Flexible G1 Interpolation of Quad Meshes

Georges-Pierre Bonneau, Stefanie Hahmann

HAL Id: hal-01064552
https://hal.inria.fr/hal-01064552

Flexible G^1 interpolation of quad meshes

Georges-Pierre Bonneau , Stefanie Hahmann

University of Grenoble, Laboratoire Jean Kuntzmann, INRIA, Grenoble

Abstract

Transforming an arbitrary mesh into a smooth G^1 surface has been the subject of intensive research works. To get a visual pleasing shape without any imperfection even in the presence of extraordinary mesh vertices is still a challenging problem in particular when interpolation of the mesh vertices is required. We present a new local method, which produces visually smooth shapes while solving the interpolation problem. It consists of combining low degree biquartic Bézier patches with minimum number of pieces per mesh face, assembled together with G^1-continuity. All surface control points are given explicitly. The construction is local and free of zero-twists. We further show that within this economical class of surfaces it is however possible to derive a sufficient number of meaningful degrees of freedom so that standard optimization techniques result in high quality surfaces.

1 Introduction

Quad meshes are a popular surface representation in Computer Graphics. In contrast to triangle meshes they are better suitable to represent shapes with symmetry axes. Since most manufactured objects have at least one symmetry axis, it is more natural to represent these objects by quads with the edges following the symmetry axis instead of using triangles. For the same reason quad meshes are preferably used by professional designers for creating 3D content such as character animations.

Converting an arbitrary manifold quad mesh into a smooth piecewise polynomial surface is the problem the present paper deals with. Whereas subdivision surfaces [3, 13] can generate a smooth shape with only a few subdivision steps from a coarse mesh, parametric polynomial spline surfaces have the following three main advantages over subdivision surfaces. First, they provide an explicit polynomial parameterization enabling exact evaluation of surface points and any higher order differential quantities such as tangent, bi-tangent, normal, curvature or derivative of curvature with a finite number of patches. Second, they are compatible with existing CAD standards. All classical modeling operations such as trimming, intersection, blending and Boolean operations can thereby be performed directly with our resulting surfaces. Third, GPU-accelerated tessellation units are based on parametric patches and thus allow for real-time rendering of animated polynomial splines surfaces. Efficient subdivision surface modelers [34] therefore convert a subdivision surface finally into a finite set of parametric patches for rendering purposes [27].

Smooth spline surfaces defined on arbitrary quad meshes are also a powerful alternative to singularly parameterized tensor product surfaces since they combine the advantages of both, the arbitrary topology of quad meshes and the smoothness of the tensor product patches. Herein, tensor product piecewise polynomial patches, so-called *macro-patches*, are assembled with tangent plane continuity, in one-to-one correspondence to the mesh faces. In this way they are capable to represent manifold shapes of arbitrary topological type. Classical tensor product surfaces on the contrary are restricted to meshes with vertices of valence four.

In the present paper we propose a new approach for inter-

polating the vertices of a given arbitrary manifold quad mesh. Even though the existing literature on this topic is large, only a few methods are dedicated specifically to the interpolation problem. Most approaches generate a smooth surface from the given mesh by approximating it. When it comes to interpolation, the lack of sufficient degrees of freedom often results in unwanted undulations. The problem can be solved by either increasing the polynomial degrees of the patches or the number of patches composing a macro-patch.

Our goal is to solve the interpolation problem by staying within the class of low degree Bézier patches, biquartic in this case. But instead of increasing the number of patches per macro-patch in order to improve visual quality, we show that it is possible to provide sufficient meaningful degrees of freedom with a minimal number of patches so that for a first time an interpolating biquartic surface of excellent visual quality can be achieved for arbitrary meshes. We demonstrate the efficiency of our method with various examples including large quad meshes and complex shapes. Figures 1 and 2 show examples of results with input quad meshes of genus topological types 0 and 2 including extraordinary vertices of valence 3, 5 and 6.

Figure 1: Vase and Apple meshes, with genus 2 and 0 topological types respectively. The input quad meshes have extraordinary vertices of valence 3, 5, and 6. Left: our interpolating piecewise quartic G^1 surfaces rendered with Blender. The smoothness of the reflections and highlights that cross several quartic patches underlines the smoothness of the surface across quartic patches. Right: input quad meshes.

1.1 Related Works

The present paper solves the problem of defining a G^1-continuous spline surface *interpolating* the vertices of an arbitrary quad mesh with low degree polynomial tensor product patches while producing shapes of high visual quality.

We further aim to solve the *interpolation* problem in a completely general manner. Several special interpolation methods exist [41, 38, 12, 37], but they all make either restrictions on the mesh topology (vertices of valence 3,4,5 only) or on the input data, and lead to higher degree patches. [11] constructs 2×2 bicubic macro-patches per quad, but the method is valid only for very special cases, the construction fails in the general setting as

Figure 2: Irregular quad meshes interpolated by a G^1-continuous surface composed of biquartic Bézier patches. Each object is composed of nine independent genus 0 meshes representing the body and the face elements separately.

pointed out in [33]. In [33] it is also shown that 4 is the minimal possible degree for localized G^1 constructions with 2×2 macro-patches. The simplest structure for general bicubic G^1-surfaces is a 3×3 macro-patch such as proposed in [7]. An important achievement here is that for regular quads, where all vertices are of valence 4, the standard C^2 plines are yield. Theoretically, the method can be used for interpolation. However, no example of interpolation with sufficiently big quad meshes are shown to assess if interpolation is free of unwanted surface behavior. [19] uses biquintic patches for adaptive fitting of triangle meshes, but solves the G^1 conditions with a global linear system. [39] proposes biquintic G^1 B-splines with many patches. Further methods [23, 4] deviate from standard polynomial representation. [29, 20, 14] solve the similar but more constraint problem of interpolation of a mesh of boundary curves and thus need to make severe restrictions on the input data. The visual smoothness of interpolatory subdivision surfaces is not satisfactory in the neighborhood of extraordinary points even with improved schemes [6, 42, 16, 18]. They generally do not provide an explicit polynomial representation. Another class of schemes produces visually smooth surfaces without having the corresponding mathematical smoothness. These approximate G^1 schemes [21] compute C^0 surfaces in order to overcome the mathematical restrictive G^1 constraints. They use simple heuristics to reduce the discontinuity. It is obvious that fewer constraints give degrees of freedom for shape design. It would be interesting to have convergent G^1 surfaces or even practical G^2 schemes, but this is not the purpose of the present paper, even though it is an actual research issue [15, 32].

A precursor of many of the surface splines mentioned above is the biquartic spline in [30]. Herein explicit expressions of all Bézier control points are computed by local averaging of neighboring face center points. The macro-patches are constructed in one-to-one correspondence to a quad sub-mesh which is previously generated by applying the mid-point refinement technique to any arbitrary input mesh. The scheme however produces zero-twists, which are known to produce poor shape quality. Interpolation is theoretically possible, but produces unwanted undula-

tions in practice. In contrast to the schemes in [30, 7], which mimic Catmull-Clark subdivision surfaces, our scheme is intentionally an interpolation scheme. The interpolation property in our method is not a by-product, it is its main characteristic. All efforts in developing the scheme in the subsequent sections focus on how to maximize the number of degrees of freedom in particular for the patch boundary curves in order to get perfect visually pleasing shapes.

1.2 Contributions and Overview

This paper presents a novel method to interpolate the vertices of a given arbitrary quad mesh with a fair surface of overall G^1-continuity. Vertex normals can either be explicitly prescribed or may be derived from neighboring mesh vertices. The surface construction is based on a 1-to-4 split of the parameter domain for each surface patch (see Section 2). It is shown in this paper that thanks to the 4-split technique, the resulting surface is of low bi-degree 4 and locally defined (see Section 4). We give explicit simple linear formulas for computing all control points of the surface. Our method provides many meaningful independent degrees of freedom for local shape control. The resulting surface is of high visual quality. It is free of zero-twists and avoids unwanted undulations although the mesh is interpolated.

