Automatic curvilinear mesh generation with smooth boundary driven by guaranteed validity and fidelity

Jing Xu and Andrey N. Chernikov

Department of Computer Science, Old Dominion University, Norfolk, VA, USA,
{jxu, achernik}@cs.odu.edu

Summary. The development of robust high-order finite element methods requires the construction of valid high-order meshes for complex geometries without user intervention. This paper presents a novel approach for automatically generating a high-order mesh with two main features: first, the boundary of the mesh is globally smooth; second, the mesh boundary satisfies a required fidelity tolerance. Invalid elements are guaranteed to be eliminated. Example meshes demonstrate the features of the algorithm.

Key words: mesh generation, high-order, Bézier polynomial basis, smoothness

1 Introduction

High-order finite element methods have been used extensively in direct numerical simulations in the last decades. The exponential rates of convergence, small dispersion and diffusion solution errors have motivated the development of the technics of the appropriate geometric representation of high-order finite elements [1, 2, 3]. How well the geometry is approximated has fundamentally important effects on the accuracy of finite element solutions [4, 5]. Therefore, valid meshes with properly curved elements must be constructed to approximate the curved domain geometry.

The discretization error results from the fact that a function of a continuous variable is represented in the computer by a finite number of evaluations. In conventional meshes with all straight-sided elements, the discretization error is usually controlled by making sufficiently small elements where geometry features occur such as on the objects' boundary. But this is not numerically efficient in the sense that the cost of assembling and solving a sparse system of linear equations in the FE method directly depends on the number of elements. The high-order methods however, decompose the solution domain into fewer elemental regions which capture the features of the geometry.

There are two ways to accomplish the generation of a curvilinear mesh when a geometric domain is given. The first is to directly create a valid curvilinear boundary and interior discretization with required size and shape of elements. The second way is to initially construct a straight-edge discretization of the model geometry, followed by the transformation of that discretization into high-order elements suitable for a high-order FE method.

Various procedures have been developed and implemented using the latter approach. Sherwin and Peiro [6] addressed an high-order unstructured mesh generation algorithm. In this paper, a linear triangular surface mesh is first generated, the transformation of that mesh into high-order surface is performed, and finally a curved mesh is constructed of the interior volume. Three strategies are adopted to alleviate the problem of invalid high-order meshes: optimization of the surface mesh that accounts for surface curvature, hybrid meshing with prismatic elements near the domain boundaries and curvature driven surface mesh adaption.

Dey et al. [14] described an iterative algorithm for curving straight-edge meshes using quadratic Lagrange interpolation functions. First, curve all mesh edges and faces classified on curved model boundaries. Second, detect and eliminate intersections between mesh edges on the model surface. Third, use local mesh modification tools to aid in correcting invalid curved mesh regions.

Shephard et al. [7] discussed the automatic generation of adaptively controlled meshes for general three-dimensional domains. The algorithm starts with isolating all of the edges and vertices in the model that will have singularities, construction of a coarse linear mesh on the boundary of the model with appropriate geometric gradation towards the isolated singular features and construction of a coarse linear mesh of the remainder of the domain. Then the algorithm curves the singular feature isolation mesh, and the remaining mesh entities classified on the curved boundaries. Finally, mesh modification is applied to ensure a valid mesh of acceptably shaped elements.

Luo et al. [8] isolates singular reentrant model entities, then generates linear elements around those features, and curves them while maintaining the gradation. Linear elements are generated for the rest of the domain, and those elements which are classified on the curved boundary, are transformed into curved elements conforming to the curved boundary. Modification operations are applied to eliminate invalid elements whenever they are introduced. Later, they extended their work to adapted boundary layer meshes to allow for higher-order analysis of viscous flows [22]. The layered structure of anisotropic elements in the boundary layer meshes is able to construct elements with proper configuration and gradation.

George et al. [10] proposed a method for constructing tetrahedral meshes of degree two from a polynomial surface mesh of degree 2. Corresponding linear surface mesh is first extracted, followed by constructing the linear volumetric mesh. Next the algorithm enriches the linear mesh to polynomial with degree 2 mesh by introducing the edge nodes. After that *Jacobian* is introduced for

guiding the correction of the invalid curved elements. Finally an optimization procedure is used to enhance the quality of the curved mesh.

