Computer-Aided Design 41 (2009) 441–455

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

High-order approximation of implicit surfaces by G^1 triangular spline surfaces

Wei-hua Tong [a], Tae-wan Kim [a,b,*]

[a] Department of Naval Architecture and Ocean Engineering, Seoul National University, Seoul 151-744, Republic of Korea

[b] Research Institute of Marine Systems Engineering, Seoul National University, Seoul 151-744, Republic of Korea

ARTICLE INFO

Article history:
Received 15 May 2008
Accepted 2 February 2009

Keywords:
G^1 continuity
Geometric Hermite interpolation
Boundary curves network
Equality constrained optimization
Vertex enclosure constraint

ABSTRACT

In this paper, we present a method for the approximation of implicit surface by G^1 triangular spline surface. Compared with the polygonization technique, the presented method employs piecewise polynomials of high degree, achieves G^1 continuity and is capable of interpolating positions, normals, and normal curvatures at vertices of an underlying base mesh. To satisfy vertex enclosure constraints, we develop a scheme to construct a C^2 consistent boundary curves network which is based on the geometric Hermite interpolation of normal curvatures. By carefully choosing the degrees of scalar weight functions, boundary Bézier curves and triangular Bézier patches, we propose a local and singularity free algorithm for constructing a G^1 triangular spline surface of arbitrary topology. Our method achieves high precision at low computational cost, and only involves local and linear solvers which leads to a straightforward implementation. Analyses of freedom and solvability are provided, and numerical experiments demonstrate the high performance of algorithms and the visual quality of results.

1. Introduction

In computer-aided design (CAD), computer graphics, and computer-aided geometric design (CAGD), there are two common ways for representing surfaces: parametric and implicit. Both representations are well developed and valuable for geometric modeling. Because the characteristics of parametric and implicit representations are complementary in respect of certain geometric operations, many researchers have addressed the issue of conversion from one form to the other. The process of converting a parametric to an implicit representation is known as *implicitization*. The reverse of implicitization is *parameterization*, which is much harder. It is well known that any parametric representation can be converted into an implicit representation. But, not every implicit surface admits a parameterization, even if it is algebraic. The alternative pragmatic solution is to approximate an implicit representation with a parametric representation.

Over recent decades, the polygonization of implicit surface has been extensively studied by many researchers, because it has many potential applications in many fields. However, this technique suffers from two major drawbacks. First, a high-precision approximation of an implicit surface may require a huge number of polygons, due to the poor approximation power of piecewise linear function. Second, a piecewise linear representation only has C^0 continuity, while many applications need higher order of continuity. For instance, in CAD/CAM, G^1 continuity is usually the minimum requirement.

1.1. Contributions and organization

In this paper, we address the problem of approximating implicit surfaces by triangular spline surfaces, which have G^1 continuity and interpolate positions, normals, and normal curvatures at vertices of an underlying base mesh. And we make contributions to the geometric Hermite interpolation and the construction of geometrically continuous spline surface as follows:

- A geometric Hermite interpolation scheme is proposed, which uses normal curvatures instead of curvatures. By restating interpolation as an equality constrained least-squares fitting problem, we solve this problem efficiently through the local SQP method.
- An algorithm for constructing G^1 triangular spline surface of arbitrary topological type is developed, which is local and free of singularities. To achieve this goal, we separate all the G^1 continuity constraints into vertex and edge G^1 continuity systems, and solve them by the equality constrained quadratic programs.

To our knowledge, no other G^1 scheme is available for this purpose yet. For similar problems, Derose and Mann [1] proposed a cubic geometric Hermite interpolant that interpolates positions, normals and second fundamental forms at triangulated data, but

* Corresponding author at: Department of Naval Architecture and Ocean Engineering, Seoul National University, Seoul 151-744, Republic of Korea. Tel.: +82 2 880 1434; fax: +82 2 888 9298.
E-mail address: taewan@snu.ac.kr (T.-w. Kim).

0010-4485/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2009.02.012

their surface was only C^0. Further, in Mann's dissertation [2], this scheme was used to approximate implicit surfaces.

The remainder of this paper is organized as follows. Section 2 examines some related work. Section 3 gives some preliminaries and an overview of algorithm framework. Section 4 presents a method for generating a triangular base mesh. Section 5 proposes an algorithm for constructing a C^2 consistent boundary curves network. Section 6 describes our way in which continuity constraints are solved to obtain a G^1 triangular spline surface. Section 7 analyzes our scheme in detail. Section 8 shows the performance of algorithms and the visual quality of results. Section 9 concludes this paper with proposals for future work.

2. Related work

As such there is a large body of related work, we shall review it as two separate topics, focusing on the approaches that are most closest to ours.

2.1. Geometric Hermite interpolation

The research on geometric Hermite interpolation was initiated by the well-known paper of de Boor et al. [3], in which a segmented curve is approximated by a G^2 continuous piecewise cubic curve with the same position, tangent and curvature at each end-point. The same problem was independently considered by Goodman and Unsworth [4], who presented a shape-preserving interpolation algorithm.

Instead of using polynomials, Goodman [5] investigated the possibility of geometric Hermite interpolation by rational cubic splines. This method was subsequently extended in [6] using multiple lines as constraints on the data points, which must lie on one side of these lines. Meek et al. [7] introduced a polyline as a more general version of this constraint. Recently attempts to obtain a fair curve automatically have involved using C-shaped curves [8] and Pythagorean Hodograph (PH) curves [9] to interpolate the given G^2 Hermite data.

The geometric Hermite interpolation of space curve has been studied by Höllig [10] using rational cubic spline curves. Subsequently in Höllig and Koch [11], the standard cubic Hermite interpolation was improved by interpolation through a third point within the parameter interval, achieving the optimal approximation order 5. Further research on theoretical aspects of the approximation order and smoothness have been presented in Höllig and Koch [12].

Starting from an analysis of the scheme proposed by Höllig [10], Peters [13] used polynomial piecewise quartic space curves to interpolate positional, tangent and curvature data. However, Schaback [14] asserted that the degree of G^2 piecewise polynomial curves in Hermite interpolation must be at least five in order to cope with all geometrical situations. A recent survey on the development in geometric Hermite interpolation theory can be found in [15].

2.2. Construction of geometrically continuous spline surfaces

Requirements for geometric or visual continuity instead of parametric continuity are prevalent in many research fields, and excellent surveys can be found in [16,17]. The G^1 *continuity*, i.e. tangent-plane continuity, is particularly popular in freeform parametric surface design. For the sake of clarity, we will restrict ourselves to reviewing the relevancy to the construction of G^1 spline surfaces of arbitrary topological type. Recursive subdivision surfaces are well known to deal with arbitrary topologies, but they are not parametric surfaces. We will not review them here; and readers are referred to Warren and Weimer [18] and Peter and Reif [19] among others.

The main obstacle to be overcome in constructing a G^1 spline surface of arbitrary topological type is to satisfy the vertex enclosure constraint or twist compatibility condition, which has been widely addressed by Peters [20], and Mann et al. [21] etc. As discussed in [20], there are several feasible approaches. Early schemes, such as those due to Farin [22], and Shirman and Séquin [23] were based on *domain splitting technique*, in which multiple patches were used to span each face, this successfully decouples the cycle of constraints. One known problem of this approach is how to choose the free parameters for achieving pleasant shapes. *Convex combination* methods which have received attentions from Nielson [24], Hagen and Pottmann [25] and others, use side–side or side-vertex blending operators to handle the twist compatibility, but this produces rational patches with inconsistently defined twists at the vertices. The use of *singular parameterizations*, such as that due to Neamtu and Pfluger [26], is another possibility, but they seem to have problems in achieving pleasant shapes.

The method presented in this paper falls into the *boundary curve schemes*, in which a C^2 consistent boundary curves network will be constructed and then fill in the polynomial patches. Peters [20] presented an algorithm for the interpolation of a cubic curve mesh by bi-quartic G^1 surfaces, in which a C^2 consistent boundary curves network of quartic space curves is constructed using a previous method [13]. But there are several restrictions on its input data.

By employing the theory of circulant matrices, Loop [27] devised another way to construct a C^2 consistent boundary curves network using affine combinations of the vertices of base mesh. In that scheme, the degree of boundary curves is 4, and the degree of the fill-in patches is 6. Although its degree is one lower than that of our scheme, Loop's surface can interpolate the position data only. And as the author pointed out, this interpolation is theoretically possible, but the patch boundary curves may cause severe undulations in the final surface.

Hahmann and Bonneau [28] generalized Loop's scheme with the help of domain splitting techniques, in which each macro-patch consists of 4 quintic triangular Bézier patches. Their boundary curves network is composed of piecewise cubic C^1 curves. More, Yvart et al. [29] introduced the hierarchical triangular splines for locally adapting the level detail. Although ingenious, these schemes are still beset by the problems common to all domain splitting techniques. Another extension of Loop's scheme was done by Liu and Mann [30], who reduced the degree of fill-in patches from 6 to 5 and improved the shape of surface by constructing a virtual mesh, but achieved approximate G^1 continuity.

Recently, Cho et al. [31] proposed a local method of constructing G^1 Bézier surfaces for ship hull design, by interpolating the given curves network with odd- and 4-valent vertices. In order to deal with topologically irregular regions, they presented several subdivision schemes for preprocessing the given curves network. Analysis and avoidance of singularities for this method are presented in [32]. However, these schemes still have some topologic restrictions on the given curves network.

3. Preliminaries and overview

In this section, we give some preliminary definitions and an overview of our algorithm framework.

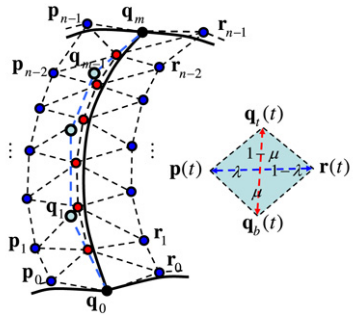


Fig. 1. The G^1 continuity constraints.

3.1. Implicit surfaces

An *implicit surface* is defined by a function $f : \Omega \subset \mathbb{R}^3 \to \mathbb{R}$ that assigns a scalar value to each point in the space Ω. The surface is the set of points $\mathbf{p} = (x, y, z) \in \Omega$ such that $f(\mathbf{p}) = c$, and is sometimes called the level-set of f, which can be written as $S = f^{-1}(c) = \{\mathbf{p} \in \Omega : f(\mathbf{p}) = c\}$; the function f is called the implicit surface function, or more commonly the implicit function.

In order for surface normals to be well defined on S, we assume that the function f is continuous and differentiable. If the gradient is non-null at a point $\mathbf{p}$, then $\mathbf{p}$ is said to be regular and $\nabla f(\mathbf{p})$ is normal to the surface at $\mathbf{p}$. Otherwise, the normal is indeterminate and the point $\mathbf{p}$ is singular. If every $\mathbf{p} \in f^{-1}(c)$ is regular, then c is a regular value of f.

In this paper, we will assume that we are given an implicit surface, i.e. the implicit function f and the regular value c, and our aim is to approximate this surface with a G^1 triangular spline surface.

3.2. G^1 triangular spline surfaces

In a triangular parametric domain $\triangle \subset \mathbb{R}^2$, every point in $\triangle$ has the barycentric coordinates $\mathbf{u} = (u, v, w)$ with respect to the vertices of $\triangle$. A *triangular Bézier patch* is a map of $\triangle$ to $\mathbb{R}^3$, which is defined by

$$\mathbf{b}(\mathbf{u}) = \sum_{|\mathbf{i}|=n} \mathbf{b}_\mathbf{i} B^n_\mathbf{i}(\mathbf{u}), \quad \mathbf{u} \in \triangle,$$

where n is the degree, and $B^n_\mathbf{i}(\mathbf{u}) = \frac{n!}{i!j!k!} u^i v^j w^k$ with $\mathbf{i} = (i, j, k)$, and $\{\mathbf{b}_\mathbf{i} \in \mathbb{R}^3\}$ are control points.

The union of some triangular Bézier patches, which meet the G^1 continuity constraint so that any two adjacent patches have a shared and continuously varying tangent plane along their common boundary curve, is a *G^1 triangular spline surface*.

3.3. G^1 continuity constraints

Let two adjacent triangular Bézier patches of degree n meet with C^0 continuity. Their common boundary is a Bézier curve $\mathbf{q}(t)$ of degree m as follow

$$\mathbf{q}(t) = \sum_{i=0}^{m} \mathbf{q}_i B^m_i(t) = \sum_{i=0}^{n} \mathbf{c}_i B^n_i(t), \quad m \leqslant n,$$

where $\{\mathbf{q}_i\}_{i=0}^m$ are the control points and $\{\mathbf{c}_i\}_{i=0}^n$ are the evaluated control points.

The G^1 continuity constraints between them (see Fig. 1) can be represented by the following equation [17]:

$$[1 - \lambda(t)]\mathbf{p}(t) + \lambda(t)\mathbf{r}(t) = [1 - \mu(t)]\mathbf{q}_b(t) + \mu(t)\mathbf{q}_t(t), \qquad (1)$$

where $\mathbf{p}(t) = \sum_{i=0}^{n-1} \mathbf{p}_i B^{n-1}_i(t)$, and $\mathbf{r}(t) = \sum_{i=0}^{n-1} \mathbf{r}_i B^{n-1}_i(t)$, and $\mathbf{q}_b(t) = \sum_{i=0}^{m-1} \mathbf{q}_i B^{m-1}_i(t)$, and $\mathbf{q}_t(t) = \sum_{i=0}^{m-1} \mathbf{q}_{i+1} B^{m-1}_i(t)$. The $\mathbf{p}_i$ and $\mathbf{r}_i$ are the control points from the first-left and first-right rows of the control net along this boundary curve respectively.

Table 1
The degrees of $\lambda(t)$, $\mu(t)$, triangular Bézier patches, and boundary Bézier curves.

$r=1$	n	s	$m=4$	$r=1$	n	s	$m=5$
1	5	2	4	1	6	2	5
1	6	3	4	1	7	3	5
1	7	4	4	1	8	4	5
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

The $\lambda(t)$ and $\mu(t)$ are called *scalar weight functions*. In our schemes, we make $\lambda(t)$ and $\mu(t)$ polynomials, which are expressed in terms of Bernstein polynomials as follow

$$\begin{aligned} \lambda(t) &= \sum_{i=0}^{r} \lambda_i B^r_i(t), \qquad 1 - \lambda(t) = \sum_{i=0}^{r} \bar{\lambda}_i B^r_i(t), \\ \mu(t) &= \sum_{i=0}^{s} \mu_i B^s_i(t), \qquad 1 - \mu(t) = \sum_{i=0}^{s} \bar{\mu}_i B^s_i(t), \end{aligned} \qquad (2)$$

where r and s are the degrees of $\lambda(t)$ and $\mu(t)$.