The interpolation scheme applies to any arbitrary 2-manifold surface quad mesh. The scheme is completely general in the sense that it does not make any restriction of the valence of the vertices, i.e. no restriction of the number of patches joining around a common vertex.

Furthermore, we explain how shape optimization can be performed on this new surface model (see Section 5). Finally we demonstrate the efficiency of our method by analyzing the quality of the resulting shapes with isophotes and comparing with previous analogous works (see Section 6).
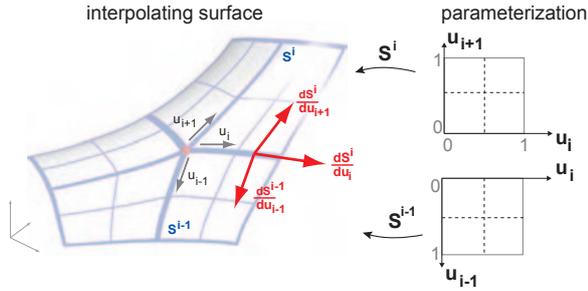
Figure 3: 4-split of the parameter domain. Parameterization of the surface patches around a mesh vertex. Always four Bézier patches are constructed in correspondence to each mesh face, called macro-patch. The benefits of the 4-split are mainly that the resulting surface is local and of low degree while interpolating the mesh vertices.

2 4-splitting

2.1 Notations

Let $\mathcal{M}$ denote the input mesh. $\mathcal{M}$ is a set of vertices, edges and quadrilateral faces that describes a 2-manifold oriented surface in $\mathbf{R}^3$ with or without boundary and of arbitrary topological genus. The mesh faces do not need to be planar, and there is no restriction on the number of faces meeting at a vertex. $\mathcal{M}$ is called an *arbitrary quad mesh*. The number of edges incident in a vertex is referred to as *valence of a vertex*. The set of vertices sharing the edges incident to a vertex is called *vertex neighborhood*.

2.2 4-split

The problem we address is the computation of the Bézier control points of a biquartic G^1-continuous surface $\mathscr{S}$ interpolating the vertices of $\mathcal{M}$. Since we want to be able to model a surface of arbitrary genus each surface patch can be seen as the image of one quadrilateral domain. Thus instead of considering a classical map from a regular, chess board planar domain onto the surface, we consider many maps of one unit quadrilateral domain onto different patches of the surface, see Figure 3.

In our setting, each map associates in one-to-one correspondence a collection of four biquartic Bézier patches to one quadrilateral parameter domain. In fact, we subdivide the parameter domain into 4 sub-domains as illustrated in Figure 3. The collection of four patches is referred to as *macro-patch*. The macro-patches join together with G^1 continuity. They are C^1 continuous inside. The four corner vertices of a macro-patch interpolate the four mesh vertices of the corresponding mesh facet.

This 4-split is crucial since it enables the surface scheme to have many important properties:

- the surface scheme is local,
- the mesh vertices and (optionally) the normals are interpolated,
- explicit formulas for all control points are given,
- several degrees of freedom for each input vertex, edge and macro-patch are available.

2.3 Macro-patch

When constructing a patchwork of polynomial patches with G^1-continuity special attention has to be paid to what happens at the mesh vertices. Let us consider a mesh vertex $\mathbf{v} \in \mathbf{R}^3$ of valence n, and its vertex neighborhood $\mathbf{v}_1, \ldots, \mathbf{v}_n$ ordered in a counterclockwise sense. The macro-patches around the vertex are called $S^1, \ldots, S^n$. The super- and subscripts $i = 1, \ldots, n$ are taken modulo

n. Let $u_i \in [0,1]$ be the parameter corresponding to the curve between $\mathbf{v}$ and $\mathbf{v}_i$, the surface macro-patch S^i is then parameterized as illustrated in Figure 3.

3 Simplified G^1 conditions

The interpolating surface is required to be polynomial and tangent plane continuous, i.e. G^1-continuous. This means that the surface is C^∞ everywhere except at the boundaries between patches where it has continuously varying tangent planes.

Let S^i and S^{i-1} be two adjacent macro-patches parameterized as in Figure 3, with a common boundary curve $S^i(u_i,0) = S^{i-1}(0,u_i)$. We require the following G^1-continuity condition to hold between S^i and S^{i-1}:

$$\Phi_i(u_i)\,\frac{\partial S^i}{\partial u_i}(u_i,0) = \frac{1}{2}\,\frac{\partial S^i}{\partial u_{i+1}}(u_i,0) + \frac{1}{2}\,\frac{\partial S^{i-1}}{\partial u_{i-1}}(0,u_i),$$

$$u_i \in [0,1], \tag{1}$$

where Φ_i is a scalar function.

These simplified G^1 conditions are generally used (see [22, 30, 10]) in order to keep the scheme of as low degree as possible. The degree of the surface is related to the degree of the scalar function Φ_i since both sides of equation (1) need to be of same polynomial degree. In order to keep the degree of the patches low, the degree of Φ_i has to be low too. Ideally this would mean to take Φ_i linear. But then the surface construction would not be local, since $\Phi_i'(0)$ would depend on the value $\Phi_i(1)$. This means, it would depend on the valence n_i of the neighboring vertex $\mathbf{v}_i$. A local definition is important since the modification of one mesh vertex should not modify the whole surface but only the macro-patches around. The outside boundary of this group of macro-patches should not be affected by this modification. Therefore all computations around a vertex $\mathbf{v}$ should not depend on information related to the neighboring vertices $\mathbf{v}_i$. This is the reason why Φ_i has been chosen quadratic in [22] for the construction of triangular patches and quadratic in [30] for the construction of biquartic tensor product patches. In [10] it has been shown that piecewise linear Φ_i-functions can be used for triangular patches. This is not possible for the setting here.

In our paper, and similar to [31], we choose Φ_i to be the piecewise quadratic function defined by:

$$\Phi_i(u_i) \equiv \begin{cases} \cos\left(\frac{2\pi}{n}\right) B_0^2(2u_i), & u_i \in [0, \frac{1}{2}] \\ -\cos\left(\frac{2\pi}{n_i}\right) B_2^2(2u_i), & u_i \in [\frac{1}{2}, 1]. \end{cases} \tag{2}$$

Let us denote the coefficients by

$$\Phi^0 := \cos\left(\frac{2\pi}{n}\right) \qquad \text{and} \qquad \Phi^1 := \cos\left(\frac{2\pi}{n_i}\right).$$

4 Solving G^1-continuous interpolation

In this section we derive the explicit formulas for computing the control points of the G^1 continuous piecewise quartic surface interpolating the input quad mesh. We begin in Section 4.1 by examining the constraints on the control points of the boundary curves between adjacent macro-patches. Then in Section 4.2 we express the simplified G^1-conditions (1) in terms of Bézier control points. The two Sections 4.3 and 4.4 give an explicit solution of these conditions. As a result, the first two rows of control points are computed along all macro-patch boundaries. Eventually, Section 4.5 explains how to compute the remaining control points inside each macro-patch.

A key point of our approach consists in constantly working out all possible degrees of freedom instead of fixing them to specific values as it is usually done. This is essential in order to get a visually smooth shape when interpolating a mesh. In the three Sections 4.3, 4.4 and 4.5 we thus explicitly state all DOFs (degrees of freedom) used in the computation of the control points. How to set-up these DOF is explained in Section 5.