Lu et al. [12] presented a parallel mesh adaptation method with curved element geometry. The core of the algorithm is two classes of mesh modification. Curved entity reshape operation and local mesh modification operation explicitly resolve element invalidity and improve the shape quality.

The validity of a curved mesh is crucial to the successful execution of high-order finite element simulations. To verify the validity, it is efficient to calculate the determinant of the Jacobian matrix (Jacobian). A curved element is valid if and only if its Jacobian is strictly positive everywhere. However, it is cumbersome to verify the element validity when Lagrangian polynomial being used because calculating Jacobian becomes computationally and geometrically complex. Prior work shows that the properties of Bézier polynomials provide an attractive solution [9, 10, 11, 12]. A lower bound for the determinant of the Jacobian matrix can be evaluated by the convex hull property of the Bézier polynomial. If the lower bound is not tight enough, either degree elevation procedure or subdivision procedure is selected to yield a tighter lower bound [9, 10, 12]. Johnen [11] expands the Jacobian determinant using Bézier polynomial basis. Based on its properties, boundedness and positivity were obtained to provide an efficient way to determine the validity and to measure the distortion.

In this paper, a new approach is proposed for automatically generating a high-order mesh to represent geometry with smooth mesh boundaries and graded interior with guaranteed fidelity. Cubic Bézier polynomial basis is selected for the geometric representation of the elements because it provides a convenient framework supporting the smooth operation while maintaining guaranteed fidelity. We list the contributions in this paper here. To our knowledge, no consideration was given to them in prior work.

1. Curved mesh boundary is globally smooth, i.e., its tangent is everywhere continuous.
2. Curved mesh boundary everywhere satisfies a user-defined fidelity tolerance.

The procedure starts with the automatic construction of a graded linear mesh that simultaneously satisfies the quality and the fidelity requirements. The edges of those linear elements which are classified on curved boundary are then curved using cubic Bézier polynomial basis while maintaining the smoothness. To resist inverted elements, the procedure next curves the interior elements by solving for the equilibrium configuration of an elasticity problem. A validity verification procedure demonstrates that intersection edges and highly distorted elements are eliminated.

The rest of the paper is organized as follows. in Sect. 2, we review some basic definitions and material. Sect. 3 gives a description of the automatic construction of graded linear mesh, while Sect. 4 describes the transformation of those linear mesh elements into high-order elements. Sect. 5 presents the

validity checking algorithms. Sect. 6 proves mesh fidelity. We present meshing results in Sect. 7 and conclude in Sect. 8.

2 Preliminaries

The method uses cubic Bézier polynomial basis to construct a high-order mesh that has smooth boundaries. The idea is to deform the linear mesh edges such that the curved edges conform to the expected domain boundary. The determinant of the Jacobian matrix is used to determine the validity. In this section we review the Bézier curve, Bézier triangle and the Jacobian.

2.1 Bézier curve and Bézier triangle

We will express Bézier curves in terms of Bernstein polynomials. The $n+1$ Bernstein basis polynomials of degree n are defined explicitly by

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i} \quad i = 0, ..., n, t \in [0,1], \tag{1}$$

where the binomial coefficients are given by

$$\binom{n}{i} = \begin{cases} \frac{n!}{i!(n-i)!} & \text{if } 0 \leq i \leq n \\ 0 & \text{else} \end{cases} \tag{2}$$

Then a Bézier curve can be defined after the definition of Bernstein polynomials

$$b^n(t) = \sum_{i=0}^{n-r} b_i^r(t) B_i^{n-r}(t), \tag{3}$$

where $b_i^r(t)$ are the intermediate de Casteljau points which can be expressed in terms of Bernstein polynomials of degree r. They can be interpreted as control points of a Bézier curve of degree $n-r$

$$b_i^r(t) = \sum_{j=0}^{r} b_{i+j} B_j^r(t) \quad i \in 0, ..., n-r. \tag{4}$$

Specifically, a cubic Bézier curve can be written in the form of the barycentric coordinates for convenience,

$$b^3(u,v) = \sum_{i+j=3} B_{ij}^3(u,v) P_{ij}, \tag{5}$$

where $B_{ij}^3(u,v) = \frac{3!}{i!j!} u^i v^j$, $u \in [0,1]$ and $v \in [0,1]$ are the barycentric coordinates and $u+v=1$.