Let $r + n = s + m$ and we substitute Eq. (2) into Eq. (1), and rearrange the coefficients of the polynomials, then we obtain the following G^1 continuity constraint system:

$$\begin{aligned} &\sum_{i=0}^{k} (\bar{\lambda}_i \mathbf{p}_{k-i} + \lambda_i \mathbf{r}_{k-i}) \binom{r}{i} \binom{n-1}{k-i} \\ &= \sum_{i=0}^{k} (\bar{\mu}_i \mathbf{q}_{k-i} + \mu_i \mathbf{q}_{k-i+1}) \binom{s}{i} \binom{m-1}{k-i} \end{aligned} \qquad (3)$$

where $k = 0, 1, 2, \ldots, r + n - 1$.

There are many possible configurations for r, n, s and m, and some examples are given in Table 1. As explained in Section 7.1, we show that the optimal configuration is $r = 1, s = 4, m = 4$, and $n = 7$. Then Eq. (3) becomes (see Table 2).

3.4. Algorithm framework

Given a surface S defined by the implicit function f and a regular value c, i.e. $S = f^{-1}(c)$, our goal is to approximate S by a G^1 triangular spline surface S'. The basic steps in our algorithm are as follows (Fig. 2):

(i) Polygonize the implicit surface S to obtain a triangular mesh M which is a piecewise-linear approximation and is homeomorphic to S. Decimate M to yield a simplified mesh M', while preserving its topology. The vertices of M' are then projected onto S, so as to obtain a triangular base mesh B (e.g. Fig. 3(a)).

(ii) At every vertex of B, compute the normal $\mathbf{N}$, the principal curvatures $k_{\min}$ and $k_{\max}$, and the associated principal directions $\mathbf{t}_{\min}$ and $\mathbf{t}_{\max}$, all with respect to S. For each edge of B, construct a Bézier curve by the geometric Hermite interpolation. Together, these Bézier curves compose a boundary curves network C (e.g. Fig. 3(b)).

(iii) Map each facet of B onto S by projection and fit the resulting image with a triangular Bézier patch T, which need to interpolate the boundary curves of C. All these patches comprise a G^0 triangular surface (e.g. Fig. 3(c)).

(iv) Relocate the control points of each pair of adjacent patches to make them satisfy the G^1 continuity constraint. To achieve this, we separate all the continuity constraints into vertex and edge G^1 continuity systems, and solve them by the equality-constrained quadratic programs.

Table 2
The G^1 continuity constraint system.

$(\bar{\lambda}_0\mathbf{p}_0 + \lambda_0\mathbf{r}_0) =$	$(\bar{\mu}_0\mathbf{q}_0 + \mu_0\mathbf{q}_1),$	$k = 0$
$6(\bar{\lambda}_0\mathbf{p}_1 + \lambda_0\mathbf{r}_1) =$	$-(\bar{\lambda}_1\mathbf{p}_0 + \lambda_1\mathbf{r}_0) + 3(\bar{\mu}_0\mathbf{q}_1 + \mu_0\mathbf{q}_2) + 4(\bar{\mu}_1\mathbf{q}_0 + \mu_1\mathbf{q}_1),$	$k = 1$
$15(\bar{\lambda}_0\mathbf{p}_2 + \lambda_0\mathbf{r}_2) =$	$-6(\bar{\lambda}_1\mathbf{p}_1 + \lambda_1\mathbf{r}_1) + 3(\bar{\mu}_0\mathbf{q}_2 + \mu_0\mathbf{q}_3) + 12(\bar{\mu}_1\mathbf{q}_1 + \mu_1\mathbf{q}_2) + 6(\bar{\mu}_2\mathbf{q}_0 + \mu_2\mathbf{q}_1),$	$k = 2$
$20(\bar{\lambda}_0\mathbf{p}_3 + \lambda_0\mathbf{r}_3) + 15(\bar{\lambda}_1\mathbf{p}_2 + \lambda_1\mathbf{r}_2) =$	$(\bar{\mu}_0\mathbf{q}_3 + \mu_0\mathbf{q}_4) + 12(\bar{\mu}_1\mathbf{q}_2 + \mu_1\mathbf{q}_3) + 18(\bar{\mu}_2\mathbf{q}_1 + \mu_2\mathbf{q}_2) + 4(\bar{\mu}_3\mathbf{q}_0 + \mu_3\mathbf{q}_1),$	$k = 3$
$15(\bar{\lambda}_0\mathbf{p}_4 + \lambda_0\mathbf{r}_4) + 20(\bar{\lambda}_1\mathbf{p}_3 + \lambda_1\mathbf{r}_3) =$	$4(\bar{\mu}_1\mathbf{q}_3 + \mu_1\mathbf{q}_4) + 18(\bar{\mu}_2\mathbf{q}_2 + \mu_2\mathbf{q}_3) + 12(\bar{\mu}_3\mathbf{q}_1 + \mu_3\mathbf{q}_2) + (\bar{\mu}_4\mathbf{q}_0 + \mu_4\mathbf{q}_1),$	$k = 4$
$15(\bar{\lambda}_1\mathbf{p}_4 + \lambda_1\mathbf{r}_4) =$	$-6(\bar{\lambda}_0\mathbf{p}_5 + \lambda_0\mathbf{r}_5) + 6(\bar{\mu}_2\mathbf{q}_3 + \mu_2\mathbf{q}_4) + 12(\bar{\mu}_3\mathbf{q}_2 + \mu_3\mathbf{q}_3) + 3(\bar{\mu}_4\mathbf{q}_1 + \mu_4\mathbf{q}_2),$	$k = 5$
$6(\bar{\lambda}_1\mathbf{p}_5 + \lambda_1\mathbf{r}_5) =$	$-(\bar{\lambda}_0\mathbf{p}_6 + \lambda_0\mathbf{r}_6) + 4(\bar{\mu}_3\mathbf{q}_3 + \mu_3\mathbf{q}_4) + 3(\bar{\mu}_4\mathbf{q}_2 + \mu_4\mathbf{q}_3),$	$k = 6$
$(\bar{\lambda}_1\mathbf{p}_6 + \lambda_1\mathbf{r}_6) =$	$(\bar{\mu}_4\mathbf{q}_3 + \mu_4\mathbf{q}_4),$	$k = 7$

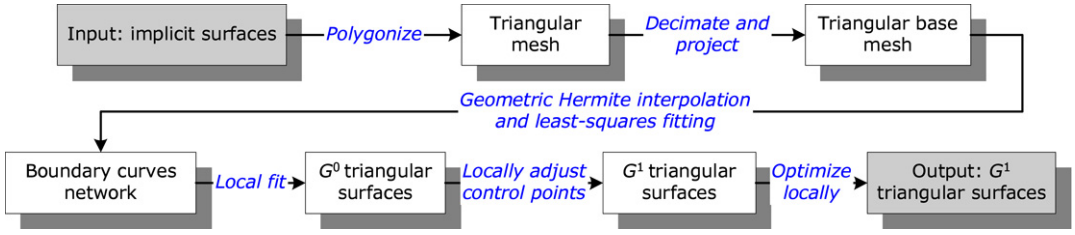


Fig. 2. Algorithm framework.

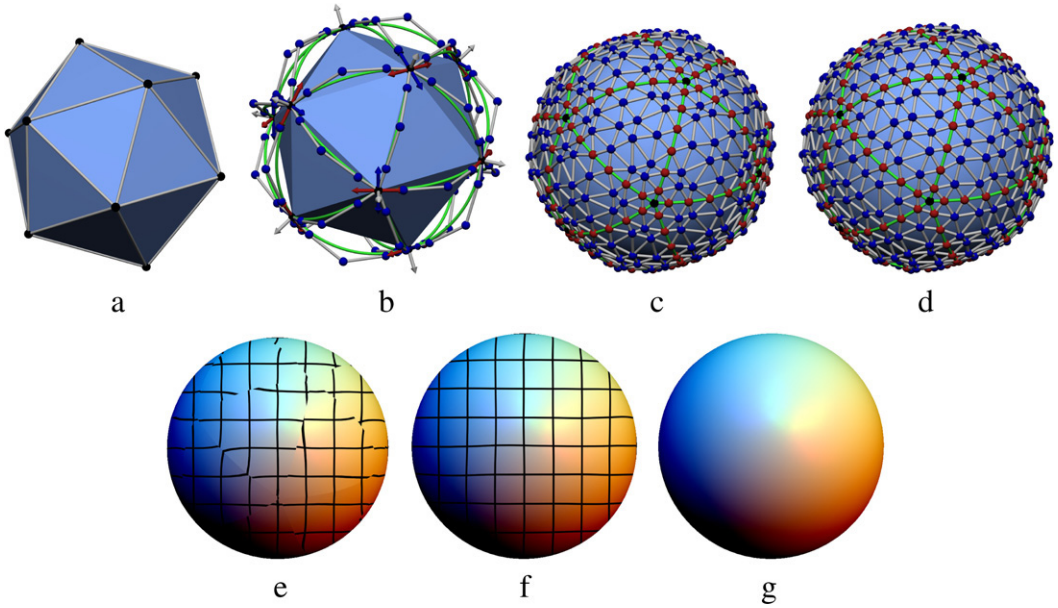


Fig. 3. An example G^1 triangular spline surface which approximates a unit sphere: (a) triangular base mesh; (b) boundary curves network; (c) G^0 surface with control nets; (d) G^1 surface with control nets; (e) G^0 surface with reflection lines; (f) G^1 surface with reflection lines; (g) G^1 surface.

(v) In every patch, there still remain some control points which can be freely moved without affecting the G^1 continuity constraints. They can be used to improve the shape of individual patches optionally.

The union of all the triangular Bézier patches is the G^1 triangular spline surface S' (e.g. Fig. 3(d), (g)).

4. Generating the triangular base mesh

In this section, we will describe a method of generating a triangular base mesh that interpolates the given implicit surface S through its vertices.

4.1. Implicit surface polygonization

The process of generating a polygon mesh M from an implicit surface S is generally known as polygonization, or tessellation. Ad hoc, if the triangles are used, it is commonly called triangulation. Modern real-time graphics environments are heavily optimized for polygon display and manipulation, so that implicit surfaces are often converted into a polygonal mesh. As a result, there have been a lot of efforts given to developing efficient algorithms for polygonization. Excellent surveys can be found in [33–35]. From a range of candidate algorithms, we chose Bloothmenthal's polygonizer [36] because of its efficiency and robustness. The resulting triangular mesh M is a piecewise-linear approximation which is homeomorphic to S.

4.2. Triangular mesh decimation

Usually the mesh M has too many triangles to serve as a base mesh. So we need to decimate M. Mesh decimation is crucial to processing on digital geometry models, such as compression, level of detail regulation, and progressive transmission. There are many different approaches, which have been surveyed in [37,38]. In our implementation, we use the quadric error metric (QEM)

algorithm [39,40], which perhaps strikes the best balance between speed, fidelity, and robustness. The resulting triangular mesh M' has fewer triangles and is still homeomorphic to S.

4.3. Triangular mesh projection

In order to ensure that the vertices of M' lie on the surface S, we use a projection process. The projection of each vertex $\mathbf{v}$ of M' onto S is computed as follows:

$$\tilde{\mathbf{v}}_0 = \mathbf{v}, \quad \text{do } \tilde{\mathbf{v}}_{k+1} \leftarrow \tilde{\mathbf{v}}_k - \frac{f(\tilde{\mathbf{v}}_k)\nabla f(\tilde{\mathbf{v}}_k)}{\| \nabla f(\tilde{\mathbf{v}}_k) \|^2} \text{ until } \frac{|f(\tilde{\mathbf{v}}_k)|}{\| \nabla f(\tilde{\mathbf{v}}_k) \|} \leqslant \frac{\varepsilon}{l}, \quad (4)$$

where ε is a user-specified precision parameter, and l is the length of a main diagonal of the bounding box of M'. In practice, we set $\varepsilon = 10^{-6}$ as the termination criterion. Usually only a few iterations of Eq. (4) are required, since $\mathbf{v}$ is already close to S. The projection process completes the preparation of the triangular base mesh B.

5. Constructing the boundary curves network

As the base mesh B is available, we may construct the boundary curves network C. For each edge of B, we will generate a quartic Bézier curve, which interpolates positions, normals, and normal curvatures of S at its endpoints.

5.1. Curvature of implicit surfaces

Let an implicit surface S be defined by $f^{-1}(c)$. For each point $\mathbf{p} \in S$, the surface unit normal vector at $\mathbf{p}$ is given by $\mathbf{n} = \left.\frac{\nabla f}{\|\nabla f\|}\right|_{\mathbf{p}}$, where $\nabla = \left[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right]^{\mathrm{T}}$. And the Hessian matrix at $\mathbf{p}$ is defined by

$$\mathbf{H}(f) = \left.\begin{bmatrix} \frac{\partial^2 f}{\partial^2 x} & \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial x \partial z} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial^2 y} & \frac{\partial^2 f}{\partial y \partial z} \\ \frac{\partial^2 f}{\partial z \partial x} & \frac{\partial^2 f}{\partial z \partial y} & \frac{\partial^2 f}{\partial^2 z} \end{bmatrix}\right|_{\mathbf{p}}.$$

From the matrix representation $\nabla\mathbf{n}$ of curvature tensor, we can derive the formula

$$\nabla\mathbf{n} = \frac{(\mathbf{I} - \mathbf{n}\mathbf{n}^{\mathrm{T}})\mathbf{H}(f)}{\|\nabla f\|}, \quad (5)$$

where $\mathbf{I}$ is a 3×3 identity matrix. The matrix $\nabla\mathbf{n}$ has three eigenvalues 0, $k_{\min}$, and $k_{\max}$, which are respectively associated with three eigenvectors $\mathbf{n}$, $\mathbf{t}_{\min}$ and $\mathbf{t}_{\max}$, where $k_{\min}$, and $k_{\max}$ are the principal curvatures, and $\mathbf{t}_{\min}$ and $\mathbf{t}_{\max}$ are the principal directions. For the umbilic points (i.e. $k_{\min}$ equals to $k_{\max}$), the principal directions are not determined uniquely, but we may choose a pair of orthogonal directions in tangent plane arbitrarily as the principal directions.

