is evident on this mesh. Nagata triangles, on the contrary, reproduce exactly all the

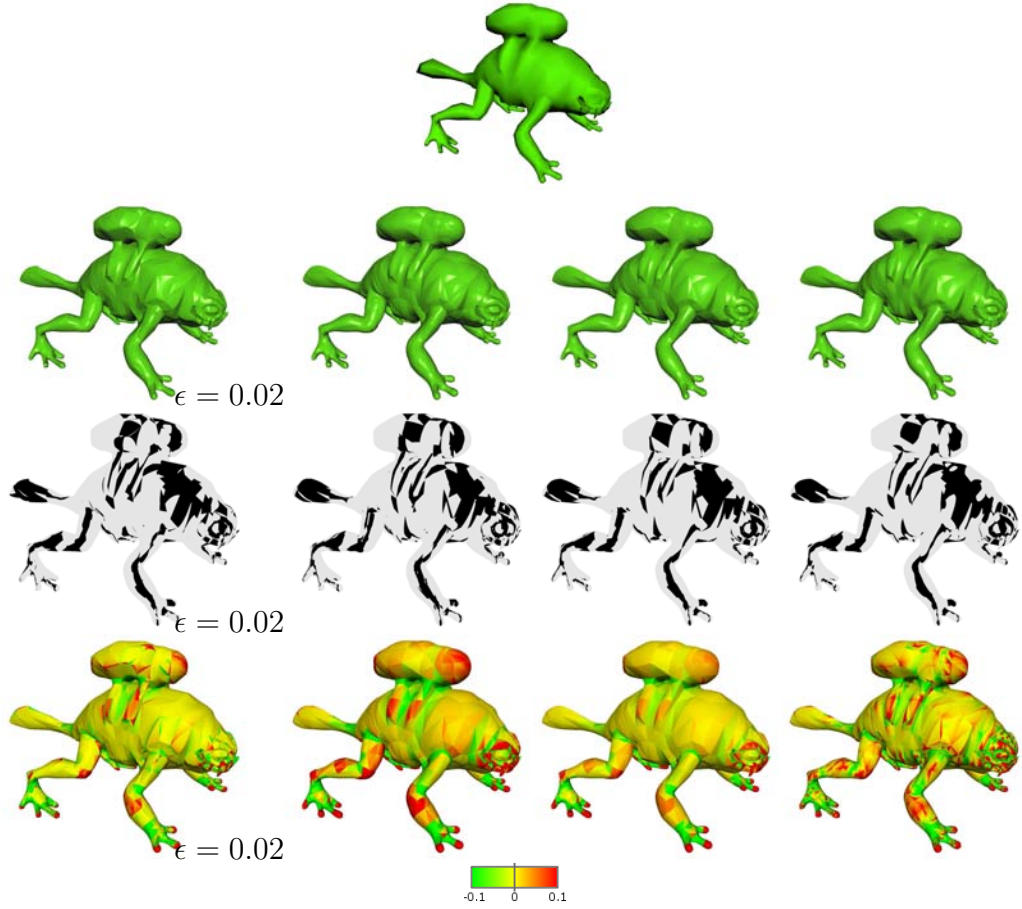


Figure 13: Monsterfrog in columns from left to right: Nagata, Phong, NLSA, PN triangles. First row: shaded planar mesh; second row: shaded surfaces; third row: highlight lines; fourth row: Gaussian curvature.

tangent planes in the corners but connect patches with curvature of different sign. The corner caps have positive Gaussian curvature, the others have negative Gaussian curvature, which results in a bad approximate tangent-plane continuity along edges.

5. Separate normal interpolation patch

It is evident from the examples in the last section that C^0 continuity is not enough for providing the desired smooth appearance in applications.