Figure 4 illustrates our notations for the three rows of control points involved in the G^1 continuity along a common boundary between two macro-patches. Due to the 4-split, these control points belong to four quartic Bézier patches. The $\sim$ is used for the two Bézier patches on the right. The b_i denote the control points of the common boundary curve, the c_i (resp. d_i) denote control points of the top (resp. bottom) Bézier patches.
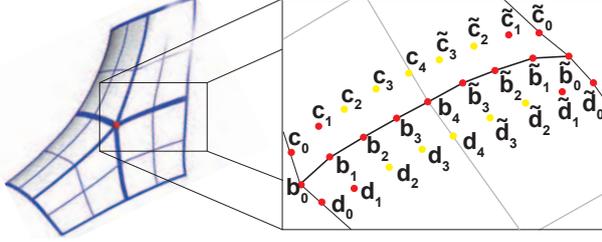

Figure 4: All control points involved in the G^1-conditions between two macro-patches.

4.1 Common boundary curve

Since S^i and S^{i-1} are composed of biquartic Bézier patches, the cross-boundary derivatives $\frac{\partial S^i}{\partial u_{i+1}}(u_i,0)$ and $\frac{\partial S^{i-1}}{\partial u_{i-1}}(0,u_i)$ are piecewise quartic polynomials in u_i. Given the degree of Φ_i, a solution to (1) can be found only if the common boundary curve $S^i(u_i,0) = S^{i-1}(0,u_i)$ is a piecewise cubic Bézier curve.

Furthermore we require C^1 continuity inside the macro-patches which implies that the left and right segments of the common boundary curve must joint with C^1 continuity. Since there exists a unique piecewise C^1 cubic curve interpolating given position, first and second order derivatives at both endpoints, the boundary curve must be equal to this curve. By degree elevation of this piecewise cubic curve, it follows that the control points $b_3, b_4 = \widetilde{b}_4$ and $\widetilde{b}_3$ can be computed from b_0, b_1, b_2 and $\widetilde{b}_0, \widetilde{b}_1, \widetilde{b}_2$ by:

$$
\begin{aligned}
b_4 = \widetilde{b}_4 \quad &= \tfrac{1}{2}(2b_2 - \tfrac{4}{3}b_1 + \tfrac{1}{3}b_0 + 2\widetilde{b}_2 - \tfrac{4}{3}\widetilde{b}_1 + \tfrac{1}{3}\widetilde{b}_0) \\
b_3 \quad &= \tfrac{1}{4}(b_4 + 6b_2 - 4b_1 + b_0) \\
\widetilde{b}_3 \quad &= \tfrac{1}{4}(b_4 + 6\widetilde{b}_2 - 4\widetilde{b}_1 + \widetilde{b}_0).
\end{aligned}
\tag{3}
$$

How to compute the control points b_0, b_1, b_2 and $\widetilde{b}_0, \widetilde{b}_1, \widetilde{b}_2$ is explained in Section 4.3.

4.2 G^1-conditions on control points

In this section we express the simplified G^1-conditions (1) in terms of the Bézier control points.

The three control points of the quadratic scalar function Φ_i of the left piece of the curve corresponding to the parameter domain $[0, \frac{1}{2}]$ are $(\Phi_0, 0, 0)$. From Section 4.1 we know that the derivative $\frac{\partial S^i}{\partial u_i}(u_i,0)$ along the boundary curve is a quadratic polynomial. Its three control points are $8(b_1 - b_0), 4b_0 - 16b_1 + 12b_2, 8(b_4 - b_3)$. Multiplying $\Phi_i(u_i)$ by $\frac{\partial S^i}{\partial u_i}(u_i,0)$ leads to a quartic polynomial whose five control points are

$$
8\Phi^0(b_1 - b_0), 2\Phi^0(b_0 - 4b_1 + 3b_2), \tfrac{4}{3}\Phi^0(b_4 - b_3), 0, 0.
$$

On the other hand, the five control points of $\frac{1}{2} \frac{\partial S^i}{\partial u_{i+1}}(u_i,0) + \frac{1}{2} \frac{\partial S^{i-1}}{\partial u_{i-1}}(0,u_i)$ for $u_i \in [0, \frac{1}{2}]$ are $4(c_i - 2b_i + d_i)$ for $i = 0, \dots, 4$. Therefore the simplified G^1-conditions are given by:

$$
8\Phi^0(b_1 - b_0) = 4(c_0 - 2b_0 + d_0) \tag{4}
$$

$$
2\Phi^0(b_0 - 4b_1 + 3b_2) = 4(c_1 - 2b_1 + d_1) \tag{5}
$$

$$
\tfrac{4}{3}\Phi^0(b_4 - b_3) = 4(c_2 - 2b_2 + d_2) \tag{6}
$$

$$
0 = 4(c_3 - 2b_3 + d_3) \tag{7}
$$

$$
0 = 4(c_4 - 2b_4 + d_4) \tag{8}
$$

and by the analogous five conditions for the right piece of curve with $\sim$ control points and Φ^0 replaced by Φ^1.

The following three sections give explicit formulas of all control points involved in these G^1-conditions and the available DOF.

4.3 Control points around input vertices

Among the G^1-conditions stated in the previous section, only (6), (7) and (8) can be solved at each boundary of the macro-patch independently from the other boundaries. This will be detailed in the next section. Conditions (4) and (5) involve the control points b_0, d_0, c_1, d_1 which also appear in the G^1-conditions between S_{i-2} and S_{i-1} and between S_i and S_{i+1}. In order to find a compatible solution we need to solve conditions (4) and (5) for all macro-patches around the common input mesh vertex v of valence n. As a result, the points b_1, c_1, d_1 and b_2 will be computed for all macro-patch boundaries around v.
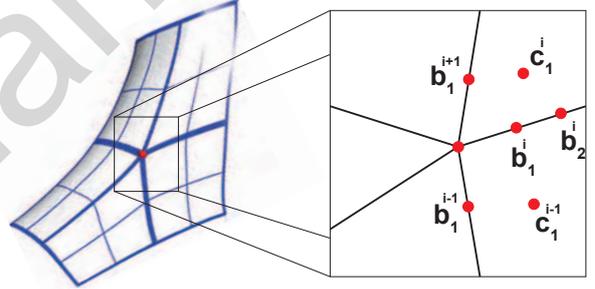

Figure 5: Notations for the control points involved in the G^1-conditions between macro-patches around an input vertex.

Let us first introduce some notations for the control points around vertex v as illustrated in Figure 5. b_0^i, b_1^i, b_2^i denote the first three control points of the curve shared by the macro-patches S_{i-1} and S_i. We consider b_j^i to be a 1×3 line matrix containing the three Cartesian coordinates. We denote by $\mathbf{b}_j = [b_j^1, \dots, b_j^n]^T$, $j = 0, 1, 2$, the $n \times 3$ matrix collecting these control points around the input mesh vertex v. Note that $b_0^i = v$ for all i (interpolation). Analogously, we use $\mathbf{c}_1 = [c_1^1, \dots, c_1^n]^T$ for collecting all control points c_1^i. Following these notations, conditions (4) and (5) can now be stated for all pairs of adjacent macro-patches (S_{i-1}, S_i) around v by:

$$
P(\mathbf{b}_1 - \mathbf{b}_0) = \mathbf{0} \tag{9}
$$

$$
T\mathbf{c}_1 = \frac{1}{4}\Phi^0\mathbf{b}_0 + (1 - \Phi^0)\mathbf{b}_1 + \frac{3}{4}\Phi^0\mathbf{b}_2, \tag{10}
$$

where P and T are two $n \times n$ matrices given in the Appendix. In order to get a smooth interpolating surface we need to maximize the number of DOF for solving (9) and (10). This can be done by the following procedure. First, we choose two 1×3 line vectors

X and Y and set

$$\mathbf{b}_1 = \mathbf{b}_0 + B_1 R \begin{pmatrix} X \\ Y \end{pmatrix}, \tag{11}$$

where B_1 and R are matrices of respective dimensions $n \times n$ and $n \times 2$ given in the Appendix. Since $PB_1 = 0$, the choice (11) solves the first condition (9). The two DOF X and Y define the tangent plane of the surface at the input mesh vertex v. It is thus possible to use these DOF to interpolate some prescribed normal vectors at the interpolated points, as explained later in Section 6.