Let $\mathbf{t}$ be a given unit tangent direction. Since $\mathbf{t}_{\min}$ and $\mathbf{t}_{\max}$ form an orthonormal basis of the tangent plane, we have

$$\mathbf{t} = \mathbf{t}_{\min}\cos\theta + \mathbf{t}_{\max}\sin\theta.$$

Using the Euler formula, we can express the normal curvature along $\mathbf{t}$ as follows:

$$k_n = k_{\min}\cos^2\theta + k_{\max}\sin^2\theta. \quad (6)$$

In our algorithm, the following computations are performed:

(i) At each vertex of the base mesh B, the surface unit normal vector $\mathbf{n}$ and the Hessian matrix $\mathbf{H}(f)$ are computed.
(ii) The matrix representation $\nabla\mathbf{n}$ of curvature tensor is obtained, and its three eigenvalues and eigenvectors are computed using the QR algorithm [41].
(iii) The three eigenvalues are compared to discover $k_{\min}$ and $k_{\max}$ with associated $\mathbf{t}_{\min}$ and $\mathbf{t}_{\max}$.

5.2. Geometric Hermite interpolation

A single edge in the base mesh B has two endpoints $\mathbf{q}_0$ and $\mathbf{q}_4$ (see Fig. 4). From the implicit surface S, we can evaluate two surface unit normal vectors $\mathbf{N}_0$ and $\mathbf{N}_1$, two pairs of principal curvatures $k^0_{\min}$, $k^0_{\max}$ and $k^1_{\min}$, $k^1_{\max}$, and two pairs of principal directions $\mathbf{t}^0_{\min}$, $\mathbf{t}^0_{\max}$ and $\mathbf{t}^1_{\min}$, $\mathbf{t}^1_{\max}$, at $\mathbf{q}_0$ and $\mathbf{q}_4$ respectively. Project the vector $\overrightarrow{\mathbf{q}_0\mathbf{q}_4}$ onto two tangent planes determined by $\mathbf{N}_0$ and $\mathbf{N}_1$ and then normalize, we obtain two tangent vectors $\mathbf{t}_0$ and $\mathbf{t}_1$. Using the formula in Eq. (6), we can compute two normal curvatures k_0 and k_1.

Our aim is to find a quartic Bézier curve $\mathbf{q}(t) = \sum_{i=0}^{4}\mathbf{q}_i B^4_i(t)$, which satisfies

$$\mathbf{q}(t)\,|_{t=0} = \mathbf{q}_0, \qquad \mathbf{q}(t)\,|_{t=1} = \mathbf{q}_4,$$
$$\left.\frac{\partial\mathbf{q}(t)}{\partial t}\right|_{t=0} \parallel \mathbf{t}_0, \qquad \left.\frac{\partial\mathbf{q}(t)}{\partial t}\right|_{t=1} \parallel \mathbf{t}_1,$$

and meets two normal curvatures k_0 and k_1 at $\mathbf{q}_0$ and $\mathbf{q}_4$ respectively. Applying the formula of derivative vectors of $\mathbf{q}(t)$ at $t = 0$ and $t = 1$, we may assume

$$\mathbf{q}_1 = \mathbf{q}_0 + \lambda\mathbf{t}_0, \qquad \mathbf{q}_3 = \mathbf{q}_4 + \mu\mathbf{t}_1. \quad (7)$$

From the definition of curvature vector, we obtain two curvature vectors of $\mathbf{q}(t)$ at $\mathbf{q}_0$ and $\mathbf{q}_4$ as follows:

$$\kappa_0\mathbf{n}_0 = \frac{3\|\mathbf{t}_0 \times (\mathbf{q}_2 - \mathbf{q}_0)\|}{4\lambda^2}\mathbf{n}_0,$$
$$\kappa_1\mathbf{n}_1 = \frac{3\|\mathbf{t}_1 \times (\mathbf{q}_2 - \mathbf{q}_4)\|}{4\mu^2}\mathbf{n}_1,$$

where the $\mathbf{n}_0$ and $\mathbf{n}_1$ are the principal normals of $\mathbf{q}(t)$ at $\mathbf{q}_0$ and $\mathbf{q}_4$, and

$$\mathbf{n}_0 = \frac{[\mathbf{t}_0 \times (\mathbf{q}_2 - \mathbf{q}_0)] \times \mathbf{t}_0}{\|[\mathbf{t}_0 \times (\mathbf{q}_2 - \mathbf{q}_0)] \times \mathbf{t}_0\|},$$
$$\mathbf{n}_1 = \frac{[\mathbf{t}_1 \times (\mathbf{q}_2 - \mathbf{q}_4)] \times \mathbf{t}_1}{\|[\mathbf{t}_1 \times (\mathbf{q}_2 - \mathbf{q}_4)] \times \mathbf{t}_1\|}.$$

So normal curvatures of $\mathbf{q}(t)$ at $t = 0$ and $t = 1$ are

$$\kappa_0(\mathbf{n}_0 \cdot \mathbf{N}_0) = k_0, \qquad \kappa_1(\mathbf{n}_1 \cdot \mathbf{N}_1) = k_1.$$

With further calculation, as described in Appendix A, we have the compact formulas:

$$k_0 = \frac{3(\mathbf{q}_2 - \mathbf{q}_0)\cdot\mathbf{N}_0}{4\lambda^2}, \qquad k_1 = \frac{3(\mathbf{q}_2 - \mathbf{q}_4)\cdot\mathbf{N}_1}{4\mu^2}. \quad (8)$$

5.3. Equality constrained least-squares fitting

As $\mathbf{q}_0$ and $\mathbf{q}_4$ are given by B, and $\mathbf{q}_1$ and $\mathbf{q}_3$ are obtained by Eq. (7), in order to determine $\mathbf{q}(t)$ fully, we need to compute the λ, μ and $\mathbf{q}_2$ which should fulfill Eq. (8).

Let e be the edge of B that corresponds to $\mathbf{q}(t)$. Our goal is to approximate the projection of e onto S by $\mathbf{q}(t)$. We parameterize e as $\mathbf{e}(t) = (1-t)\mathbf{q}_0 + t\mathbf{q}_4$, $t \in [0, 1]$, and sample some points $\mathbf{e}_j = \mathbf{e}(t_j)$, $j = 0, 1, \ldots, M$ (see Fig. 5(a)). Using the process expressed as Eq. (4), we project $\{\mathbf{e}_j\}_{j=0}^{M}$ onto S and obtain target points $\{\mathbf{p}_j\}_{j=0}^{M}$.

The equality constrained least-squares fitting problem can be stated as:

$$\min_{\lambda,\mu,\mathbf{q}_2} \sum_{j=0}^{M} \|\mathbf{q}_0 B^4_0(t_j) + (\mathbf{q}_0 + \lambda\mathbf{t}_0)B^4_1(t_j) + \mathbf{q}_2 B^4_2(t_j) + (\mathbf{q}_4 + \mu\mathbf{t}_1)B^4_3(t_j) + \mathbf{q}_4 B^4_4(t_j) - \mathbf{p}_j\|^2, \quad (9a)$$

$$\text{subject to } \begin{cases} 4\lambda^2 k_0 - 3(\mathbf{q}_2 - \mathbf{q}_0)\cdot\mathbf{N}_0 = 0, \\ 4\mu^2 k_1 - 3(\mathbf{q}_2 - \mathbf{q}_4)\cdot\mathbf{N}_1 = 0. \end{cases} \quad (9b)$$

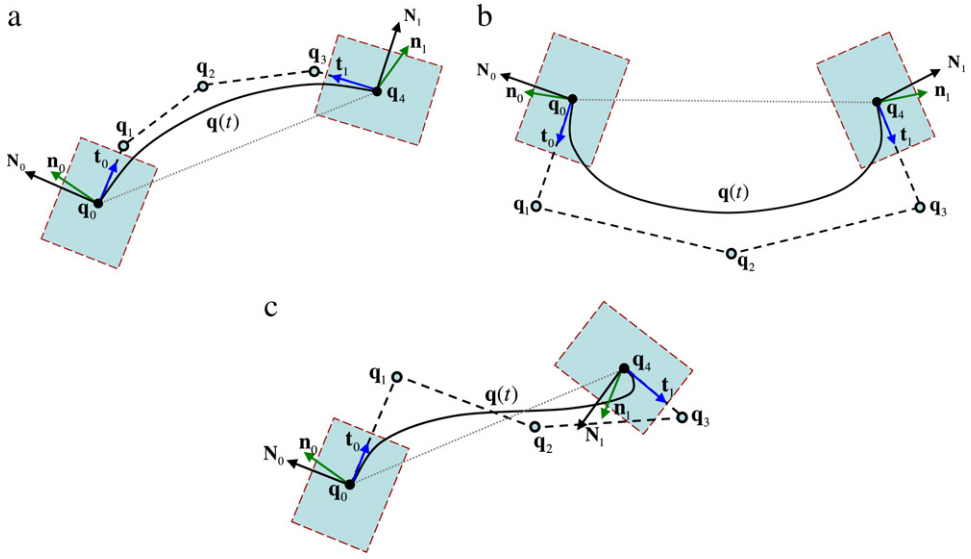


Fig. 4. Geometric Hermite interpolation: (a) convex case; (b) concave case; (c) convex and concave mixed case.

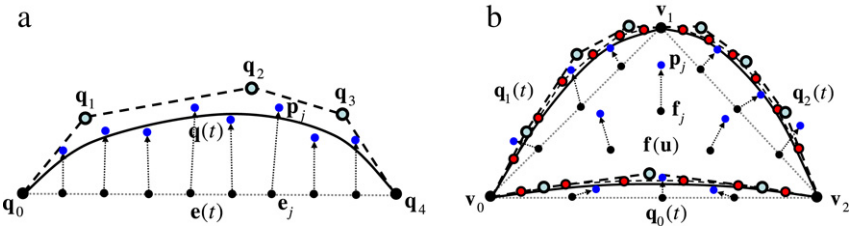


Fig. 5. Local fitting schema: (a) the quartic Bézier curve $\mathbf{q}(t)$; (b) the triangular Bézier patch $\mathbf{b}(\mathbf{u})$.

Because the equality constraints are quadratic, this is a nonlinear constrained optimization problem.

In order to solve the problem (9) efficiently, we need to provide a good initial guess, consisting of λ^0, μ^0 and $\mathbf{q}_2^0$ which is close to the optimal solution λ^*, μ^* and $\mathbf{q}_2^*$. In our implementation, we accomplish this in the following way:

- Solve the least-squares fitting problem Eq. (9) without the equality constraints (i.e. Eq. (9a)), to obtain $\mathbf{q}_2^0$.
- Substitute $\mathbf{q}_2^0$ into the equality constraints Eq. (9b), to gain λ^0 and μ^0.

Let the object function of problem (9) be $F(\mathbf{x})$, where $\mathbf{x}$ is $[\lambda, \mu, x, y, z]^{\mathrm{T}}$ and $\mathbf{q}_2 = [x, y, z]^{\mathrm{T}}$. The least-squares fitting problem Eq. (9a) can be solved by $\nabla F(\mathbf{x}) = \mathbf{0}$, which is a linear system of the form $\mathbf{A}\mathbf{x} = \mathbf{b}$, where $\mathbf{A}$ is a 5×5 matrix.

We can obtain $\mathbf{q}_2^0$ from $\mathbf{A}\mathbf{x} = \mathbf{b}$. Then λ^0 and μ^0 can be chosen as follows:

$$\lambda^0 = \begin{cases} \sqrt{\dfrac{3(\mathbf{q}_2^0 - \mathbf{q}_0)\cdot\mathbf{N}_0}{4k_0}}, & \text{if } \dfrac{3(\mathbf{q}_2^0 - \mathbf{q}_0)\cdot\mathbf{N}_0}{4k_0} \geqslant 0, \\ 0, & \text{otherwise;} \end{cases}$$

$$\mu^0 = \begin{cases} \sqrt{\dfrac{3(\mathbf{q}_2^0 - \mathbf{q}_4)\cdot\mathbf{N}_1}{4k_1}}, & \text{if } \dfrac{3(\mathbf{q}_2^0 - \mathbf{q}_4)\cdot\mathbf{N}_1}{4k_1} \geqslant 0, \\ 0, & \text{otherwise.} \end{cases}$$

Using the initial guess λ^0, μ^0 and $\mathbf{q}_2^0$, we need to seek the optimal solution λ^*, μ^* and $\mathbf{q}_2^*$. For this purpose, we employ the local sequential quadratic programming (SQP) method [42]. The basic steps are:

- Convert the problem (9) into the KKT optimality condition by the Lagrange multiplier.
- Solve the KKT optimality condition using the Newton–Lagrange method.

Let the Lagrange multiplier is $\mathbf{r} = [r_1, r_2]^{\mathrm{T}}$, and the constraint vector function is $\mathbf{c}(\mathbf{x}) = [c_1(\mathbf{x}), c_2(\mathbf{x})]^{\mathrm{T}}$, where $c_1(\mathbf{x}) = 4\lambda^2 k_0 - 3(\mathbf{q}_2 - \mathbf{q}_0)\cdot\mathbf{N}_0$ and $c_2(\mathbf{x}) = 4\mu^2 k_1 - 3(\mathbf{q}_2 - \mathbf{q}_4)\cdot\mathbf{N}_1$. The Lagrangian function for this problem (9) is $\mathcal{L}(\mathbf{x}, \mathbf{r}) = F(\mathbf{x}) - \mathbf{r}^{\mathrm{T}}\mathbf{c}(\mathbf{x})$, and any solution $(\mathbf{x}^*, \mathbf{r}^*)$ of problem (9) must satisfy the KKT optimality condition:

$$\begin{bmatrix} \nabla F(\mathbf{x}) - \mathbf{G}(\mathbf{x})\mathbf{r} \\ \mathbf{c}(\mathbf{x}) \end{bmatrix} = \mathbf{0}, \tag{10}$$

where $\nabla = [\frac{\partial}{\partial\lambda}, \frac{\partial}{\partial\mu}, \frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}]^{\mathrm{T}}$ and $\mathbf{G}(\mathbf{x}) = [\nabla c_1(\mathbf{x}), \nabla c_2(\mathbf{x})]$ is a 5×2 matrix. This yields a system of 7 equations in 7 unknowns (5 from $\mathbf{x}$ and 2 from $\mathbf{r}$), and can be solved by the Newton–Lagrange method. The Hessian matrix of the Lagrangian function is denoted by

$$\begin{aligned} \mathbf{W}(\mathbf{x}, \mathbf{r}) &= \nabla^2_{\mathbf{xx}}\mathcal{L}(\mathbf{x}, \mathbf{r}) \\ &= 2\mathbf{A} - r_1 \begin{bmatrix} 8k_0 & & \\ & 0 & \\ & & \mathbf{0} \end{bmatrix}_{5\times5} - r_2 \begin{bmatrix} 0 & & \\ & 8k_1 & \\ & & \mathbf{0} \end{bmatrix}_{5\times5}. \end{aligned}$$

At the kth iteraton, we solve the KKT system

$$\begin{bmatrix} \mathbf{W}_k & -\mathbf{G}_k \\ \mathbf{G}_k^{\mathrm{T}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u}_k \\ \mathbf{v}_k \end{bmatrix} = \begin{bmatrix} -\nabla F_k \\ -\mathbf{c}_k \end{bmatrix}, \tag{11}$$

where $\mathbf{W}_k = \mathbf{W}(\mathbf{x}_k, \mathbf{r}_k)$, $\mathbf{G}_k = \mathbf{G}(\mathbf{x}_k)$, $\nabla F_k = \nabla F(\mathbf{x}_k)$ and $\mathbf{c}_k = \mathbf{c}(\mathbf{x}_k)$. This procedure can be summarized as follows:

Algorithm 5.1 (*Local SQP Algorithm*).