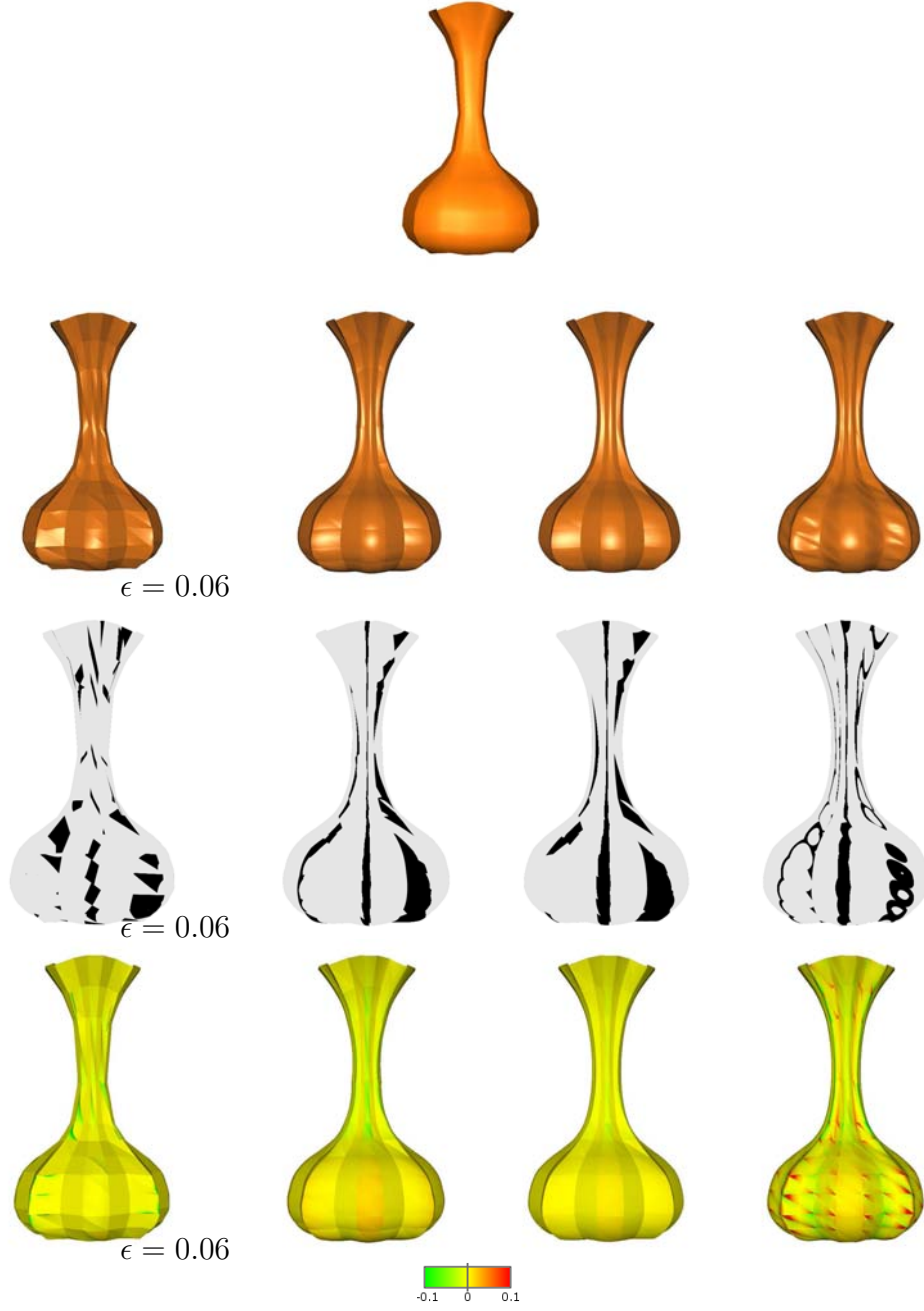


Figure 14: Vase in columns from left to right: Nagata, Phong, NLSA, PN triangles. First row: shaded planar mesh; second row: shaded surfaces; third row: highlight lines; fourth row: Gaussian curvature.

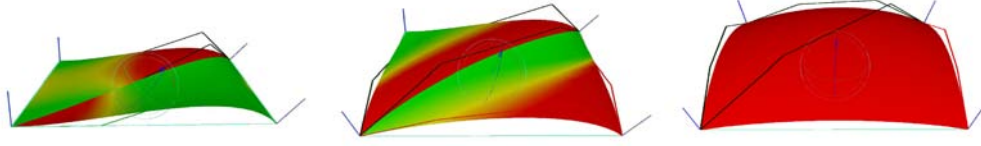


Figure 15: Different behaviors in the vicinity of a diagonal curve depending on the normals configuration for PN triangles. Left: the diagonal has an inflection. Center and Right: two different configurations with a convex diagonal curve, but only the patches in the right are convex.