Then, we have to find values of $\mathbf{c}_1$ and $\mathbf{b}_2$ solving (10). To this end we choose n DOF denoted by the $n \times 3$ matrix $\mathbf{q}$, and additionally, if n is even, we choose a 1×3 vector t as DOF and set $\mathbf{t}$ the $n \times 3$ matrix

$$\mathbf{t} = \begin{pmatrix} t \\ -t \\ \vdots \\ t \\ -t \end{pmatrix}, \qquad \text{if } n \text{ is even.}$$

Finally, we set

$$\mathbf{b}_2 = T\mathbf{q} \tag{12}$$

$$\mathbf{c}_1 = \frac{1}{4}\Phi^0 \mathbf{b}_0 + (1 - \Phi^0)\overline{\mathbf{b}}_1 + \frac{3}{4}\Phi^0 \mathbf{q} \quad \text{if } n \text{ is odd} \tag{13}$$

$$\mathbf{c}_1 = \frac{1}{4}\Phi^0 \mathbf{b}_0 + (1 - \Phi^0)\overline{\mathbf{b}}_1 + \frac{3}{4}\Phi^0 \mathbf{q} + \mathbf{t} \quad \text{if } n \text{ is even} \tag{14}$$

where

$$\overline{\mathbf{b}}_1 = \mathbf{b}_0 + \overline{B}_1 R \begin{pmatrix} X \\ Y \end{pmatrix}$$

and $\overline{B}_1$ is the $n \times n$ matrix given in the Appendix. Using the identities given in the Appendix and the fact that $T\mathbf{t} = 0$ if n is even, it is easy to prove that this choice for $\mathbf{b}_2$ and $\mathbf{c}_1$ gives a solution for (10).

To summarize, we compute the control points b_1^i, b_2^i and c_1^i around each input vertex by choosing the DOFs $X, Y, \mathbf{q}$ and additionally t if n is even, and by applying (12) and (13) if n is odd or (14) is n is even. In the special case of a regular vertex v, i.e. $n = 4$, then we replace (12) by $\mathbf{b}_2 = \mathbf{q}$.

4.4 Control points along macro-patch boundaries

The remaining 10 control points to be computed along the boundary curve, see Figure 6, are $c_2, c_3, c_4 = \widetilde{c}_4, \widetilde{c}_3, \widetilde{c}_2$ and $d_2, d_3, d_4 = \widetilde{d}_4, \widetilde{d}_3, \widetilde{d}_2$.
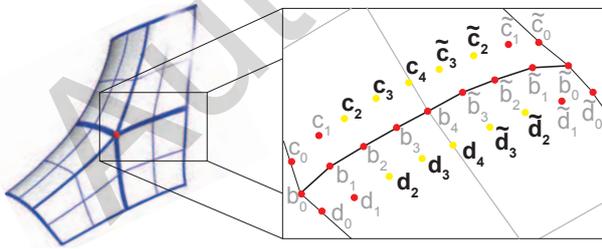

Figure 6: Notations for G^1-conditions between two macro-patches. The control points denoted with light gray letters have already been computed in Section 4.3 and 4.1. The remaining control points denoted with black letters are computed as explained in Section 4.4.

The 6 central control points are involved in both, the G^1 continuity between the macro-patches S^{i-1} and S^i, and the C^1 continuity between the Bézier patches inside each macro-patch. The

remaining G^1 conditions that still need to be fulfilled are stated in (6) , (7) and (8), whereas the C^1-conditions imply

$$\begin{aligned} c_3 - 2c_4 + \widetilde{c}_3 &= 0 \\ d_3 - 2d_4 + \widetilde{d}_3 &= 0. \end{aligned}$$

In order to solve these last two constraints as well as (7) and (8) with the maximal number of DOFs, we proceed by choosing c_3 and $c_4 = \widetilde{c}_4$ as DOFs and by setting

$$\widetilde{c}_3 = 2c_4 - c_3 \tag{15}$$

$$\widetilde{d}_3 = 2\widetilde{b}_3 - \widetilde{c}_3 \tag{16}$$

$$d_4 = \widetilde{d}_4 = 2b_4 - c_4 \tag{17}$$

$$d_3 = 2b_3 - c_3. \tag{18}$$

The last three equations ensure the G^1 continuity constraints (7) and (8) between S^{i-1} and S^i for the left and right parts of the macro-patches. (15) ensures the C^1 continuity constraints between the left and right Bézier patches inside S^i. It is easy to see that $2d_4 - \widetilde{d}_3 = d_3$, so that the C^1 continuity constraints inside S^{i-1} are also satisfied.

Finally, the G^1 condition (6) remains to be fulfilled for the left and for the right part, and the 4 control points $c_2, d_2, \widetilde{c}_2, \widetilde{d}_2$ have to be computed. Our solution is to take c_2 and $\widetilde{c}_2$ as DOFs and to set d_2 and $\widetilde{d}_2$ by

$$\begin{aligned} d_2 &= -c_2 + 2b_2 - \tfrac{1}{3}\Phi^0(b_4 - b_3) \\ \widetilde{d}_2 &= -\widetilde{c}_2 + 2\widetilde{b}_2 - \tfrac{1}{3}\Phi^1(\widetilde{b}_4 - \widetilde{b}_3). \end{aligned}$$

4.5 Control points inside macro-patches

There are 25 remaining control points to be computed inside each macro-patch, as illustrated with red and blue points in Figure 7. In order to solve the C^1 continuity constraints be-
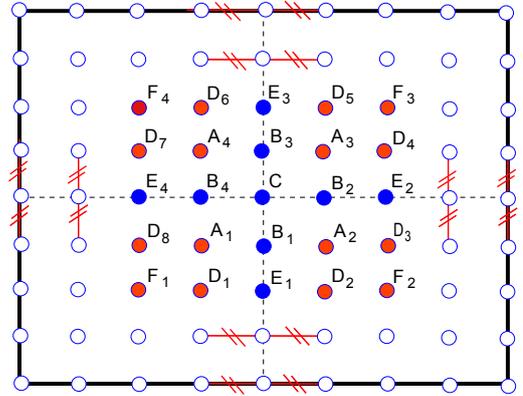

Figure 7: Bézier control points of a piecewise bi-quartic macro-patch. The empty control points ensure G^1-continuity between adjacent macro-patches. The colored control points ensure C^1-continuity inside a macro-patch.

tween the four Bézier patches, we choose the 16 control points $A_1, \ldots, A_4, D_1, \ldots, D_8, F_1, \ldots, F_4$ as DOFs, and we set the remaining 9 control points as

$$B_i = \frac{A_i + A_{i+1}}{2}, \quad C = \frac{A_1 + A_2 + A_3 + A_4}{4}, \quad E_i = \frac{D_{2i-1} + D_{2i}}{2}$$

$$i = 1, \ldots, 4.$$

5 Setting DOF

In this section we explain how to set up the DOF of our interpolating surface. The choice of default values is given in Section 5.1 while Section 5.2 tells how optimal values can be computed.

Since our interpolation method is local, all DOF influence the surface shape locally. They can be divided into three groups depending on which part of the surface they act on.