Set the initial values $\mathbf{x}^0 = (\lambda^0, \mu^0, \mathbf{q}_2^0)$, $\mathbf{r}^0 = (0, 0)$;
for $k = 0, 1, 2, \ldots, M$ **do**
Evaluate $\mathbf{W}_k$, $\mathbf{G}_k$, ∇F_k, $\mathbf{c}_k$ at $\mathbf{x}_k$, $\mathbf{r}_k$;
Solve Eq. (11) to obtain $\mathbf{u}_k$ and $\mathbf{v}_k$;

$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \mathbf{u}_k$ and $\mathbf{r}_{k+1} \leftarrow \mathbf{v}_k$;
$res = \|\mathbf{x}_{k+1} - \mathbf{x}_k\| + \|\mathbf{r}_{k+1} - \mathbf{r}_k\|$;
if $res < \varepsilon$ **then** break;
end

If the initial guess λ^0, μ^0 and $\mathbf{q}_2^0$ is far from the optimal solution λ^*, μ^* and $\mathbf{q}_2^*$, the trust-region strategy [42] should be used; but in practice, this does not seem to be necessary, because the local SQP method works well and efficiently as we can see from Table 3.

6. Solving for G^1 triangular spline surfaces

We have now constructed the triangular base mesh B and the boundary curves network C. In this section, we will show how to obtain a G^1 triangular spline surface.

6.1. G^0 triangular surfaces

A single facet f of B has three vertices, $\mathbf{v}_0$, $\mathbf{v}_1$ and $\mathbf{v}_2$. We can parameterize the facet f with $\mathbf{f}(\mathbf{u}) = u\mathbf{v}_0 + v\mathbf{v}_1 + w\mathbf{v}_2$, where $\mathbf{u} = (u, v, w)$ is the barycentric coordinates with respect to the vertices of f. We sample some points $\mathbf{f}_j = \mathbf{f}(\mathbf{u}_j)$, $j = 0, 1, \ldots, M$ (see Fig. 5(b)). Using the process expressed as Eq. (4), we project $\{\mathbf{f}_j\}_{j=0}^M$ onto S and obtain target points $\{\mathbf{p}_j\}_{j=0}^M$.

Our aim is to fit a triangular Bézier patch $T : \mathbf{b}(\mathbf{u}) = \sum_{|\mathbf{i}|=n} \mathbf{b}_\mathbf{i} B_\mathbf{i}^n(\mathbf{u})$ to the projection of f onto S, represented by $\{\mathbf{p}_j\}_{j=0}^M$. In addition, we require the boundaries of $\mathbf{b}(\mathbf{u})$ to meet the corresponding curve from C. To obtain such patches, we elevate the relevant Bézier curves $\mathbf{q}(t)$ of C from degree $m = 4$ to $n = 7$, and then use the elevated control points as the boundary control points of $\mathbf{b}(\mathbf{u})$. And the remaining control points of $\mathbf{b}(\mathbf{u})$ can be determined by least-squares fitting of the sample points by means of the following minimization:

$$\min_{\{\mathbf{b}_\mathbf{i}\}} \sum_{j=0}^{M} \|\mathbf{b}(\mathbf{u}_j) - \mathbf{p}_j\|^2, \tag{12}$$

or

$$\min_{\{\mathbf{b}_\mathbf{i}\}} \sum_{j=0}^{M} \|\mathbf{b}(\mathbf{u}_j) - \mathbf{p}_j\|^2 + \alpha \iint_\Delta \left[\left\| \frac{\partial^2 \mathbf{b}}{\partial u^2} \right\|^2 + 2 \left\| \frac{\partial^2 \mathbf{b}}{\partial u \partial v} \right\|^2 + \left\| \frac{\partial^2 \mathbf{b}}{\partial v^2} \right\|^2 \right] du dv, \tag{13}$$

where $\Delta = \{\mathbf{u} \mid u + v + w = 1,\ u, v, w \geqslant 0\}$ is the parametric domain and $\alpha \geqslant 0$ is a weight.

Using either of them, we can obtain a triangular Bézier patch T_i for each facet of B. The union of $\{T_i\}$ is a G^0 triangular surface which interpolates the boundary curves network C.

6.2. Revisiting the G^1 continuity constraints

In order to obtain a G^1 triangular spline surface, we must make each pair of adjacent patches, which share a common boundary $\mathbf{q}(t)$, satisfy the G^1 continuity constraints. We separate all the G^1 continuity constraints into two types: vertex and edge constraints. In the case of $r = 1$, $s = 4$, $m = 4$ and $n = 7$, i.e. (Table 2), the constraints corresponding to $k = 0$, 1, 6 and 7 are vertex constraints, and the constraints corresponding to $k = 2$, 3, 4 and 5 are edge constraints.

The G^1 triangular spline surface can be constructed as follows:

(i) For each vertex of the base mesh B, compose a vertex G^1 continuity system and solve it. In this step, all the control points related to the twist vectors at the corners of triangular Bézier patches are determined.
(ii) For each edge of the base mesh B, compose an edge G^1 continuity system and solve it. In this step, all the off-boundary control points of two adjacent triangular Bézier patches are determined.
(iii) In every triangular Bézier patch, the inner control points, which are not involved in the G^1 continuity constraints, can be regulated optionally.

6.3. Vertex G^1 continuity system

Let $\mathbf{o}$ is a vertex of the base mesh B, at which the n boundary Bézier curves $\{\mathbf{q}^i(t)\}_{i=1}^n$ of n triangular Bézier patches $\{\mathbf{Q}_i\}_{i=1}^n$ meet (see Fig. 6(a)). The control points of quartic boundary curve $\mathbf{q}^i(t)$ are $\mathbf{q}_0^i, \mathbf{q}_1^i, \ldots, \mathbf{q}_4^i$ and the elevated control points are $\mathbf{c}_0^i, \mathbf{c}_1^i, \ldots, \mathbf{c}_7^i$. The control points $\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n$ that are related to the twist vectors at the corner $\mathbf{o}$, need to satisfy the G^1 continuity constraints corresponding to $k = 1$ or 6. All these constraints compose a linear system which can be written as follows:

$$6 \begin{bmatrix} \bar{\lambda}_0^1 & \lambda_0^1 & 0 & \cdots & 0 & 0 \\ 0 & \bar{\lambda}_0^2 & \lambda_0^2 & \cdots & 0 & 0 \\ 0 & 0 & \bar{\lambda}_0^3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \bar{\lambda}_0^{n-1} & \lambda_0^{n-1} \\ \lambda_0^n & 0 & 0 & \cdots & 0 & \bar{\lambda}_0^n \end{bmatrix} \begin{bmatrix} \mathbf{t}_1 \\ \mathbf{t}_2 \\ \mathbf{t}_3 \\ \vdots \\ \mathbf{t}_{n-1} \\ \mathbf{t}_n \end{bmatrix} = \begin{bmatrix} \mathbf{rhs}_1 \\ \mathbf{rhs}_2 \\ \mathbf{rhs}_3 \\ \vdots \\ \mathbf{rhs}_{n-1} \\ \mathbf{rhs}_n \end{bmatrix}, \tag{14}$$

where $\{\lambda_0^i\}_{i=1}^n$ are the coefficients of the scalar weight function $\lambda(t)$ that satisfies the G^1 continuity constraints corresponding to $k = 0$ or 7:

$$\bar{\lambda}_0^i \mathbf{c}_1^{i-1} + \lambda_0^i \mathbf{c}_1^{i+1} = \bar{\mu}_0^i \mathbf{o} + \mu_0^i \mathbf{q}_1^i, \tag{15}$$

and $\{\mathbf{rhs}_i\}_{i=1}^n$ are the residual vectors:

$$\mathbf{rhs}_i = -(\bar{\lambda}_1^i \mathbf{c}_1^{i-1} + \lambda_1^i \mathbf{c}_1^{i+1}) + 3(\bar{\mu}_0^i \mathbf{q}_1^i + \mu_0^i \mathbf{q}_2^i) + 4(\bar{\mu}_1^i \mathbf{o} + \mu_1^i \mathbf{q}_1^i), \tag{16}$$

where the subscript $i := (n + i) \bmod n$ is applied.

Let the coefficient matrix and the right-hand vector array of the linear system of Eq. (14) be $\mathbf{A}$ and $\mathbf{b}$ respectively. The determinant of $\mathbf{A}$ is

$$\det \mathbf{A} = 6^n \left[\prod_{i=1}^n \bar{\lambda}_0^i + (-1)^{n+1} \prod_{i=1}^n \lambda_0^i \right].$$

Furthermore, in Appendix B we show that $\prod_{i=1}^n \bar{\lambda}_0^i = \prod_{i=1}^n \lambda_0^i$. So we get

$$\det \mathbf{A} = \begin{cases} 6^n \cdot 2 \prod_{i=1}^n \lambda_0^i, & \text{if } n \text{ is odd}, \\ 0, & \text{if } n \text{ is even}. \end{cases}$$

Without loss of generality, we can assume that $\lambda_0^i \neq 0$ for $i = 1, 2, \ldots, n$, meaning that any two adjacent boundary Bézier curves do not coincide. Under this condition, we have

$$\operatorname{rank}(\mathbf{A}) = \begin{cases} n, & \text{if } n \text{ is odd}, \\ n - 1, & \text{if } n \text{ is even}. \end{cases}$$

If n is odd, we can obtain a unique solution from Eq. (14) for the given values of $\{\mathbf{rhs}_i\}_{i=1}^n$. The situation in which n is even can be further divided into two cases:

(i) If $\operatorname{rank}(\mathbf{A}, \mathbf{b}) = \operatorname{rank}(\mathbf{A})$, there are many solutions.
(ii) If $\operatorname{rank}(\mathbf{A}, \mathbf{b}) \neq \operatorname{rank}(\mathbf{A})$, there is no solution.

So, in order to have solutions, the $\{\mathbf{rhs}_i\}_{i=1}^n$ must satisfy

$$(-1)\bar{\lambda}_0^2 \bar{\lambda}_0^3 \cdots \bar{\lambda}_0^n \mathbf{rhs}_1 + (-1)^2 \lambda_0^1 \bar{\lambda}_0^3 \cdots \bar{\lambda}_0^n \mathbf{rhs}_2 + \cdots + (-1)^n \lambda_0^1 \lambda_0^2 \cdots \lambda_0^{n-1} \mathbf{rhs}_n = \mathbf{0}.$$

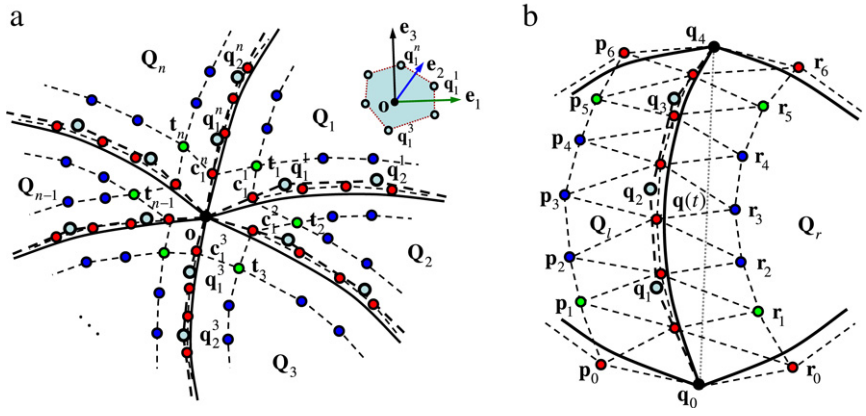


Fig. 6. G^1 continuity systems: (a) vertex G^1 continuity system; (b) edge G^1 continuity system. The vertices of base mesh are shown as black points, and the original and elevated control points of boundary curves are shown as light-blue and red points respectively. The green points are determined by vertex G^1 continuity systems, and the blue points are determined by edge G^1 continuity systems. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

This is a necessary and sufficient condition, which is usually called the vertex enclosure constraint or twist compatibility condition [20,27]. As explained in Appendix C, this constraint is related to the boundary Bézier curves. And we prove that the vertex enclosure constraint is satisfied automatically if the boundary Bézier curves $\{\mathbf{q}^i(t)\}_{i=1}^n$ be constructed as described in Section 5, so that the vertex G^1 continuity system is consistent. This means that the Eq. (14) always have solutions.

In Section 6.1 we constructed a G^0 triangular surface. Let's take some control points from this surface and denote them as $\mathbf{t}_1^0, \mathbf{t}_2^0, \ldots, \mathbf{t}_n^0$ corresponding to $\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n$. In addition, to obtain a pleasing shape, we make the coefficients $\{\mu_1^i\}_{i=1}^n$ variable and use them as shape parameters. Using $\mathbf{t}_1^0, \mathbf{t}_2^0, \ldots, \mathbf{t}_n^0$ as an initial guess, our goal is to compute $\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n$ from Eq. (14). For simplicity of description, we introduce a local coordinate system $\{\mathbf{o}; \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$, with its origin at the vertex $\mathbf{o}$ and the axes $\mathbf{e}_1 = \mathbf{t}_{\min}$, $\mathbf{e}_2 = \mathbf{t}_{\max}$, $\mathbf{e}_3 = \mathbf{N}$, where $\mathbf{t}_{\min}$ and $\mathbf{t}_{\max}$ are the two principal directions and $\mathbf{N}$ is the unit normal at $\mathbf{o}$. Under this coordinate system, we move $\{\mu_1^i\}_{i=1}^n$ to the left-hand side of Eq. (14) and rewrite it in component form as follows:

$$\begin{bmatrix} \mathbf{A} & \mathbf{0} & \mathbf{0} & \mathbf{Q}_1 \\ \mathbf{0} & \mathbf{A} & \mathbf{0} & \mathbf{Q}_2 \\ \mathbf{0} & \mathbf{0} & \mathbf{A} & \mathbf{Q}_3 \end{bmatrix} \begin{bmatrix} \mathbf{T}_1 \\ \mathbf{T}_2 \\ \mathbf{T}_3 \\ \mathbf{U} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \\ \mathbf{R}_3 \end{bmatrix}, \tag{17}$$

where

- $\mathbf{Q}_1$, $\mathbf{Q}_2$ and $\mathbf{Q}_3$ are the three components of the diagonal matrix
$$\begin{bmatrix} -4(\mathbf{q}_1^1 - \mathbf{o}) & & & \\ & -4(\mathbf{q}_1^2 - \mathbf{o}) & & \\ & & \ddots & \\ & & & -4(\mathbf{q}_1^n - \mathbf{o}) \end{bmatrix},$$
- $\mathbf{T}_1$, $\mathbf{T}_2$ and $\mathbf{T}_3$ are the three components of the vector array $[\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n]^{\mathrm{T}}$,
- $\mathbf{R}_1$, $\mathbf{R}_2$ and $\mathbf{R}_3$ are the three components of the vector array
$$\begin{bmatrix} -(\bar{\lambda}_1^1 \mathbf{c}_1^n + \lambda_1^1 \mathbf{c}_1^2) + 3(\bar{\mu}_0^1 \mathbf{q}_1^1 + \mu_0^1 \mathbf{q}_2^1) + 4\mathbf{o} \\ -(\bar{\lambda}_1^2 \mathbf{c}_1^1 + \lambda_1^2 \mathbf{c}_1^3) + 3(\bar{\mu}_0^2 \mathbf{q}_1^2 + \mu_0^2 \mathbf{q}_2^2) + 4\mathbf{o} \\ \vdots \\ -(\bar{\lambda}_1^n \mathbf{c}_1^{n-1} + \lambda_1^n \mathbf{c}_1^1) + 3(\bar{\mu}_0^n \mathbf{q}_1^n + \mu_0^n \mathbf{q}_2^n) + 4\mathbf{o} \end{bmatrix},$$
- $\mathbf{U}$ is $[\mu_1^1, \mu_1^2, \ldots, \mu_1^n]^{\mathrm{T}}$.