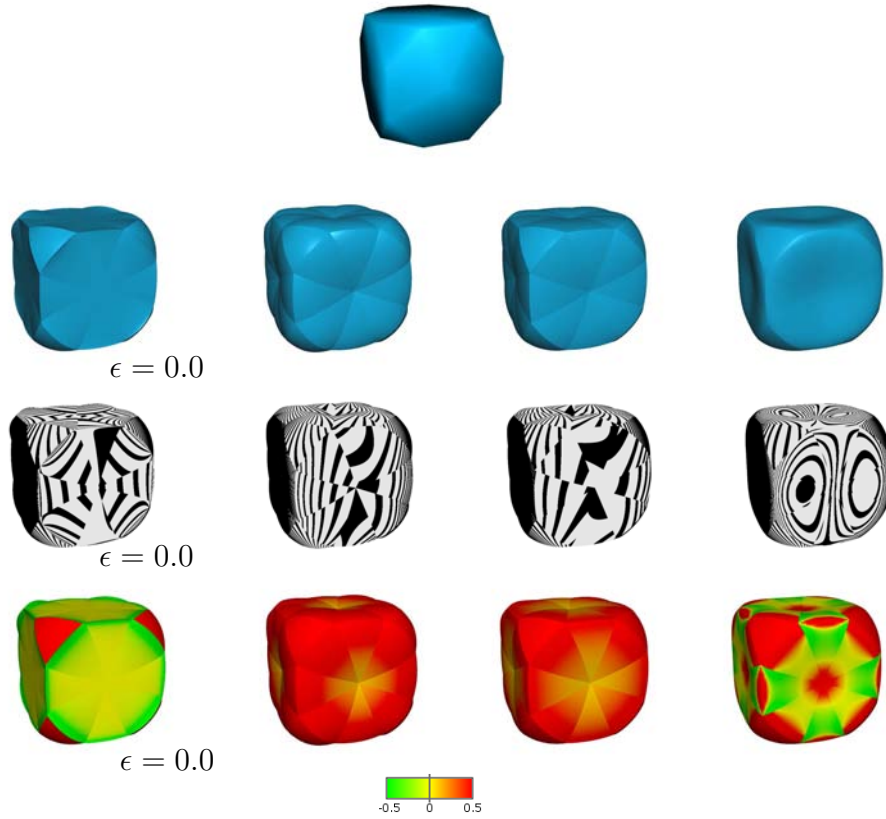


Figure 16: RoundedCube in columns from left to right: Nagata, Phong, NLSA, PN triangles. First row: shaded planar mesh; second row: shaded surfaces; third row: highlight lines; fourth row: Gaussian curvature.

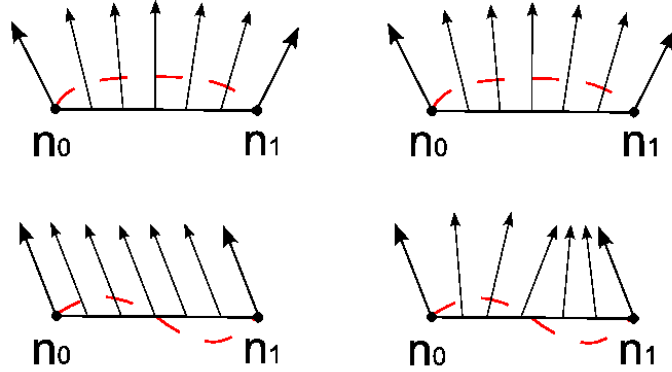


Figure 17: Comparison between the configurations obtained from a linear and a quadratic normal variation patch in two different cases. The dashed curve indicates the profile of the surfaces that should be simulated. Left: linearly varying normals. Right: quadratically varying normals.

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. We point out that in this way the surface is simply enhanced in its visualization and not at all smoothed in its geometry.

When an independent *linear normal variation patch* is used in a shading process, the value of the normal at the parameter point (u, v, w) is simply computed as the normalized average of the normal values at the vertices of the triangle:

$$\mathbf{n}(u, v, w) = \frac{u\mathbf{n}_0 + v\mathbf{n}_1 + w\mathbf{n}_2}{\|u\mathbf{n}_0 + v\mathbf{n}_1 + w\mathbf{n}_2\|}.$$

One problem with linearly varying normals, is that such a patch ignores inflections in the geometry, as illustrated for instance in Figure 17 (bottom-left). More details can be found in [21].