- **Vertex DOF**: $X, Y \in \mathbb{R}^3$, $\mathbf{q} = [q_1, \ldots, q_n]^T$, $q_i \in \mathbb{R}^3$, and if n is even, the twist vector $t \in \mathbb{R}^3$ (Section 4.3) for computing the control points around each mesh vertex.
- **Edge DOF**: $c_2, \widetilde{c}_2, c_3, c_4 = \widetilde{c}_4 \in \mathbb{R}^3$ for computing the control points along the boundary between two macro-patches (Section 4.4).
- **Face DOF**: 16 control points inside each macro-patch (Section 4.5).

Degrees of freedom can be set heuristically in two different ways, either by computing them explicitly from the mesh geometry or by taking locally some energy into account. The default values we propose belong to the first category, since they are simple and produce an initial shape instantly.

5.1 Default DOF

The default values enable to generate a smooth interpolating G^1 surface for any given quad mesh. It is a well known fact [24] that a smooth boundary curve network is mandatory for mesh interpolation schemes to generate globally smooth surfaces. To this end we set the following default values for vertex DOF heuristically by taking into account the valence of the mesh vertex v and the geometry of the local vertex neighborhood $v_1, \ldots, v_n$. They result in first order derivative points that form an affine n-gon in the tangent plane spanned by X, Y. The scalar parameter α is a tension parameter. Our default choice is $\alpha = 1$. The influence of α is illustrated in Figure 11. The three surfaces on the top right are obtained respectively with $\alpha = 0.2, \alpha = 0.6, \alpha = 1.5$ and the other DOF with default values. Similarly for the second order derivative points $\mathbf{b}_2$ the local mesh geometry is taken into account. The geometric interpretation of the default choice of $\mathbf{q}$ is that the resulting $\mathbf{b}_2$ are an affine combination of the mesh vertex $\mathbf{b}_0$ and the midpoint of the barycenter of the two neighboring triangles $\Delta(b_0, v_{i-1}, v_i)$ and $\Delta(b_0, v_i, v_{i+1})$.

$$\begin{pmatrix} X \\ Y \end{pmatrix} = \frac{\alpha}{n} R^T \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix}$$

$$\mathbf{q} = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix}$$

$$t = 0 \qquad \text{if } n \text{ is even.}$$

Note that X, Y can also be chosen as to interpolate prescribed normal vectors at the mesh vertices, see Figure 16. The remaining edge and face DOF can bet set either by computing them explicitly from the mesh geometry or by taking locally some energy into account. Let us first propose some heuristics belonging to the first category, which can then be improved with a surface fairing step. The 4 edge DOF are control points along the mesh edges. They are computed to form a parallelogram with three existing points as follows:

$$\begin{aligned} c_2 &= c_1 + b_2 - b_1 \\ \widetilde{c}_2 &= \widetilde{c}_1 + \widetilde{b}_2 - \widetilde{b}_1 \\ c_3 &= c_2 + b_3 - b_2 \\ c_4 = \widetilde{c}_4 &= c_3 + \widetilde{b}_4 - \widetilde{b}_3. \end{aligned}$$

Note that at this stage, all boundary control points and first inner rows of control points are completely determined for each macro-patch, see empty control points in Figure 7.

The parallelogram rule is also used to set the face DOF as follows. The parallelogram rule first enables to compute $F_1, \ldots, F_4$ from the three neighboring hollow points. The same procedure is then applied in order to compute $A_1, \ldots, A_4, D_1, \ldots, D_8$. Even though default values for DOFs cannot provide optimal values, they enable to generate a first smooth shape very quickly. All our initial surfaces have been computed using these settings.

A second, but computationally more expensive way to set the DOF is to take energy E into account locally inside each patch by solving small least squares systems. This may improve the shape quality slightly, even though it is a well known fact that only surface fairing based on global energy minimization [24] can yield shapes of high visual quality.

5.2 Energy minimization

Despite the fact that degrees of freedom are very useful for shape control, it is known that finding appropriate heuristics for setting values of DOFs remains a difficult task. Offering to a user the possibility to interactively change the values of DOF by using some slide bars is only useful for models with a few macro-patches. This user interaction becomes however tedious when the DOF have to be manipulated locally for many vertices, edges and macro-patches.

The standard and most effective way to set DOF optimally is to use energy optimization techniques. The linearized thin plate energy functional combined with the membrane functional

$$E = \int_\Omega (S_{uu}^2 + 2S_{uv}^2 + S_{vv}^2) + \lambda(S_u^2 + S_v^2) du dv$$

is the most popular surface fairing functional since over 30 years [28, 9, 2]. Its simplicity and efficiency makes it a standard functional for smooth surface design and fairing techniques not only for parametric surfaces but for any kind of geometry. The parameter $\lambda \geq 0$ is a tension parameter. The greater its value, the more the surface is tight between the interpolated points [36]. Note, that in the curve case the analogous functional is minimized by Nielson's ν-splines [26], ν being the tension parameter. The use of non-linear functionals, such as the total curvature, the Laplace-Beltrami operator [8], the diffusion and curvature flow operator [5] or MVC [25] are known to offer better results in some cases but to the price of much higher computational cost, and convergence can not always be guaranteed.

In this paper the optimal values of the DOF are computed using the standard linearized thin-plate energy minimization $\min_{DOF}(E)$, with $\lambda = 0$ in order to make the underlying interpolated mesh *invisible*. Since the functional E is quadratic, convex and differentiable the unique minimum is obtained if and only if $Grad(E) = 0$. The vector d for all DOF minimizing E is the solution to the normal linear equations $Md = b$, where M is a symmetric positive definite matrix of size k, where k is the number of DOF. A solution can be computed using a Cholesky factorization. For large matrices it is however more efficient to use an iterative method. We implemented a conjugate gradient solver using the Acro optimization package from Sandia National laboratories [17]. The conjugate gradient method converges for convex functionals, thus for E, in maximal k iterations. This is a theoretical bound, in practice the number of iterations is proportional to $\sqrt{cond(M)}$ [40]. One iteration costs ck elementary arithmetic operations, where c is the mean number of non-zero elements per row of M.

6 Results

Unless explicitly stated all examples shown in this section are computed using optimal degrees of freedom computed as explained in Section 5.2. The meshes were taken from Cindy Grimm's web site [1]. We use either isophotes [35] or global illumination rendering using photon mapping to highlight the visual smoothness of our results. Figure 17 illustrates many examples with various shapes, curvature behaviors and topological types.
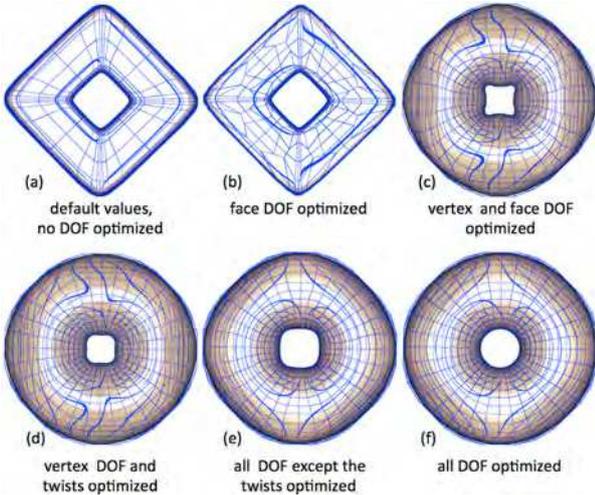
Figure 8: Optimization of different subsets of degrees of freedom. All remaining DOF are set with default values. The optimized subsets of DOF are respectively from left to right: (a) no DOF optimized, (b) face DOF, (c) vertex and face DOF, (d) vertex DOF and twists, (e) all DOF without twists, (f) all DOF. The input mesh is the torus mesh of Figure 10, second row. The Bézier control net and isophotes are drawn in blue.