Because $(\mathbf{q}_1^i - \mathbf{o}) \perp \mathbf{N}$ for $i = 1, 2, \ldots, n$, the third component $\mathbf{Q}_3$ is a zero matrix.

As explained in Appendix C, Eq. (14) and (17) can have one or many solutions if the boundary Bézier curves $\{\mathbf{q}^i(t)\}_{i=1}^n$ are constructed as described in Section 5. In order to obtain the best solution from the initial guess, we separate Eq. (17) into two subproblems as follows:

1. The tangent component subproblem:
$$\min_{\mathbf{T}_1, \mathbf{T}_2} \left\| \begin{bmatrix} \mathbf{T}_1 \\ \mathbf{T}_2 \end{bmatrix} - \begin{bmatrix} \mathbf{T}_1^0 \\ \mathbf{T}_2^0 \end{bmatrix} \right\|^2 \quad \text{subject to}$$
$$\begin{bmatrix} \mathbf{A} & \mathbf{0} & \mathbf{Q}_1 \\ \mathbf{0} & \mathbf{A} & \mathbf{Q}_2 \end{bmatrix} \begin{bmatrix} \mathbf{T}_1 \\ \mathbf{T}_2 \\ \mathbf{U} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \end{bmatrix}. \tag{18}$$
2. The normal component subproblem:
   (a) if n is odd, then $\mathrm{rank}(\mathbf{A}) = n$ and $\mathbf{T}_3 = \mathbf{A}^{-1}\mathbf{R}_3$;
   (b) if n is even, then $\mathrm{rank}(\mathbf{A}) = n - 1$ and
$$\min_{\mathbf{T}_3} \left\| \mathbf{T}_3 - \mathbf{T}_3^0 \right\|^2 \quad \text{subject to } \mathbf{A}\mathbf{T}_3 = \mathbf{R}_3. \tag{19}$$

As proved in Appendix C, the tangent and normal component subproblems have a unique solution. By solving them, we can get the $\mathbf{T}_1$, $\mathbf{T}_2$ and $\mathbf{T}_3$, which can then be transformed to previous coordinate system to obtain $\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n$.

6.4. Edge G^1 continuity system

Let $\mathbf{q}(t)$ be a single boundary Bézier curve, corresponding to an edge of the base mesh B (see Fig. 6(b)). Two triangular Bézier patches $\mathbf{Q}_l$ and $\mathbf{Q}_r$ join at $\mathbf{q}(t)$. The control points $\{\mathbf{p}_i\}_{i=0}^6$ and $\{\mathbf{r}_i\}_{i=0}^6$ are the first-left and first-right rows of the control net along $\mathbf{q}(t)$ from $\mathbf{Q}_l$ and $\mathbf{Q}_r$ respectively. They need to satisfy the G^1 continuity constraint system of (Table 2). The $\mathbf{p}_0$, $\mathbf{p}_6$, $\mathbf{r}_0$ and $\mathbf{r}_6$ are inherited from the boundary Bézier curves, and the $\mathbf{p}_1$, $\mathbf{p}_5$, $\mathbf{r}_1$ and $\mathbf{r}_5$ are determined by the vertex G^1 continuity system. So we only need to make $\{\mathbf{p}_i\}_{i=2}^4$ and $\{\mathbf{r}_i\}_{i=2}^4$ satisfy the G^1 continuity constraints corresponding to $k = 2, 3, 4$ and 5. Together these constraints constitute a linear system which can be written as follows:

$$\begin{bmatrix} 15\bar{\lambda}_0 & 15\lambda_0 & 0 & 0 & 0 & 0 \\ 15\bar{\lambda}_1 & 15\lambda_1 & 20\bar{\lambda}_0 & 20\lambda_0 & 0 & 0 \\ 0 & 0 & 20\bar{\lambda}_1 & 20\lambda_1 & 15\bar{\lambda}_0 & 15\lambda_0 \\ 0 & 0 & 0 & 0 & 15\bar{\lambda}_1 & 15\lambda_1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_2 \\ \mathbf{r}_2 \\ \mathbf{p}_3 \\ \mathbf{r}_3 \\ \mathbf{p}_4 \\ \mathbf{r}_4 \end{bmatrix} = \begin{bmatrix} \mathbf{rsh}_1 \\ \mathbf{rsh}_2 \\ \mathbf{rsh}_3 \\ \mathbf{rsh}_4 \end{bmatrix}, \tag{20}$$

where

$$\begin{aligned}
\mathbf{rhs}_1 &= -6(\bar{\lambda}_1\mathbf{p}_1+\lambda_1\mathbf{r}_1)+3(\bar{\mu}_0\mathbf{q}_2+\mu_0\mathbf{q}_3)\\
&\quad +12(\bar{\mu}_1\mathbf{q}_1+\mu_1\mathbf{q}_2)+6(\bar{\mu}_2\mathbf{q}_0+\mu_2\mathbf{q}_1),\\
\mathbf{rhs}_2 &= (\bar{\mu}_0\mathbf{q}_3+\mu_0\mathbf{q}_4)+12(\bar{\mu}_1\mathbf{q}_2+\mu_1\mathbf{q}_3)\\
&\quad +18(\bar{\mu}_2\mathbf{q}_1+\mu_2\mathbf{q}_2)+4(\bar{\mu}_3\mathbf{q}_0+\mu_3\mathbf{q}_1),\\
\mathbf{rhs}_3 &= 4(\bar{\mu}_1\mathbf{q}_3+\mu_1\mathbf{q}_4)+18(\bar{\mu}_2\mathbf{q}_2+\mu_2\mathbf{q}_3)\\
&\quad +12(\bar{\mu}_3\mathbf{q}_1+\mu_3\mathbf{q}_2)+(\bar{\mu}_4\mathbf{q}_0+\mu_4\mathbf{q}_1),\\
\mathbf{rhs}_4 &= -6(\bar{\lambda}_0\mathbf{p}_5+\lambda_0\mathbf{r}_5)+6(\bar{\mu}_2\mathbf{q}_3+\mu_2\mathbf{q}_4)\\
&\quad +12(\bar{\mu}_3\mathbf{q}_2+\mu_3\mathbf{q}_3)+3(\bar{\mu}_4\mathbf{q}_1+\mu_4\mathbf{q}_2),
\end{aligned} \tag{21}$$

and the coefficients λ_0, λ_1, μ_0, μ_1, μ_3 and μ_4 are already determined by the G^1 continuity constraints corresponding to $k=0$, 1, 6 and 7, leaving μ_2 as the only undetermined coefficient of scalar weight functions.

Let the coefficient matrix and the right-hand vector array of the linear system Eq. (20) be $\mathbf{A}$ and $\mathbf{b}$ respectively. In Appendix D, we prove that

$$\operatorname{rank}(\mathbf{A})=\begin{cases}4 & \text{if } \lambda_0\neq\lambda_1,\\ 3 & \text{otherwise.}\end{cases}$$

In order to have solutions for the latter case, $\{\mathbf{rhs}_i\}_{i=1}^4$ must satisfy

$$\mathbf{rsh}_1-\mathbf{rsh}_2+\mathbf{rsh}_3-\mathbf{rsh}_4=\mathbf{0}. \tag{22}$$

Substituting Eq. (21) and $\lambda_0=\lambda_1$ into Eq. (22), we have

$$(\mu_0-4\mu_1+6\mu_2-4\mu_3+\mu_4)(-\mathbf{q}_0+4\mathbf{q}_1-6\mathbf{q}_2+4\mathbf{q}_3-\mathbf{q}_4)=\mathbf{0}.$$

Because μ_2 is free, it is possible to satisfy the Eq. (22) by setting

$$\mu_2=\frac{-\mu_0+4\mu_1+4\mu_3-\mu_4}{6}. \tag{23}$$

So Eq. (20) always has solutions.

As we did in the previous section, take some control points from the G^0 triangular surface again and denote them as $\{\mathbf{p}_i^0\}_{i=2}^4$, $\{\mathbf{r}_i^0\}_{i=2}^4$ corresponding to $\{\mathbf{p}_i\}_{i=2}^4$ and $\{\mathbf{r}_i\}_{i=2}^4$. As a matter of convenience, we let $\mathbf{x}=[\mathbf{p}_2,\mathbf{r}_2,\mathbf{p}_3,\mathbf{r}_3,\mathbf{p}_4,\mathbf{r}_4]^{\mathrm{T}}$ and $\mathbf{x}^0=[\mathbf{p}_2^0,\mathbf{r}_2^0,\mathbf{p}_3^0,\mathbf{r}_3^0,\mathbf{p}_4^0,\mathbf{r}_4^0]^{\mathrm{T}}$. With the initial guess $\mathbf{x}^0$ and the variable shape parameter μ_2, our goal is to compute $\mathbf{x}$ from Eq. (20). This problem can be solved as follows:

(i) If $\lambda_0=\lambda_1$, then
- Compute μ_2 from Eq. (23).
- Solve the equality constrained quadratic program:

$$\min_{\mathbf{x}}\left\|\mathbf{x}-\mathbf{x}^0\right\|^2 \quad \text{subject to } \mathbf{A}\mathbf{x}=\mathbf{b}. \tag{24}$$

(ii) If $\lambda_0\neq\lambda_1$, then solve the equality constrained quadratic program:

$$\min_{\mathbf{x},\mu_2}\left\|\mathbf{x}-\mathbf{x}^0\right\|^2 \quad \text{subject to } (\mathbf{A},\mathbf{q}')\begin{pmatrix}\mathbf{x}\\ \mu_2\end{pmatrix}=\mathbf{b}', \tag{25}$$

where
- $\mathbf{q}'$ is the vector array $\left[6(\mathbf{q}_0-\mathbf{q}_1),\,18(\mathbf{q}_1-\mathbf{q}_2),\,18(\mathbf{q}_2-\mathbf{q}_3),\,6(\mathbf{q}_3-\mathbf{q}_4)\right]^{\mathrm{T}}$,
- $\mathbf{b}'$ is the vector array

$$\begin{bmatrix}-6(\bar{\lambda}_1\mathbf{p}_1+\lambda_1\mathbf{r}_1)+3(\bar{\mu}_0\mathbf{q}_2+\mu_0\mathbf{q}_3)+12(\bar{\mu}_1\mathbf{q}_1+\mu_1\mathbf{q}_2)+6\mathbf{q}_0\\ (\bar{\mu}_0\mathbf{q}_3+\mu_0\mathbf{q}_4)+12(\bar{\mu}_1\mathbf{q}_2+\mu_1\mathbf{q}_3)+4(\bar{\mu}_3\mathbf{q}_0+\mu_3\mathbf{q}_1)+18\mathbf{q}_1\\ 4(\bar{\mu}_1\mathbf{q}_3+\mu_1\mathbf{q}_4)+12(\bar{\mu}_3\mathbf{q}_1+\mu_3\mathbf{q}_2)+(\bar{\mu}_4\mathbf{q}_0+\mu_4\mathbf{q}_1)+18\mathbf{q}_2\\ -6(\bar{\lambda}_0\mathbf{p}_5+\lambda_0\mathbf{r}_5)+12(\bar{\mu}_3\mathbf{q}_2+\mu_3\mathbf{q}_3)+3(\bar{\mu}_4\mathbf{q}_1+\mu_4\mathbf{q}_2)+6\mathbf{q}_3\end{bmatrix}.$$

From the theory of equality constrained quadratic program, the above problem has a unique solution. We have now determined all the control points that are related with the G^1 continuity constraints.

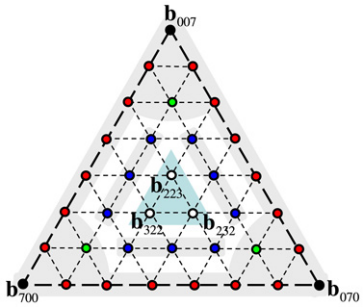


Fig. 7. Unconstrained inner control points.

6.5. Unconstrained inner control points

In every triangular Bézier patch $T:\ \mathbf{b}(\mathbf{u})=\sum_{|\mathbf{i}|=n}\mathbf{b}_{\mathbf{i}}B_{\mathbf{i}}^n(\mathbf{u})$, there remain some inner control points which are not involved in the G^1 continuity constraints. In the case of $r=1, s=4, m=4$ and $n=7$, these control points are $\mathbf{b}_{322}$, $\mathbf{b}_{232}$ and $\mathbf{b}_{223}$ (see Fig. 7). They can be used to improve the local shape of their patch optionally.

7. Analysis of the algorithm

In this section, we will present the underlying factors which determine our choice of boundary Bézier curves, scalar weight functions and triangular Bézier patches, together with a rough analysis of time complexity.