As a solution to this problem the option of a *quadratic normal variation patch* is proposed by the authors of PN triangles [11] who took the idea from [21]. A quadratic function $\mathbf{n}(u, v, w)$ is used to compute normals at the

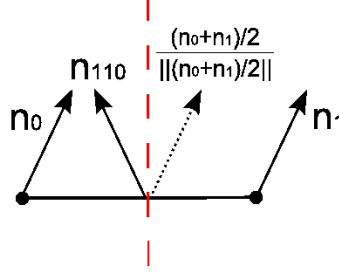


Figure 18: Construction of the mid-edge normal coefficient \mathbf{n}_{110} for quadratically varying normals: the average of the end-normals is reflected across the plane perpendicular to the edge.

evaluation points (u, v, w) :

$$\mathbf{n}(u, v, w) = \sum_{i+j+k=2} \mathbf{n}_{ijk} u^i v^j w^k, \quad w = 1 - u - v, \quad u, v, w \geq 0.$$

The values of $\mathbf{n}(u, v, w)$ are then normalized and passed on to the shading process. The choice of the mid-control normal from [11] is sketched in Figure 18.

In Figure 19 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

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

In this case we can analyse the real surface geometry as in the models of the last section. The second and third row show the difference between linear and quadratic normals, which give the impression of higher continuity in the interior. Here the visual appearance of all the schemes is very similar and differences in the geometry are only visible at the silhouette. Linear normal patches result in more flat-appearing surfaces. For instance, compare the transition from the mouth region towards the rest of the model.

6. Conclusions

In this paper we made a comparison of local parametric C^0 curved shape schemes, based on a reformulation of their original constructions in terms of