Our method is flexible in the sense that the DOF can be set in various ways in order to get the desired shape effect, in particular to get visually pleasant shapes. In a first step we always compute a surface with the default values (Section 5.1) DOF. These default values are however not optimal, since they use heuristics based on local point neighborhoods. It is then possible to improve the surface quality globally using the iterative solver (Section 5.2) to compute optimal DOF.

The iterative solver also allows to select subsets of DOF for optimization, whereas the remaining DOF keep their default values. The different shape effects of computing subsets of DOF optimally can be observed in Figure 8 on a torus quad mesh. From left to right different subsets of DOF are computed optimally whereas the remaining DOF are set with their default values. The surface obtained by optimizing all DOF is shown in Figure 8(f). This example clearly shows why it is crucial to maximize the number of degrees of freedom when developing a method for smooth surface interpolation. We illustrate this observation further with a zoom on the knotty data set in Figure 9. The left image shows the surface for which all DOF except the twists have been optimized. The resulting surface is bumpy, because an important type of DOF is missing. The set of DOF is the same as in Figure 8(e), but the bigger size of the input mesh emphasizes the imperfections which cannot be seen with small size meshes. Figure 9-right shows our surface with all DOF optimized. In Figure 17 further surfaces interpolating large-sized meshes with our scheme are shown, and the isophotes assess the flexibility of the present interpolant with a pleasant visual behavior.
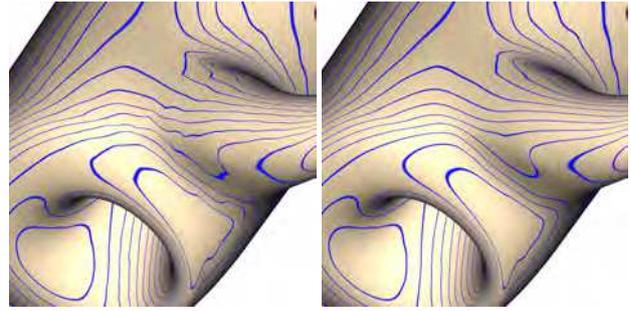
Figure 9: Optimization with all DOF except the twists (left) and all DOF (right). The isophotes are smoother when all DOF are optimized.

More complex examples of our interpolating G^1-continuous surface are shown in Figure 2. Each object representing the character Gumby is composed of nine independent genus 0 meshes representing the body and the face elements separately. Each object has in total 1032 faces 1050 vertices. 23686 vector DOF have been computed optimally as explained in Section 5.2. One of these Gumby quad meshes has been used in Figure 12 for a comparison of our interpolating surfaces with Catmull Clark surfaces. Both techniques produce visually smooth results. Note however, that interpolation allows for a more direct control of the shape and enables to reproduce sharper features.

Comparison to interpolatory schemes for quad meshes. In Figure 10 a comparison to Kobbelt's [16] interpolatory subdivision on quad meshes is shown. A further comparison in Figure 11 shows that the visual quality of our interpolating spline surfaces is not altered when tightening or inflating the shape. We choose the cube mesh since a further comparison is hereby possible with another method of Li and Zheng [18] who construct interpolatory subdivision schemes based on existing approximating subdivisions, see the shape and reflection lines on the cube example in Figure 8 in [18]. Quad meshes can also be interpolated by triangular schemes, but splitting quads into triangles introduces artifacts visible in the interpolating surface.

Interpolation vs. approximation. Even though it is difficult to compare interpolation and approximation schemes, it is worth mentioning that the interpolation property is important for several applications where a direct shape control is required, such as connecting surfaces of different representations together or direct shape manipulations. Furthermore it is easier to control the final shape for a given input mesh with interpolating surfaces, as the comparison to Catmull-Clark subdivision surfaces in Figure 12 show. This is especially true when the input quad mesh is coarse, in which case the Catmull-Clark subdivision surface may be very distant from the control mesh.

Splines vs. subdivision surfaces. As mentioned in the introduction spline surfaces have some important advantages over subdivision surface. Let us provide an example of the compatibility of spline surface with existing CAD standards. CAD-Systems such as CATIA offer powerful tools for any kind of modeling operation for splines. To test several of them with our interpolating surface, we imported the vase surface of Figure 1 as a collection of Bézier patches via the STEP format [1] into CATIA and applied the following four operations illustrated in Figure 13. We transformed our surface into a volume and then hollowed out our vase. To this end, we applied a *Boolean difference* with another surface. The latter has been created as a *surface of revolution* generated by a spline curve around the z-axis of our vase. The resulting opening on the top of the vase had a sharp border that we round off with another standard CAD operation, a *constant radius blending*. Fi-
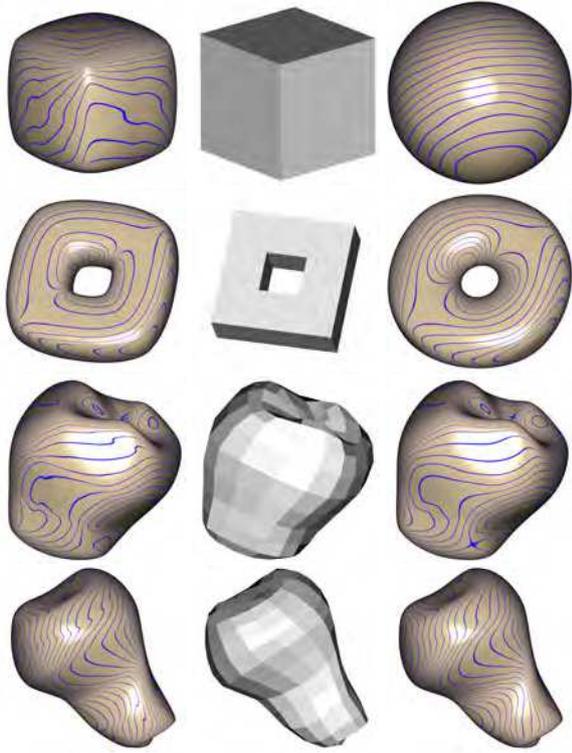
7

Figure 10: Comparison of the interpolating subdivision scheme of [16] (left) with our quartic interpolant (right). In all cases the isophotes are smoother with our quartic interpolant.

nally we visualize a *cut* of our model by the *yz*-plane.

Another advantage is the ability of our surface to model sharp edges or even cuts in the surface. This can either be obtained by relaxing the continuity constraints along mesh edges, or by post processing the Bézier control points individually for each patch, as it has been done for the examples in Figures 14 and 15.

Normal vectors as design handles. The capability to interpolate given vertex normal vectors provides in fact some design handles to the user, which can interactively manipulate then. From a given normal vector n at a mesh vertex v the tangent plane vectors X, Y are computed as

$$
\begin{pmatrix} X \\ Y \end{pmatrix} = \frac{\alpha}{n} R^T \begin{pmatrix} v_1^* \\ \vdots \\ v_n^* \end{pmatrix}
$$

where v_i^* are the orthogonal projections of the vertex neighborhood points v_i onto the plane orthogonal to n. Figure 16 gives an example of normal interpolation whereas the remaining DOF are still set optimally.