This lead to a simple definition of a Bézier triangle of degree three

$$\mathcal{T}^3(u,v,w) = \sum_{i+j+k=3} B^3_{ijk}(u,v,w)P_{ijk}, \tag{6}$$

where $B^3_{ijk}(u,v,w) = \frac{3!}{i!j!k!}u^i v^j w^k$, $u \in [0,1]$, $v \in [0,1]$and $w \in [0,1]$ be the barycentric coordinates and $u+v+w=1$.

2.2 The Jacobian

We explore the concept of a derivative of a coordinate transformation, which is known as the *Jacobian* of the transformation.

Consider, for instance, a mapping with *shape functions* (or *basis functions*) $N_a, a = 1, 2, ...$ from the set of local coordinates $\hat{x}$, $\hat{y}$ to a corresponding set of global coordinates x, y. By the *chain rule* of partial differentiation we have

$$\left\{ \frac{\partial N_a}{\partial \hat{x}} \;\; \frac{\partial N_a}{\partial \hat{y}} \right\} = \left\{ \frac{\partial N_a}{\partial x} \;\; \frac{\partial N_a}{\partial y} \right\} \begin{bmatrix} \frac{\partial x}{\partial \hat{x}} & \frac{\partial x}{\partial \hat{y}} \\ \frac{\partial y}{\partial \hat{x}} & \frac{\partial y}{\partial \hat{y}} \end{bmatrix} = \left\{ \frac{\partial N_a}{\partial x} \;\; \frac{\partial N_a}{\partial y} \right\} J, \tag{7}$$

$$J = \begin{bmatrix} \frac{\partial x}{\partial \hat{x}} & \frac{\partial x}{\partial \hat{y}} \\ \frac{\partial y}{\partial \hat{x}} & \frac{\partial y}{\partial \hat{y}} \end{bmatrix}. \tag{8}$$

J is known as the *Jacobian matrix* for the transformation. As x, y are explicitly given by the relation defining the curvilinear coordinates, the matrix J can be found explicitly in terms of the local coordinates.

3 Linear mesh construction

We adopt the image-to-mesh conversion algorithm [13], for four reasons: (1) it allows for a guaranteed angle bound (quality), (2) it allows for a guaranteed bound on the distance between the boundaries of the mesh and the boundaries of the tissues (fidelity), (3) it coarsens the mesh to a much lower number of elements with gradation in the interior, (4) it is formulated to work in both two and three dimensions. Once we have a high quality linear mesh, we are about to construct curvilinear mesh based on it as the next step.

4 Curvilinear mesh transformation from linear meshes

Although it is attractive to construct valid high-order meshes by curving mesh entities classified on curved boundaries and the remainder of the domain simultaneously, in practice we transform the linear mesh entities classified on boundaries followed by curving mesh entities in the interior while eliminating invalid elements. Bézier curve basis is selected because its mathematical descriptions are compact, intuitive, and elegant. It is easy to compute, easy to use in higher dimensions (3D and up), and can be stitched together to represent any shape.

4.1 Constructing smooth Bézier paths from boundary mesh entities

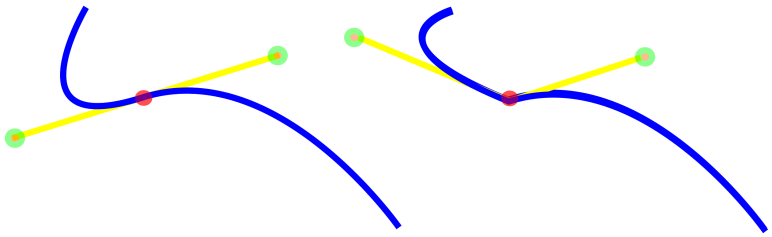

Fig. 1. Two Bézier paths with their control points (in green), made out of two quadratic Bézier curves connected by the *endpoints* (in red). The yellow line segments are tangents to the double sides of the red Bézier *endpoint*. Left: a smooth Bézier path because the two green control points and the red *endpoint* lie in a straight line. Right: a Bézier path with a cusp where the curves connect, because the two green control points and the red *endpoint* do not lie in a straight line.