7.1. The degree of the boundary Bézier curves

Choosing the degree of boundary Bézier curve $\mathbf{q}(t)=\sum_{i=0}^m\mathbf{q}_iB_i^m(t)$ is the most important factor in the design of algorithm. There are several mutually dependent considerations. One is the solvability of the equality constrained least-squares fitting problem of Eq. (9). Another is the consistency of vertex G^1 continuity system, which depends on $\mathbf{q}_2$, as explained in Appendix C and also by Du and Schmitt [43]. In addition, the shape of boundary Bézier curve $\mathbf{q}(t)$ greatly influence the quality of the final G^1 surface. These considerations suggest that the degree m of $\mathbf{q}(t)$ should be as high as possible. On the other hand, we want to keep the degree n of triangular Bézier patches as low as possible. In Section 3.3, we assume that $r+n=s+m$. Therefore, if we fix r and s, it will lead to a contradiction on the choices of m and n. As we will explained, the best choice seems to be $m=4$ and $n=7$.

With $m=4$, we can show that the equality constrained least-squares fitting problem of Eq. (9) is solvable except for three extraordinary situations. Using the notation of Section 5.3, and define

$$\begin{aligned}
H_0&=\left\{\mathbf{q}_2\in\mathbb{R}^3:\operatorname{sgn}(k_0)(\mathbf{q}_2-\mathbf{q}_0)\cdot\mathbf{N}_0\geqslant 0\right\},\\
H_1&=\left\{\mathbf{q}_2\in\mathbb{R}^3:\operatorname{sgn}(k_1)(\mathbf{q}_2-\mathbf{q}_4)\cdot\mathbf{N}_1\geqslant 0\right\}.
\end{aligned} \tag{26}$$

It can be easily seen that the solvability of the equality constrained least-squares fitting problem of Eq. (9) depends on whether the intersection of H_0 and H_1 is an empty set or not. The geometrical viewpoint makes it obvious that the intersection set of H_0 and H_1 is not empty if $\mathbf{N}_0\not\parallel\mathbf{N}_1$. Exceptional cases only occur when $\mathbf{N}_0\parallel\mathbf{N}_1$ and $\mathbf{q}_0$ and $\mathbf{q}_4$ are located as shown in Fig. 8, and k_0 and k_1 have the signs given in that figure. And if they occur, we need to subdivide this boundary Bézier curve $\mathbf{q}(t)$. In practice, we have never encountered these extraordinary situations because of their specialization.

However, if we decrease m to 3, then the situation changes drastically, and there is no longer enough freedom to regulate

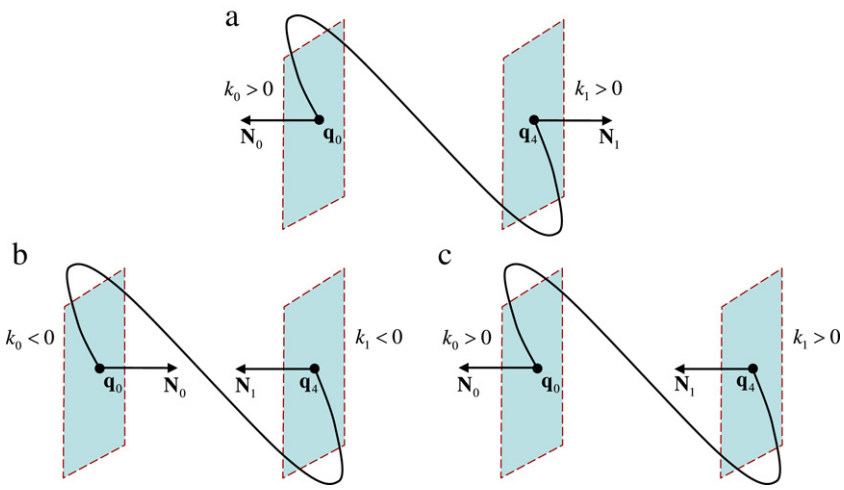


Fig. 8. The extraordinary situations.

the shape of boundary Bézier curve $\mathbf{q}(t)$. What is even worse is that many unsolvable situations now occur in the equality constrained least-squares fitting problem of Eq. (9). Furthermore, the G^1 continuity system can no longer be solved locally.

7.2. The degree of the scalar weight functions

The problem of choosing scalar weight functions has been extensively discussed in the literature, by authors such as Peters [20] and Hahmann and Bonneau [28]. The major issues are the type of functions (e.g. polynomial or rational) and their degrees (e.g. constant, linear or quadratic). The simplest situation is to assume that $\lambda(t)$ and $\mu(t)$ are both linear polynomials, just as Farin does in formulating his G^1 continuity conditions [44]. For the sake of simplicity, we also keep $\lambda(t)$ linear. However, as discussed in Section 6, the lowest possible degree of $\mu(t)$ appears to be 4. In fact, the values of μ_0 and μ_4 are determined by the boundary Bézier curves. In order to solve the vertex G^1 continuity system, two variables (μ_1 and μ_3) are necessary at least. Moreover, to eliminate the singularities from the edge G^1 continuity system requires one additional variable. Therefore, we make $\lambda(t)$ linear and $\mu(t)$ quartic, i.e $r = 1$ and $s = 4$.

7.3. The degree of the triangular Bézier patches

After we have chosen the degrees of boundary Bézier curves and scalar weight functions, we can determine the degree of triangular Bézier patches from the relation $r + n = s + m$. From the considerations already discussed, the best choice is $r = 1, s = 4$, $m = 4$ and $n = 7$. If one wished to decrease n by 1 while retaining enough freedom still to approximate S, it may be possible to set $r = 2, s = 4, m = 4$ and $n = 6$. But this makes the G^1 continuity constraint system global.

7.4. Time complexity

Assume the triangular base mesh B has v vertices, e edges and f faces. The cost of constructing the boundary curves network is dominated by the solution of the equality constrained least-squares fitting problem Eq. (9). Although this problem is both nonlinear and equality constrained, it can be solved efficiently by the local SQP algorithm with just a few iterations. And at each iteration only a linear system of 7 equations in 7 unknowns needs to be solved. Therefore, the time complexity for constructing boundary curves network is $\mathcal{O}(e)$.

The most computationally expensive step is to determine a G^0 triangular surface using Eq. (12) or Eq. (13). This step involves sampling the implicit surface, and solving a linear system of 15 equations in 15 unknowns. Its time complexity is $\mathcal{O}(f)$. As described in Section 6.3, we need to solve three equality constrained quadratic programs, two for the vertex G^1 continuity system (i.e. the tangent and normal component subproblems) and one for the edge G^1 continuity system. They can be solved with linear complexity $\mathcal{O}(v) + \mathcal{O}(e)$. The additional but optional computation of the unconstrained inner control points has a complexity of $\mathcal{O}(f)$.

In summary, the overall time complexity of our algorithm is proportional to $\mathcal{O}(v + e + f)$. During the whole process, we only need to solve local and linear systems. So, our algorithm is straightforward to implement and hence computationally effective.

8. Results and discussion

We have implemented all the algorithms described in previous sections. Using the triangular base meshes produced by the Bloothmenthal's polygonizer and the quadric error metric (QEM) algorithm, a range of implicit surfaces with increasing complexity have been tested on our implementation.

8.1. Performance

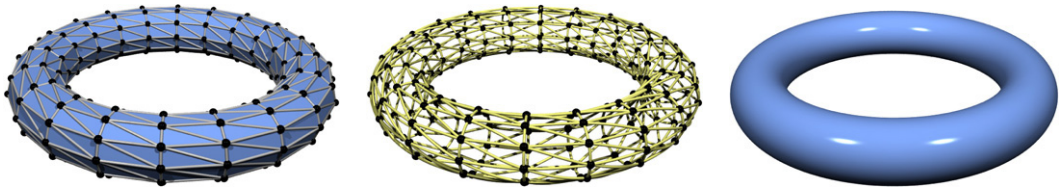
The following timings were taken on a Pentium IV 3.6 GHz machine with 1 GB of RAM:

Although nonlinear equality constrained optimization problems have to be solved during the process of constructing the boundary curves network, this stage doesn't take much computational time. In the sixth column of Table 3, we show the average number of iterations in the local SQP algorithm for each edge, with the stopping criterion $\varepsilon = 10^{-6}$. The results indicate that just 2 to 3 iterations are usually enough.

From this table we can observe that the total computation time is a near linear function of the number of faces in the triangular base mesh with a constant coefficient. And profiting from its local and linear properties, the peak memory usage of our algorithm is low and increases slowly. The presented method achieves high performance.

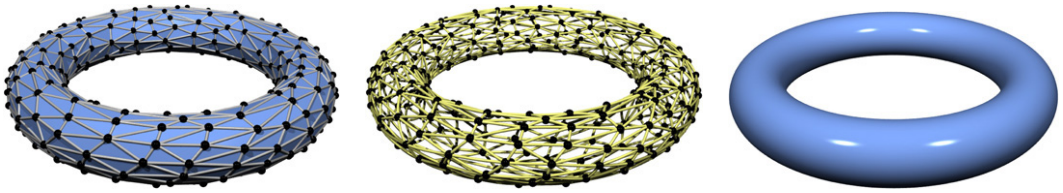
Table 3
Performance on a variety of implicit surfaces. All running times are measured in seconds.

Model	Polygonization	Decimation	Number of faces	Boundary network	Average iterations	G^0 surface	Vertex G^1 system	Edge G^1 system	Peak memory	Total time
Fig. 9	–	–	400	0.22	2.40	1.33	0.05	0.11	25,320 K	1.71
Fig. 10	0.11	0.52	600	0.33	2.47	1.97	0.06	0.16	26,760 K	3.15
Fig. 11	0.14	0.83	1200	0.56	2.32	3.85	0.14	0.33	30,092 K	5.85
Fig. 12	0.20	1.06	1800	0.83	2.23	5.70	0.24	0.47	33,152 K	8.50
Fig. 13	0.27	1.65	2000	0.92	2.20	6.35	0.25	0.53	34,968 K	9.97
Fig. 14	0.55	2.91	4000	3.17	2.32	14.80	0.50	1.06	37,080 K	22.99
Fig. 15	0.71	3.52	5000	7.70	2.40	23.99	0.61	1.36	39,804 K	37.89



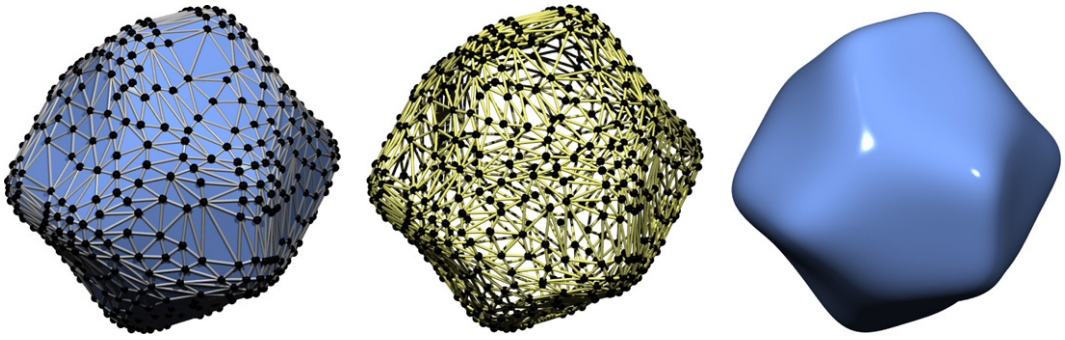
(a) Triangular base mesh. (b) Boundary curves network. (c) G^1 triangular spline surface.

Fig. 9. Approximation of a torus on a regular base mesh: $f(x, y, z) = (\sqrt{x^2 + y^2} - 5)^2 + z^2 - 1$.



(a) Triangular base mesh. (b) Boundary curves network. (c) G^1 triangular spline surface.

Fig. 10. Approximation of a torus on an irregular base mesh: $f(x, y, z) = \left(\sqrt{x^2 + y^2} - 5\right)^2 + z^2 - 1$.



(a) Triangular base mesh. (b) Boundary curves network. (c) G^1 triangular spline surface.

Fig. 11. Approximation of the dodecahedron surface: $f(x, y, z) = x^6 + y^6 + z^6 + 20(x^4y^2 + y^4z^2 + z^4x^2) - 1$.

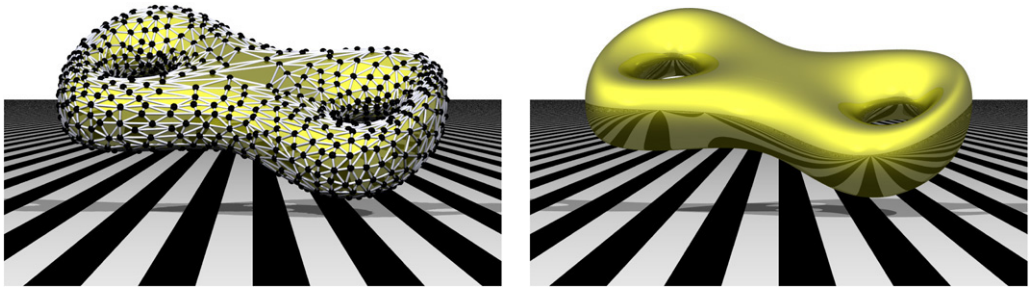
8.2. Discussion

To illustrate the influence of triangular base mesh on the final G^1 surface, we make two approximations of a torus. In Fig. 9 a regular triangular base mesh is constructed by sampling the parametric domain of the torus. While in Fig. 10, an irregular triangular base mesh is automatically generated by the method presented in Section 4. Although the two output G^1 surfaces have almost the same appearance, the one based on the regular mesh consists of fewer patches than the other. The construction of an optimal triangular base mesh with a fixed number of faces is a topic that remains to be investigated.

In order to verify the robustness of our algorithm for constructing boundary curves networks, various types of implicit surfaces were tested. For example, Fig. 12 for convex case, Fig. 13 for concave case and Fig. 11 for mixed case. In addition, a variety of implicit surfaces with different genus are tested on our algorithm. Figs. 12, 14 and 15 show surfaces of genus 2, 3 and 13 respectively. The results give confidence in the ability of our method to approximate general implicit surfaces.

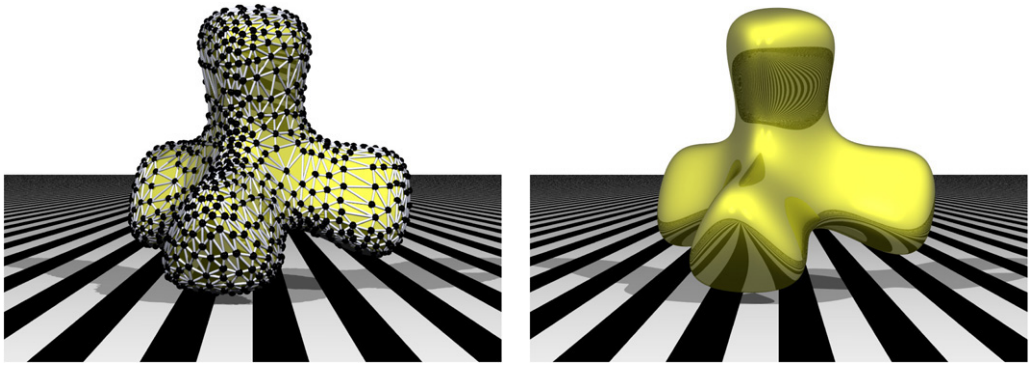
Figs. 12–15 were produced by rendering the output G^1 triangular spline surface as a realistic scene, where the surface's material properties were allocated as golden metal. The resulting reflections of black and white interlaced strips confirm the smoothness and visual quality of the final G^1 triangular spline surfaces.