7 Conclusion

In this paper, a new method for interpolating an arbitrary 2-manifold quad mesh with a G^1-continuous piecewise quartic Bézier surface is introduced. Explicit linear formulas for computing all Bézier control points are given. Particular effort has been done to extract as many as possible degrees of freedom. We demonstrate with many examples the ability of our method to generate visually smooth shapes. The method is local and well adapted to parallel programming. We plan to explore in future works the use of a CUDA implementation that would allow to interact in real-time with our surfaces.
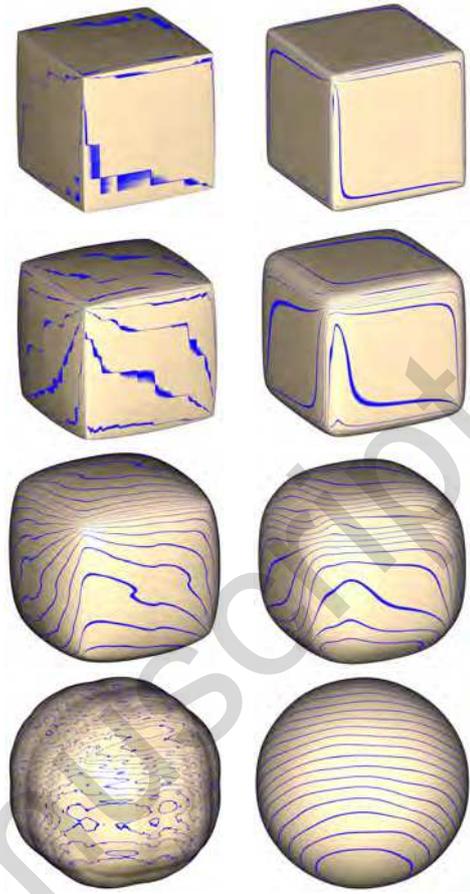

Figure 11: Comparison of the interpolating subdivision scheme of [16] (left) with our quartic interpolant (right) for a cube mesh. The left column shows the results with the coefficient ω of [16] equal to $0.2, 0.6, 1.0$ and 1.5 from top to bottom. The right column shows from top to bottom our quartic interpolant with default DOF and α equal to $0.5, 1., 1.2$ and all optimal DOF on the bottom right. This example allows a further comparison to the interpolatory method of Li and Zheng [18] (see Figure 8 in [18]).

Appendix

In the following matrices the indices are taken modulo *n*.

$$
R_{i1} = \cos\left(\frac{2i\pi}{n}\right), \qquad R_{i2} = \sin\left(\frac{2i\pi}{n}\right), \quad i = 1, \dots, n,
$$

$$
\{B_1\}_{ij} = \cos\left(\frac{2\pi(j-i)}{n}\right), \quad i, j = 1, \dots, n,
$$

$$
\{\overline{B}_1\}_{i,j} = \left[\cos\left(\frac{2\pi(j-i)}{n}\right) + \tan\left(\frac{\pi}{n}\right) \sin\left(\frac{2\pi(j-i)}{n}\right)\right].
$$

8

Figure 12: Comparison between our interpolating surface on the left and (non interpolating) Catmull-Clark surface on the right using the same input quad mesh.
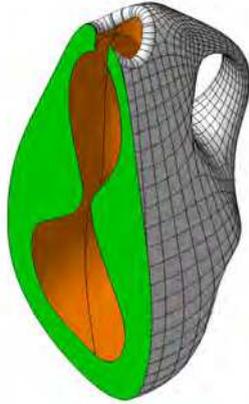

Figure 13: Our quartic G^1 interpolant can be post-processed in standard CAD systems. This example has been created using CATIA. The vase surface from Figure 1 is first transformed into a volume. We then computed another volume bounded by a surface of revolution. Aligning the axis of both objects and application of the Boolean difference results in the hollowed-out vase. The sharp border at the top of the vase has been rounded off by using a constant radius blending operator. The outer surface is exactly part of our G^1 quartic interpolant.

$$P = \begin{bmatrix} -\Phi^0 & \frac{1}{2} & 0 & \cdots & 0 & \frac{1}{2} \\ \frac{1}{2} & -\Phi^0 & \frac{1}{2} & & & 0 \\ 0 & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & & & & \frac{1}{2} \\ \frac{1}{2} & & & & \frac{1}{2} & -\Phi^0 \end{bmatrix}$$

$$T = \begin{bmatrix} \frac{1}{2} & 0 & \cdots & & 0 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \cdots & & & 0 \\ & & \ddots & & & \\ 0 & \cdots & & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & \cdots & & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix},$$

The following identities hold:

$$PB_1 = 0 \qquad T\overline{B}_1 = B_1.$$

Figure 14: Input mesh (top left). Sharp edges (top right) and cuts (bottom left and right) can be simply inserted by modifying the control points.

Figure 15: Texture mapping and rendering of some of the surfaces from Figure 14

References

[1] Iso 10303-1:1994 industrial automation systems and integration product data representation and exchange - overview and fundamental principles, international standard. ISO TC184/SC4, 1994.

[2] G. P. Bonneau and H. Hagen. Variational design of rational Bézier curves and surfaces. In L. Laurent and L. Schumaker, editors, Curves and Surfaces, volume II, pages 51–58. 1994.
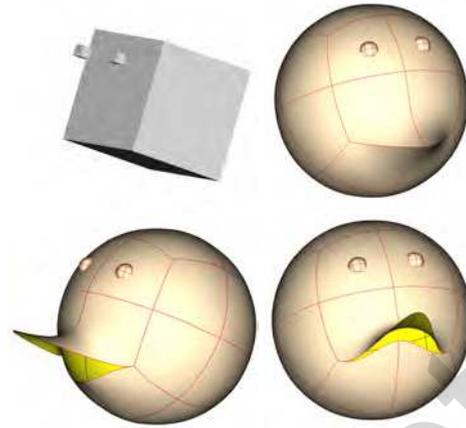
[3] E. Catmull and J. Clark. Recursive generated B-spline surfaces on arbitrary topological meshes. Computer Aided-Design, 10:350–355, 1978.

[4] H. Chiyokura and F. Kimura. Design of solids with free-form surfaces. In Computer Graphics Proceedings (SIGGRAPH '83), pages 289–298, 1983.

[5] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '99, pages 317–324, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[6] N. Dyn, D. Levine, and J. A. Gregory. A butterfly subdivision scheme for surface interpolation with tension control. ACM Trans. Graph., 9(2):160–169, 1990.

[7] J. Fan and J. Peters. Smooth bi-3 spline surfaces with fewest knots. Comput. Aided Des., 43(2):180–187, 2011.

[8] G. Greiner. Variational design and fairing of spline surfaces. In Proc. Eurographics 1994, pages 143–154, 1994.

[9] H. Hagen and G. Schulze. Automatics smoothing with geometric surface patches. Computer Aided Geometric Design, pages 231–236, 1987.

[10] S. Hahmann and G.-P. Bonneau. Triangular G^1 interpolation by 4-splitting domain triangles. Computer Aided Geometric Design, 17(8):731–757, 2000.