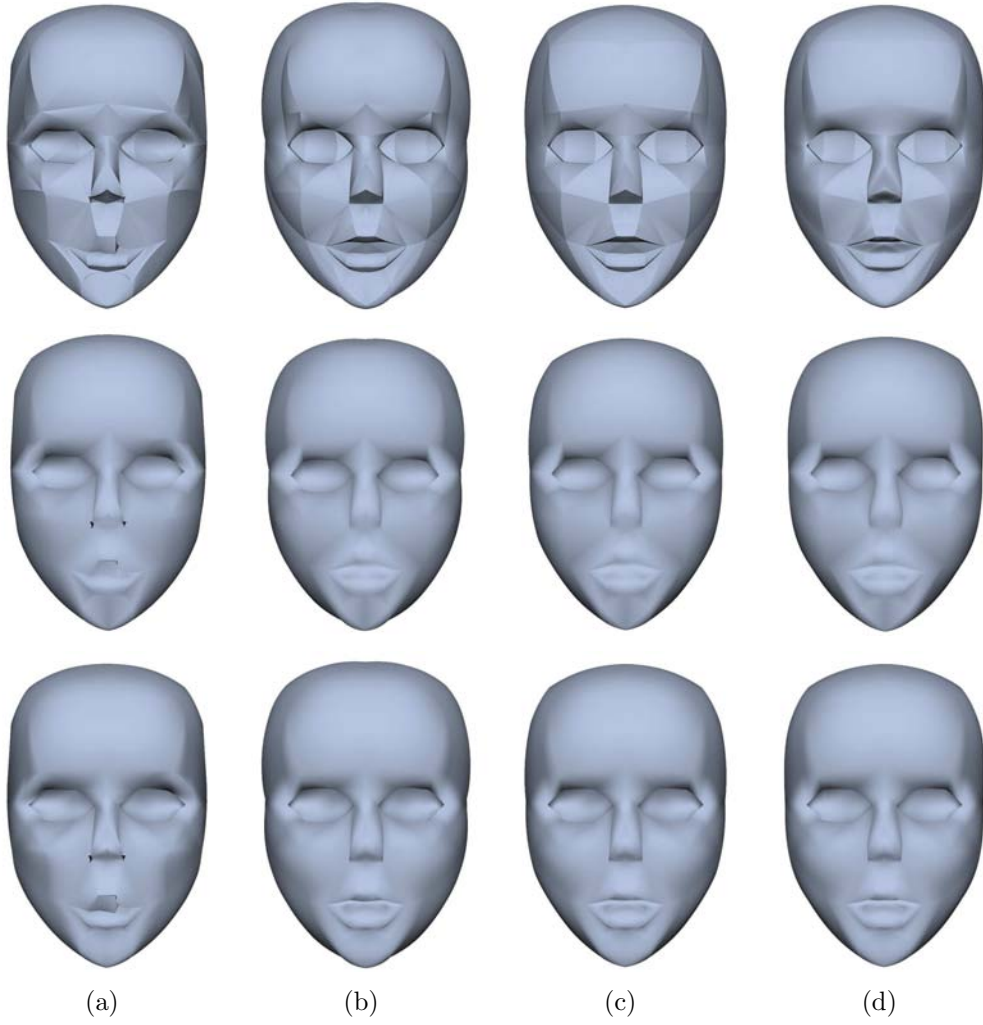


Figure 19: A triangular head model rendered with different schemes and normals: (a) Nagata triangles ($\epsilon = 0.04$), (b) Phong tessellation, (c) NLSA triangles and (d) PN triangles. First row: Analytic normals. Second row: Linear normals. Third row: Quadratic normals.

polynomial Bézier triangles. The main emphasis of this comparison is to look at the surface quality of the different schemes available. Related techniques to improve the surface quality or the silhouette only (as the one in [26]) were not taken into account, except for the use of a separated normal interpolation patch briefly treated in section 5. The comparison comprises 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.

Our reformulation in terms of triangular Bézier patches allows a geometric interpretation of the control points defining the patches. In particular, for Nagata triangles our geometric interpretation gives a very clear explanation of the stability problem explained in section 3. Moreover, we found that a NLSA triangles patch is equal to a Phong tessellation patch with $\alpha = \frac{1}{2}$ and this result leads to two important observations. The first is that the Phong tessellation patch with $\alpha = \frac{1}{2}$ minimizes the near least square acceleration. This non-obvious result is readily seen from the NLSA triangles construction. The second observation is related to computational costs. The cost of the construction of NLSA triangles can be highly reduced by computing it using the Phong tessellation method.

In fact, Phong tessellation is extremely less expensive in terms of computational costs, and in general PN triangles are more expensive in the time required for the tessellation both on the CPU and GPU, due to their higher degree. In particular, for the GPU evaluation, the schemes with small total costs (Phong tessellation and Nagata triangles) are advantageous. PN triangles are slightly slower, and NLSA triangles are the slowest.

From the geometric analysis we made, it also follows that PN triangles and Nagata triangles are the only two schemes that exactly reproduce the tangent planes given by the normals in input. This is an advantage in the approximation behaviour of these two schemes with respect to a known surface, like the sphere and the torus treated in section 4.2. Nagata has in general the best approximation behaviour, followed by PN triangles and NLSA triangles methods which behave similarly, whereas Phong tessellation has the worst approximation behaviour.

We have performed a wide range of experiments and statistics involving meshes with a low triangle count. Analysis on the Gaussian curvature values helped to investigate regularity of the surfaces in detail. By comparing curvature values for NLSA triangles and Phong tessellation we found that the shape parameter, in a certain sense, controls the curvature of the constructed surface. Moreover, we had a confirmation that the stability problem

in Nagata triangles deeply affects the resulting overall surface shape.

For real-world models the differences in terms of approximate G^1 continuity between all the schemes are quite small, but our results permit the following classification: PN triangles have the smallest angles of adjacent tangent-planes along edges to neighbours, followed by Phong tessellation and NLSA triangles methods. Nagata's method suffers from stability problems and performs worst for these real word examples. But for known analytic surfaces, such as the sphere and the torus, Nagata outperforms the other methods confirming its best approximation behaviour.

Based on the above comparison results a user thus has the means of deciding which method best suites his/her specific application, in terms of criteria such as computational cost, analytic surface reproduction, Gaussian curvature behaviour and approximate G^1 continuity.

7. Acknowledgements

Maria Boschioli acknowledges the financial support in form of travel allowances from French-Italian University within the Da Vinci program (<http://www.universite-franco-italienne.org/>).