In our algorithm, there are several free parameters. Most of them can be set as default values or chosen by programs automatically, except the number of faces f in the triangular base mesh. In practice, we supply this value by estimating the size and complexity of the approximated implicit surface. The necessity for this choice seems to be the sole obstacle to our algorithm leaving fully automatic.



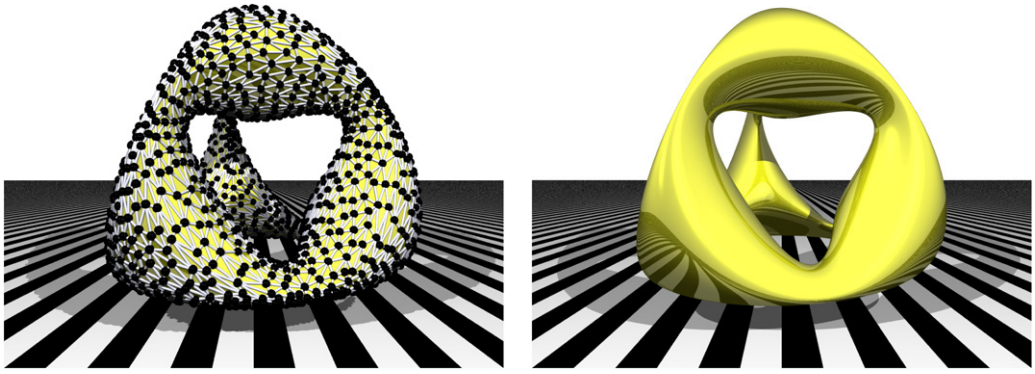
(a) Boundary curves network. (b) G^1 triangular spline surface.

Fig. 12. Approximation of the double torus: $f(x, y, z) = x^8 - 2x^6 + x^4 + 2x^4y^2 - 2x^2y^2 + y^4 + z^2 - 0.04$.



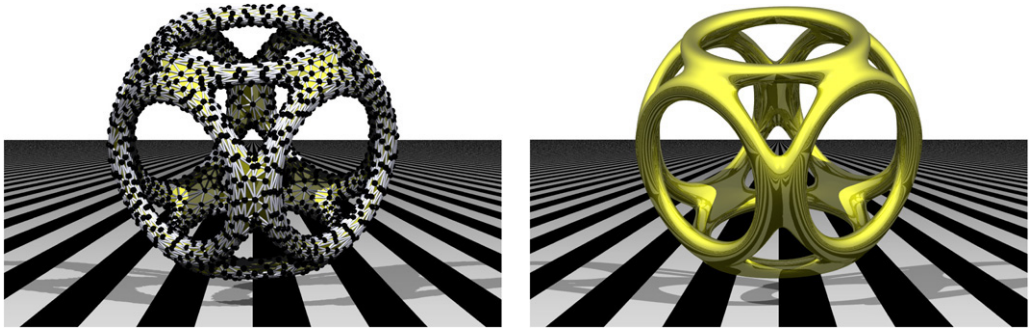
(a) Boundary curves network. (b) G^1 triangular spline surface.

Fig. 13. Approximation of the McMullen surface: $f(x, y, z) = (x^2 + 1)(y^2 + 1)(z^2 + 1) + 8xyz - 2$.



(a) Boundary curves network. (b) G^1 triangular spline surface.

Fig. 14. Approximation of the chair surface: $f(x, y, z) = (x^2 + y^2 + z^2 - ak^2)^2 - b[(z - k)^2 - 2x^2][(z + k)^2 - 2y^2]$, with $a = 0.95$, $b = 0.8$, $k = 5$.



(a) Boundary curves network. (b) G^1 triangular spline surface.

Fig. 15. Approximation of the decocube surface: $f(x, y, z) = [(x^2+y^2-a^2)^2+(z^2-1)^2][(y^2+z^2-a^2)^2+(x^2-1)^2][(z^2+x^2-a^2)^2+(y^2-1)^2]-b$, with $a = 0.8$, $b = 0.02$.

9. Summary and future work

In this paper, we have presented a method for the approximation of implicit surfaces by G^1 triangular spline surfaces. We have investigated a geometric Hermite interpolation scheme using normal curvature, and used it to develop a method for constructing boundary curves networks. By carefully analyzing degrees of freedom and consistency, we have devised an algorithm for obtaining a G^1 triangular spline surface of arbitrary topology, which is both local and free of singularities. Our method can achieve high precision at a low computational cost, and is straightforward to implement because it only involves local and linear solvers. The effectiveness, high precision and visual quality have been evidenced by numerical examples.

In the future we wish to improve the generation of the triangular base mesh, probably using error-driven adaptive subdivision strategies. It may be possible to determine the number of faces in the triangular base mesh automatically. There is also room for improvement in the criteria used in determining the twist and off-boundary control points in Eqs. (18), (19), (24) and (25). More precise metrics for measuring geometric errors may be accommodated, and a theoretic analysis on the approximation power will be the subject of further investigation. Because the presented G^1 spline surface interpolation scheme can work with the second order data defined on a triangular base mesh, we expect that they will find more applications due to their desirable properties.

Acknowledgments

This work was supported by grant No. R01-2005-000-11257-0 from the Basic Research Program of the Korea Science and Engineering Foundation, by grant No. KRF-2006-070-C00014 from the Korea Research Foundation Grant funded by the Korean Government and in part by the Seoul R&BD Program. The work of Wei-hua Tong was partially supported by the National Natural Science Foundation of China under grant No. 10726007. The authors would like to thank Ahmad Nasri for his continuous encouragement and early discussions on G^1 continuity issues for points cloud, and the anonymous referees for their valuable comments and suggestions.

Appendix A. The formula for normal curvature

Using the notation of Section 5.2, we have

$$k_0 = \frac{3\|\mathbf{t}_0 \times (\mathbf{q}_2 - \mathbf{q}_0)\|}{4\lambda^2} \times \left(\frac{[\mathbf{t}_0 \times (\mathbf{q}_2 - \mathbf{q}_0)] \times \mathbf{t}_0}{\|[\mathbf{t}_0 \times (\mathbf{q}_2 - \mathbf{q}_0)] \times \mathbf{t}_0\|} \cdot \mathbf{N}_0 \right).$$

From the formulas $(\mathbf{a} \times \mathbf{b}) \cdot (\mathbf{c} \times \mathbf{d}) = (\mathbf{a} \cdot \mathbf{c})(\mathbf{b} \cdot \mathbf{d}) - (\mathbf{a} \cdot \mathbf{d})(\mathbf{b} \cdot \mathbf{c})$ and $\|\mathbf{t}_0\|^2 = 1$, we obtain

$$\begin{aligned} \|\mathbf{t}_0 \times (\mathbf{q}_2 - \mathbf{q}_0)\|^2 &= [\mathbf{t}_0 \times (\mathbf{q}_2 - \mathbf{q}_0)] \cdot [\mathbf{t}_0 \times (\mathbf{q}_2 - \mathbf{q}_0)] \\ &= \|\mathbf{q}_2 - \mathbf{q}_0\|^2 - [\mathbf{t}_0 \cdot (\mathbf{q}_2 - \mathbf{q}_0)]^2. \end{aligned}$$

And from the formulas $(\mathbf{a} \times \mathbf{b}) \times \mathbf{c} = (\mathbf{a} \cdot \mathbf{c})\mathbf{b} - (\mathbf{a} \cdot \mathbf{b})\mathbf{c}$ and $\|\mathbf{t}_0\|^2 = 1$, we get

$$\begin{aligned} [\mathbf{t}_0 \times (\mathbf{q}_2 - \mathbf{q}_0)] \times \mathbf{t}_0 &= (\mathbf{t}_0 \cdot \mathbf{t}_0)(\mathbf{q}_2 - \mathbf{q}_0) - [\mathbf{t}_0 \cdot (\mathbf{q}_2 - \mathbf{q}_0)]\mathbf{t}_0 \\ &= (\mathbf{q}_2 - \mathbf{q}_0) - [\mathbf{t}_0 \cdot (\mathbf{q}_2 - \mathbf{q}_0)]\mathbf{t}_0. \end{aligned}$$

Therefore, we see that

$$\begin{aligned} \|[\mathbf{t}_0 \times (\mathbf{q}_2 - \mathbf{q}_0)] \times \mathbf{t}_0\|^2 &= \|(\mathbf{q}_2 - \mathbf{q}_0) - [\mathbf{t}_0 \cdot (\mathbf{q}_2 - \mathbf{q}_0)]\mathbf{t}_0\|^2 \\ &= \big((\mathbf{q}_2 - \mathbf{q}_0) - [\mathbf{t}_0 \cdot (\mathbf{q}_2 - \mathbf{q}_0)]\mathbf{t}_0\big) \\ &\quad \cdot \big((\mathbf{q}_2 - \mathbf{q}_0) - [\mathbf{t}_0 \cdot (\mathbf{q}_2 - \mathbf{q}_0)]\mathbf{t}_0\big) \\ &= \|\mathbf{q}_2 - \mathbf{q}_0\|^2 - [\mathbf{t}_0 \cdot (\mathbf{q}_2 - \mathbf{q}_0)]^2. \end{aligned}$$

Since $\mathbf{t}_0 \cdot \mathbf{N}_0 = 0$, we can write

$$\big([\mathbf{t}_0 \times (\mathbf{q}_2 - \mathbf{q}_0)] \times \mathbf{t}_0\big) \cdot \mathbf{N}_0 = (\mathbf{q}_2 - \mathbf{q}_0) \cdot \mathbf{N}_0.$$

Combining all these formulas, we finally arrive at Eq. (8).

Appendix B. The tangent plane constraint

Using the notation of Section 6.3, we prove that the coefficients $\{\lambda_i\}_{i=1}^n$ satisfy

$$\prod_{i=1}^{n} \bar{\lambda}_0^i = \prod_{i=1}^{n} \lambda_0^i.$$

Proof. The geometric meaning of Eq. (15) is that all the control points $\mathbf{o}$, $\{\mathbf{q}_1^i\}_{i=1}^n$ and $\{\mathbf{c}_1^i\}_{i=1}^n$ are located on a common plane, which is the tangent plane. If we subtract $\mathbf{o}$ from both sides of each equation in Eq. (15), we obtain

$$\bar{\lambda}_0^i(\mathbf{c}_1^{i-1} - \mathbf{o}) + \lambda_0^i(\mathbf{c}_1^{i+1} - \mathbf{o}) = \mu_0^i(\mathbf{q}_1^i - \mathbf{o}), \quad i = 1, 2, \ldots, n. \tag{B.1}$$

This means that the vectors $(\mathbf{c}_1^{i-1} - \mathbf{o})$, $(\mathbf{c}_1^{i+1} - \mathbf{o})$ and $(\mathbf{q}_1^i - \mathbf{o})$ are coplanar. Since $(\mathbf{c}_1^i - \mathbf{o}) \parallel (\mathbf{q}_1^i - \mathbf{o})$, we take the cross product of both side of Eq. (B.1) with $(\mathbf{c}_1^i - \mathbf{o})$ and then their inner product with $\mathbf{N}$, which yields

$$\begin{aligned} &\bar{\lambda}_0^i\big((\mathbf{c}_1^{i-1} - \mathbf{o}) \times (\mathbf{c}_1^i - \mathbf{o})\big) \cdot \mathbf{N} \\ &\quad + \lambda_0^i\big((\mathbf{c}_1^{i+1} - \mathbf{o}) \times (\mathbf{c}_1^i - \mathbf{o})\big) \cdot \mathbf{N} = 0, \quad i = 1, 2, \ldots, n, \end{aligned}$$

where $\mathbf{N}$ is the normal to the plane on which $\mathbf{o}$, $\{\mathbf{q}_1^i\}_{i=1}^n$ and $\{\mathbf{c}_1^i\}_{i=1}^n$ are located. We can rearrange this equation as

$$\frac{\bar{\lambda}_0^i}{\lambda_0^i} = -\frac{\big((\mathbf{c}_1^{i+1} - \mathbf{o}) \times (\mathbf{c}_1^i - \mathbf{o})\big) \cdot \mathbf{N}}{\big((\mathbf{c}_1^{i-1} - \mathbf{o}) \times (\mathbf{c}_1^i - \mathbf{o})\big) \cdot \mathbf{N}}, \quad i = 1, 2, \ldots, n.$$

Hence we have

$$\begin{aligned} \prod_{i=1}^{n} \frac{\bar{\lambda}_0^i}{\lambda_0^i} &= (-1)^n \frac{\big((\mathbf{c}_1^2 - \mathbf{o}) \times (\mathbf{c}_1^1 - \mathbf{o})\big) \cdot \mathbf{N}}{\big((\mathbf{c}_1^n - \mathbf{o}) \times (\mathbf{c}_1^1 - \mathbf{o})\big) \cdot \mathbf{N}} \\ &\quad \times \frac{\big((\mathbf{c}_1^3 - \mathbf{o}) \times (\mathbf{c}_1^2 - \mathbf{o})\big) \cdot \mathbf{N}}{\big((\mathbf{c}_1^1 - \mathbf{o}) \times (\mathbf{c}_1^2 - \mathbf{o})\big) \cdot \mathbf{N}} \\ &\quad \times \cdots \times \frac{\big((\mathbf{c}_1^1 - \mathbf{o}) \times (\mathbf{c}_1^n - \mathbf{o})\big) \cdot \mathbf{N}}{\big((\mathbf{c}_1^{n-1} - \mathbf{o}) \times (\mathbf{c}_1^n - \mathbf{o})\big) \cdot \mathbf{N}} \\ &= 1, \end{aligned}$$

which ends our proof. ■

Appendix C. The consistency of the vertex G^1 continuity system

Using the notation of Section 6.3, we will prove that: if the boundary Bézier curves $\{\mathbf{q}^i(t)\}_{i=1}^n$ are constructed as specified in Section 5, then the vertex G^1 continuity system is consistent, so that Eqs. (14) and (17) possess at least one solution.

Proof. As discussed in Section 6.3, the above statement is obviously correct when n is odd. Hence, we need only prove the case when n is even, which we now assume. Since Eq. (14) is equivalent to Eq. (17), it is sufficient to prove either of them. Here, we choose to prove that Eq. (17) is consistent. Eq. (17) can be decomposed into two subproblems, i.e. the tangent component subproblem and the normal component subproblem. If we can prove that each of these two subproblems have one solution, then the proof will be finished.