[11] S. Hahmann, G.-P. Bonneau, and B. Caramiaux. Bicubic G^1 interpolation of irregular quad meshes using a 4-split. In GMP'08: Proceedings of the 5th international conference on Advances in geometric modeling and processing, pages 17–32. Springer-Verlag, 2008.
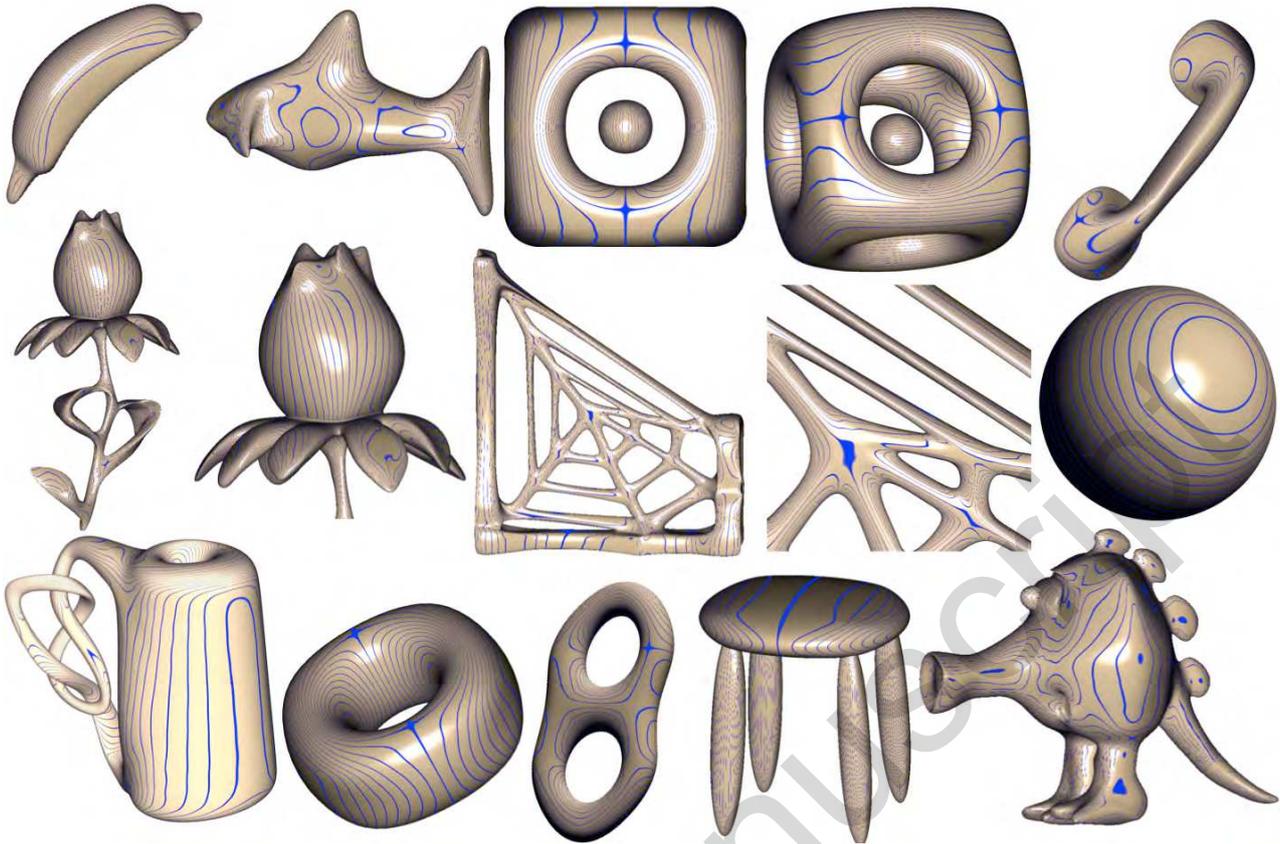
Figure 17: Various interpolating surfaces rendered with isophotes. Different shapes, curvature behavior and topological types can be observed.

Figure 16: Normal vector interpolation. Both surfaces interpolate the same cube mesh. Left: the normal vectors are prescribed at the 8 mesh vertices with the remaining DOF set optimally. Right: interpolated cube mesh with all optimal DOF.

[12] J. Hahn. Filling polygonal holes with rectangular patches. In Theory and practice of geometric modeling, pages 81–91. Springer-Verlag New York, Inc., 1989.

[13] M. Halstead, M. Kass, and T. DeRose. Efficient, fair interpolation using catmull-clark surfaces. In Computer Graphics Proceedings (SIGGRAPH 93), pages 35–44, 1993.

[14] M. Kapl, M. Byrtus, and B. Jüttler. Triangular bubble spline surfaces. Comput. Aided Des., 43(11):1341–1349, 2011.

[15] K. Karciauskas and J. Peters. Biquintic g2 surfaces. In R. J. Cripps, G. Mullineux, and M. A. Sabin, editors, The Mathematics of Surfaces XIV, pages 213–236. The Institute of Mathematics and its Applications, 2013.

[16] L. Kobbelt. Interpolatory subdivision on open quadrilateral nets

with arbitrary topology. Computer Graphics Forum, 15(3):409–420, 1996.

[17] S. N. Laboratoriesl'auteur. Acro: A common repository for optimizers. https://software.sandia.gov/trac/acro.

[18] X. Li and J. Zheng. An alternative method for constructing interpolatory subdivision from approximating subdivision. Comput. Aided Geom. Des., 29(7):474–484, 2012.

[19] H. Lin, W. Chen, and H. Bao. Adaptive patch-based mesh fitting for reverse engineering. Comput. Aided Des., 39(12):1134–1142, 2007.

[20] Q. Liu and T. C. Sun. G^1 interpolation of mesh curves. Computer-Aided Design, 26(4):259–267, 1994.

[21] Y. Liu and S. Mann. Approximate continuity for parametric Bézier patches. In SPM '07 ACM symposium on Solid and physical modeling, pages 315–321, 2007.

[22] C. Loop. A G^1 triangular spline surface of arbitrary topological type. Computer Aided Geometric Design, 11:303–330, 1994.

[23] C. Loop and T. D. DeRose. Generalized B-spline surfaces of arbitrary topology. In Computer Graphics Proceedings (SIGGRAPH 90), pages 347–356, 1990.

[24] M. Lounsbery, S. Mann, and T. DeRose. Parametric surface interpolation. IEEE Computer Graphics and Applications, 12(5):45–52, Sept. 1992.

[25] H. P. Moreton and C. H. Séquin. Functional optimisation for fair surface design. Computer Graphics, 26(2):167–176, 1992.

[26] G. Nielson. Some piecewise polynomial alternatives to splines under tension, pages 209–235. Academic Press, New York, 1974.

[27] M. Nießner, C. Loop, M. Meyer, and T. Derose. Feature-adaptive gpu rendering of catmull-clark subdivision surfaces. ACM Trans. Graph., 31(1):6:1–6:11, Feb. 2012.

[28] H. Nowacki and D. Reese. Design and fairing of ship surfaces. In R. Barnhill and W. Boehm, editors, Surfaces in CAGD, pages 121–134. North-Holland, Amsterdam, 1983.

[29] J. Peters. Smooth interpolation of a mesh of curves. Constructive Approx., 7:221–246, 1991.

[30] J. Peters. Biquartic C^1-surface splines over irregular meshes. Computer-Aided Design, 27(12):895–903, 1995.

[31] J. Peters. C^1 surface splines. SIAM Journal of Numerical Analysis, 32(2):645–666, 1995.

[32] J. Peters. C^2 free-form surfaces of degree (3, 5). Computer Aided Geometric Design, 19(2):113–126, 2002.

[33] J. Peters and J. Fan. On the complexity of smooth spline surfaces from quad meshes. Comput. Aided Geom. Des., 27(1):96–105, 2010.

[34] Pixar. Open subdiv, subdivision surface library. http://graphics.pixar.com/opensubdiv/, 2012.

[35] T. Poeschl. Detecting surface irregularities using isophotes. Computer Aided Geometric Design, 1(2):163 – 168, 1984.

[36] H. Pottmann and M. Hofer. A variational approach to spline curves on surfaces. Comput. Aided Geom. Des., 22(7):693–709, Oct. 2005.

[37] U. Reif. Biquadratic G-spline surfaces. Computer Aided Geometric Design, 12(2):193–205, 1995.

[38] R. F. Sarraga. G^1 interpolation of generally unrestricted cubic Bézier curves. Computer Aided Geometric Design, 4(1-2):23–39, 1987.

[39] X. Shi, T. Wang, and P. Yu. A practical construction of G^1 smooth biquintic B-spline surfaces over arbitrary topology. Computer-Aided Design, 36(5):413–424, 2004.

[40] G. Strang. Introduction to Applied Mathematics. Wellesley-Cambridge Press, 1986.

[41] J. J. V. Wijk. Bicubic patches for approximating non-rectangular control-point meshes. Computer Aided Geometric Design, 3(1):1–13, 1986.

[42] D. Zorin, P. Schröder, and W. Sweldens. Interpolating subdivision for meshes with arbitrary topology. In Computer Graphics Proceedings (SIGGRAPH 96), pages 189–192, 1996.

11