References

- [1] Collins, S.. Kinematics, dynamics, biomechanics: Evolution of autonomy in game animation. *Computer Graphics Forum* 2005;24(3).
- [2] Baran, I., Popović, J.. Automatic rigging and animation of 3D characters. *ACM Trans Graph* 2007;26(3):72.1–72.8.
- [3] Kautz, J.. Hardware lighting and shading: a survey. *Computer Graphics Forum* 2004;23(1):85–112.
- [4] Szirmay-Kalos, L., Umenhoffer, T., Patow, G., Szécsi, L., Sbert, M.. Specular effects on the GPU: State of the art. *Computer Graphics Forum* 2009;28(6):1586–1617.
- [5] Farin, G.. *Curves and surfaces for CAGD: a practical guide*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.; 5th ed.; 2002.

- [6] 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: IEEE Computer Society; 1998, p. 72–78.
- [7] 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: IEEE Computer Society; 1997, p. 3–12.
- [8] Liu, Y., Mann, S.. Approximate continuity for parametric Bézier patches. In: SPM '07: Proceedings of the 2007 ACM symposium on Solid and physical modeling. New York, NY, USA: ACM; 2007, p. 315–321.
- [9] Liu, Y., Mann, S.. Parametric triangular Bézier surface interpolation with approximate continuity. In: SPM '08: Proceedings of the 2008 ACM symposium on Solid and physical modeling. New York, NY, USA: ACM; 2008, p. 381–387.
- [10] Walton, D.J., Meek, D.S.. A triangular G^1 patch from boundary curves. *Computer-Aided Design* 1996;28(2):113–123.
- [11] 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: ACM; 2001, p. 159–166.
- [12] Farin, G.. Smooth interpolation to scattered 3D data. In: Barnhill, R.E., Boehm, W., editors. *Surfaces in Computer-Aided Geometric Design*. North-Holland; 1983, p. 43–63.
- [13] Mao, Z., Ma, L., Tan, W.. A modified Nielson's side-vertex triangular mesh interpolation scheme. In: *Computational Science and Its Applications ICCSA 2005*; vol. 3480. Springer Berlin / Heidelberg; 2005, p. 776–785.
- [14] Nielson, G.M.. A transfinite, visually continuous, triangular interpolant. In: Farin, G., editor. *Geometric modeling*. Philadelphia, PA: SIAM; 1987, p. 235–246.

- [15] Boubekur, T., Reuter, P., Schlick, C.. Scalar tagged PN triangles. In: EUROGRAPHICS 2005 (Short Papers). Eurographics; 2005,.
- [16] Dyken, C., Reimers, M., Seland, J.. Real-time GPU silhouette refinement using adaptively blended Bézier patches. Computer Graphics Forum 2008;27(1):1–12.
- [17] Boubekur, T., Alexa, M.. Phong tessellation. In: SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers. New York, NY, USA; 2008, p. 1–5.
- [18] Nagata, T.. Simple local interpolation of surfaces using normal vectors. Computer Aided Geometric Design 2005;22(4):327–347.
- [19] 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, p. 17–22.
- [20] Fünzig, C., Müller, K., Hansford, D., Farin, G.. PNG1 triangles for tangent plane continuous surfaces on the GPU. In: GI '08: Proceedings of graphics interface 2008. Toronto, Ont., Canada: Canadian Information Processing Society; 2008, p. 219–226.
- [21] Van Overveld, C.W.A.M., Wyvill, B.. Phong normal interpolation revisited. ACM Trans Graph 1997;16(4):397–419.
- [22] Hahmann, S., Belyaev, A., Buse, L., Elber, G., Mourrain, B., Roessl, C.. Shape interrogation - a state of the art. In: De Floriani, L., Spagnuolo, M., editors. Shape Analysis and Structuring. Springer; 2008, p. 1–52.
- [23] Albrecht, G., Bécar, J., Xiang, X.. Géométrie des points d'inflexion et des singularités d'une cubique rationnelle. Revue Electronique Francophone d'Informatique Graphique 2008;2(1):33–46.
- [24] Fünzig, C., Thomin, P., Albrecht, G.. Haptic manipulation of rational parametric planar cubics using shape constraints. In: SAC '10: Proceedings of the 2010 ACM Symposium on Applied Computing. New York, NY, USA: ACM; 2010, p. 1253–1257.

- [25] Peters, J.. PN-quads. Tech. Rep. 2008-421; Dept. CISE, University of Florida; 2008.
- [26] 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 1999;.