Starting with the tangent component subproblem, we assert that the rank $\left(\begin{bmatrix} \mathbf{A} & \mathbf{0} & \mathbf{Q}_1 \\ \mathbf{0} & \mathbf{A} & \mathbf{Q}_2 \end{bmatrix}\right)$ equals to $2n$. Because $2n \geqslant$ rank

$\left(\begin{bmatrix}\mathbf{A} & \mathbf{0} & \mathbf{Q}_1\\ \mathbf{0} & \mathbf{A} & \mathbf{Q}_2\end{bmatrix}\right) \geqslant \operatorname{rank}\left(\begin{bmatrix}\mathbf{A} & \mathbf{0}\\ \mathbf{0} & \mathbf{A}\end{bmatrix}\right) = 2n-2$, $\operatorname{rank}\left(\begin{bmatrix}\mathbf{A} & \mathbf{0} & \mathbf{Q}_1\\ \mathbf{0} & \mathbf{A} & \mathbf{Q}_2\end{bmatrix}\right)$ can only equal to $2n$, $2n-1$, or $2n-2$. However, if $\operatorname{rank}\left(\begin{bmatrix}\mathbf{A} & \mathbf{0} & \mathbf{Q}_1\\ \mathbf{0} & \mathbf{A} & \mathbf{Q}_2\end{bmatrix}\right) < 2n$, we can deduce that one or both of $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are zero matrices. This contradicts with the fact that the vectors $\{(\mathbf{q}_1^i - \mathbf{o})\}_{i=1}^n$ span a tangent plane. Hence, we arrive at our assertion. In addition, we observe that the dimension of matrix $\begin{bmatrix}\mathbf{A} & \mathbf{0} & \mathbf{Q}_1\\ \mathbf{0} & \mathbf{A} & \mathbf{Q}_2\end{bmatrix}$ is $2n \times 3n$ and has full row rank. The theory of equality constrained quadratic programming [42], allows us to affirm that the tangent component subproblem has a unique solution.

For the normal component subproblem, we assert that $\operatorname{rank}(\mathbf{A}, \mathbf{R}_3) = \operatorname{rank}(\mathbf{A})$. In fact, if let

$$\mathbf{x} = \begin{bmatrix} 2(\mathbf{c}_1^1 - \mathbf{o})^{\mathrm{T}}\mathbf{M}(\mathbf{c}_1^n - \mathbf{o})/3 \\ 2(\mathbf{c}_1^2 - \mathbf{o})^{\mathrm{T}}\mathbf{M}(\mathbf{c}_1^1 - \mathbf{o})/3 \\ \vdots \\ 2(\mathbf{c}_1^i - \mathbf{o})^{\mathrm{T}}\mathbf{M}(\mathbf{c}_1^{i-1} - \mathbf{o})/3 \\ \vdots \\ 2(\mathbf{c}_1^n - \mathbf{o})^{\mathrm{T}}\mathbf{M}(\mathbf{c}_1^{n-1} - \mathbf{o})/3 \end{bmatrix}, \qquad \mathbf{M} = \begin{bmatrix} k_{\min} & & \\ & k_{\max} & \\ & & \mathbf{0} \end{bmatrix},$$

then we can show that

$$\begin{aligned}[\mathbf{Ax}]_i &= 4\bar{\lambda}_0^i(\mathbf{c}_1^i - \mathbf{o})^{\mathrm{T}}\mathbf{M}(\mathbf{c}_1^{i-1} - \mathbf{o}) + 4\lambda_0^i(\mathbf{c}_1^{i+1} - \mathbf{o})^{\mathrm{T}}\mathbf{M}(\mathbf{c}_1^i - \mathbf{o}) \\ &= 4(\mathbf{c}_1^i - \mathbf{o})^{\mathrm{T}}\mathbf{M}[\bar{\lambda}_0^i(\mathbf{c}_1^{i-1} - \mathbf{o}) + \lambda_0^i(\mathbf{c}_1^{i+1} - \mathbf{o})] \\ &= 4\mu_0^i(\mathbf{c}_1^i - \mathbf{o})^{\mathrm{T}}\mathbf{M}(\mathbf{c}_1^i - \mathbf{o}) \\ &= 3\mu_0^i(\mathbf{q}_2^i - \mathbf{o}) \cdot \mathbf{N} \\ &= [\mathbf{R}_3]_i, \quad i = 1, 2, \ldots, n.\end{aligned}$$

Again, the theory of equality constrained quadratic programming allows us to affirm that the normal component subproblem has a unique solution. ■

Appendix D. The consistency of the edge G^1 continuity system

In this section, we analysis the consistency of the edge G^1 continuity system. Using the notation of Section 6.4, we prove that:

$$\operatorname{rank}(\mathbf{A}) = \begin{cases} 4 & \text{if } \lambda_0 \neq \lambda_1, \\ 3 & \text{otherwise.} \end{cases}$$

Proof. If $\lambda_0 \neq \lambda_1$, there is at least one 4×4 nonsingular submatrix which lies on the intersection of rows 1, 2, 3 and 4 with columns 1, 3, 5 and 6, as follows:

$$\det \mathbf{A}\begin{pmatrix}1 & 2 & 3 & 4\\ 1 & 3 & 5 & 6\end{pmatrix} = \begin{vmatrix} 15\bar{\lambda}_0 & 0 & 0 & 0 \\ 15\bar{\lambda}_1 & 20\bar{\lambda}_0 & 0 & 0 \\ 0 & 20\bar{\lambda}_1 & 15\bar{\lambda}_0 & 15\lambda_0 \\ 0 & 0 & 15\bar{\lambda}_1 & 15\lambda_1 \end{vmatrix}$$

$$= \begin{vmatrix} 15\bar{\lambda}_0 & 0 & 0 & 0 \\ 15\bar{\lambda}_1 & 20\bar{\lambda}_0 & 0 & 0 \\ 0 & 20\bar{\lambda}_1 & 15(1-\lambda_0/\lambda_1) & 0 \\ 0 & 0 & 15\bar{\lambda}_1 & 15\lambda_1 \end{vmatrix} \neq 0.$$

This shows that $\operatorname{rank}(\mathbf{A}) = 4$.

If $\lambda_0 = \lambda_1$ and denote the ith row vector of $\mathbf{A}$ as $\mathbf{A}_i$, we observe that

$$\mathbf{A}_1 - \mathbf{A}_2 + \mathbf{A}_3 - \mathbf{A}_4 = \mathbf{0} \Longrightarrow \det(\mathbf{A}) = 0 \quad \text{and} \quad \operatorname{rank}(\mathbf{A}) < 4.$$

Moreover, there is at least one 3×3 nonsingular submatrix which lies on the intersection of rows 1, 2 and 3 with columns 1, 3 and 5, as follows:

$$\det \mathbf{A}\begin{pmatrix}1 & 2 & 3\\ 1 & 3 & 5\end{pmatrix} = \begin{vmatrix} 15\bar{\lambda}_0 & 0 & 0 \\ 15\bar{\lambda}_0 & 20\bar{\lambda}_0 & 0 \\ 0 & 20\bar{\lambda}_0 & 15\bar{\lambda}_0 \end{vmatrix} \neq 0.$$

Therefore we can assert that $\operatorname{rank}(\mathbf{A}) = 3$. In order to make the linear system of Eq. (20) consistent, the $\{\mathbf{rhs}_i\}_{i=1}^4$ must satisfy Eq. (22), which can be accomplished by choosing the μ_2 as Eq. (23). ■

References

[1] DeRose T, Mann S. An approximately G^1 cubic surface interpolant. In: Lyche T, Schumaker LL, editors. Mathematical methods in computer aided geometric design II. San Diego: Academic Press; 1992. p. 185–96.
[2] Mann S. Surface approximation using geometric hermite patches. Ph.D. thesis. Seattle (WA): University of Washington; 1992.
[3] de Boor C, Höllig K, Sabin MA. High accuracy geometric Hermite interpolation. Computer Aided Geometric Design 1987;4(4):269–78.
[4] Goodman TNT, Unsworth K. Shape preserving interpolation by curvature continuous parametric curves. Computer Aided Geometric Design 1988;5(4):323–40.
[5] Goodman TNT. Shape preserving interpolation by parametric rational cubic splines. In: Agarwal RP, Chow YM, Wilson SJ, editors. Numerical mathematics Singapore 1988. International series of numerical mathematics, 1988. p. 149–58.
[6] Goodman TNT, Ong BH, Unsworth K. Constrained interpolation using rational cubic splines. In: NURBS for curve and surface design. Philadelphia: SIAM; 1991. p. 59–74.
[7] Meek DS, Ong BH, Walton DJ. Constrained interpolation with rational cubics. Computer Aided Geometric Design 2003;20(5):253–75.
[8] Meek DS, Walton DJ. Planar G^2 Hermite interpolation with some fair, C-shaped curves. Journal of Computational and Applied Mathematics 2002;139(1):141–61.
[9] Walton DJ, Meek DS. G^2 curve design with a pair of Pythagorean Hodograph quintic spiral segments. Computer Aided Geometric Design 2007;24(5):267–85.
[10] Höllig K. Algorithms for rational spline curves. In: Transactions of the army conference on applied mathematics and computing. 1987. p. 287–300.
[11] Höllig K, Koch J. Geometric Hermite interpolation. Computer Aided Geometric Design 1995;12(6):567–80.
[12] Höllig K, Koch J. Geometric Hermite interpolation with maximal order and smoothness. Computer Aided Geometric Design 1996;13(8):681–95.
[13] Peters J. Local generalized Hermite interpolation by quartic C^2 space curves. ACM Transactions on Graphics 1989;8(3):235–42.
[14] Schaback R. Optimal geometric Hermite interpolation of curves. In: Proceedings of the international conference on mathematical methods for curves and surfaces II. 1998. P. 417–28.
[15] Degen WLF. Geometric Hermite interpolation: In memoriam Josef Hoschek. Computer Aided Geometric Design 2005;22(7):573–92.
[16] Peters J. Geometric continuity. In: Farin G, Hoschek J, Kim M-S, editors. Handbook of computer aided geometric design. Amsterdam: Elsevier; 2002. p. 193–227.
[17] Farin G. Curves and surfaces for CAGD: A practical guide. 5th ed. San Francisco: Morgan Kaufmann; 2002.
[18] Warren J, Weimer H. Subdivision methods for geometric design: A constructive approach. San Francisco: Morgan Kaufmann; 2001.
[19] Peters J, Reif U. Subdivision surfaces. Berlin: Springer-Verlag; 2008.
[20] Peters J. Smooth interpolation of a mesh of curves. Constructive Approximation 1991;7(1):221–46.
[21] Mann S, Loop C, Lounsbery M, Meyers D, Painter J, DeRose T, Sloan K. A survey of parametric scattered data fitting using triangular interpolants. In: Hagen H, editor. Curve and surface design. Philadelphia: SIAM; 1992. p. 145–72.
[22] Farin G. A modified Clough–Tocher interpolant. Computer Aided Geometric Design 1985;2(1-3):19–27.
[23] Shirman LA, Séquin CH. Local surface interpolation with Bézier patches. Computer Aided Geometric Design 1987;4(4):279–95.
[24] Nielson G. A transfinite, visually continuous, triangular interpolant. In: Farin G, editor. Geometric modeling: Algorithms and new trends. Philadelphia: SIAM; 1987. p. 235–46.
[25] Hagen H, Pottmann H. Curvature continuous triangular interpolants. In: Mathematical methods in computer aided geometric design. San Diego: Academic Press; 1989. p. 373–84.
[26] Neamtu M, Pfluger PR. Degenerate polynomial patches of degree 4 and 5 used for geometrically smooth interpolation in $\mathbb{R}^3$. Computer Aided Geometric Design 1994;11(4):451–74.
[27] Loop C. A G^1 triangular spline surface of arbitrary topological type. Computer Aided Geometric Design 1994;11(3):303–30.
[28] Hahmann S, Bonneau G-P. Triangular G^1 interpolation by 4-splitting domain triangles. Computer Aided Geometric Design 2000;17(8):731–57.
[29] Yvart A, Hahmann S, Bonneau G-P. Hierarchical triangular splines. ACM Transactions on Graphics 2005;24(4):1374–91.
[30] Liu Y, Mann S. Paramteric triangular Bézier surface interpolation with approximate continuity. In: SMP'08: Proceedings of the 2008 ACM symposium on Solid and physical modeling. 2008. p. 381–7.
[31] Cho D-Y, Lee K-Y, Kim T-W. Interpolating G^1 Bézier surfaces over irregular curve networks for ship hull design. Computer-Aided Design 2006;38(6):641–60.
[32] Cho D-Y, Lee K-Y, Kim T-W. Analysis and avoidance of singularities for local G^1 surface interpolation of Bézier curve network with 4-valent nodes. Computing 2007;79(2–4):261–79.
[33] Bloomenthal J, Bajaj C, Blinn J, Cani-Gascuel M-P, Rockwood A, Wyvill B, Wyvill G. Introduction to implicit surfaces. San Francisco: Morgan Kaufmann; 1997.

[34] Velho L, Gomes J, Figueiredo LH. Implicit objects in computer graphics. New York: Springer-Verlag; 2002.
[35] de Araújo BR, Jorge JAP. Adaptive polygonization of implicit surfaces. Computers & Graphics 2005;29(5):686–96.
[36] Bloomenthal J. An implicit surface polygonizer. In: Graphics gems IV. San Diego: Academic Press; 1994. p. 324–49.
[37] Garland M. Multiresolution modeling: Survey & future opportunities. In: Proceedings of Eurographics 1999 – State of the Art Reports. 1999. p. 111–31.
[38] Luebke D, Reddy M, Cohen JD, Varshney A, Watson B, Huebner R. Level of detail for 3D graphics. San Francisco: Morgan Kaufmann; 2003.
[39] Garland M, Heckbert PS. Surface simplification using quadric error metrics. In: Proceedings of ACM SIGGRAPH 1997. 1997. p. 209–16.
[40] Garland M, Zhou Y. Quadric-based simplification in any dimension. ACM Transactions on Graphics 2005;24(2):209–39.
[41] Golub GH, Loan CFV. Matrix computations. 3rd ed. Baltimore: The Johns Hopkins University Press; 1996.
[42] Nocedal J, Wright S. Numerical optimization. New York: Springer; 1999.
[43] Du W-H, Schmitt FJM. On the G^1 continuity of piecewise Bézier surfaces: A review with new results. Computer-Aided Design 1990;22(9):556–73.
[44] Farin G. A construction for visual C^1 continuity of polynomial surface patches. Computer Graphics and Image Processing 1982;20(3):272–82.