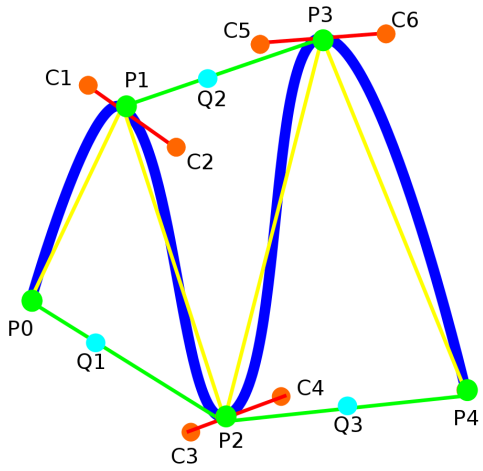

Fig. 2. An example of finding control points of a smooth cubic Bézier path. For the curve between P_1 and P_2, we need C_2 and C_3. On segment P_0P_2, find a point Q_1 such that$|P_0Q_1|/|Q_1P_2|= |P_0P_1|/|P_1P_2|$. Translate segment P_0P_2 so that point Q_1 lies on point P_1, and scale the length of translated segment P_0P_2, then the new position of point P_2 is the position of control point C_2. Similarly, the position of control point C_3 can be found by translating segment P_1P_3 so that point Q_2 lies on point P_2.

A curve or surface can be described as having G^n continuity, n being the measure of smoothness. Consider the segments on either side of a point on a curve:

G^0: The curves touch at the join point.
G^1: The curves also share a common tangent direction at the join point.

We aim to find a smooth G^1 curve passing through all the mesh boundary points given in order. A Bézier path is smooth provided that each endpoint and its two surrounding control points lie in a straight line. In other words, the two tangents at each Bézier endpoint are parallel. Fig. 1 shows two Bézier paths with their control points.

The basic idea is to calculate control points around each endpoint so that they lie in a straight line with the endpoint. However, curved segments would not flow smoothly together when quadratic Bézier form (three control points) is used. Instead, we need to go one order higher to a cubic Bézier (four control points) so we can build "S" shaped segments.

The points we have in hand are only endpoints of boundary segments, so the task becomes to find the other two control points to define the Bézier curve. We find these control points by translating the segments formed by the lines between the previous *endpoint* and the next *endpoint* such that these segments become the tangents of the curve at the *endpoints*. We scale these segments to control the curvature. An example is illustrated in Fig. 2.

4.2 Curving mesh entities in the interior

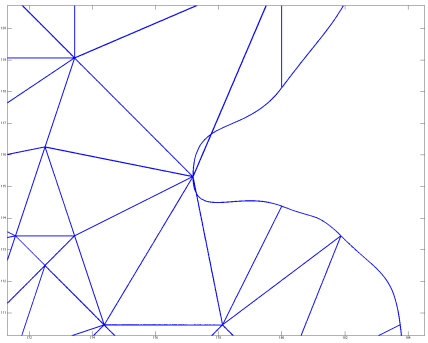

Fig. 3. Invalid elements.

It is usually not enough to curve only the boundary mesh edges because self-intersecting mesh edges may appear which lead to invalid elements. In such

cases, interior mesh elements should also be curved to eliminate the invalidity. Fig. 3 gives an example of such an occasion. Local mesh modifications such as minimizing the deformation, edge or facet deletion, splitting, collapsing, swapping as well as shape manipulation have been used to correct an invalid region [14, 8, 10, 12].

Persson [15] proposed a node relocation strategy for constructing well-shaped curved mesh. They use a nonlinear elasticity analogy, where the geometry of the domain to be meshed is represented as an elastic solid. By solving for the equilibrium configuration, vertices located in the interior are relocated as a result of a prescribed boundary displacement. We will follow this idea in this section.

For each mesh edge, we find the positions of the two points which are located in the one third and two thirds ratio of each edge of the mesh. These positions are original positions of these points before deformation. We find the control points corresponding to the new positions of these points for the interior mesh edges after the mesh is deformed. We deform the mesh such that the boundary vertices of the linear mesh assume the coordinates of the corresponding vertices (with respect to their radial ordering) on the curved boundary. The target coordinates of all the other vertices in the interior are computed by solving an elastic finite element problem [16]. As a result, the elements of the linear mesh are deformed minimally and proportionally to their distance to the points lying on the curved mesh boundary and to the amount of the displacement at these boundary vertices. Fig. 4 illustrates this step. Using the new positions of these points after deformation, the corresponding

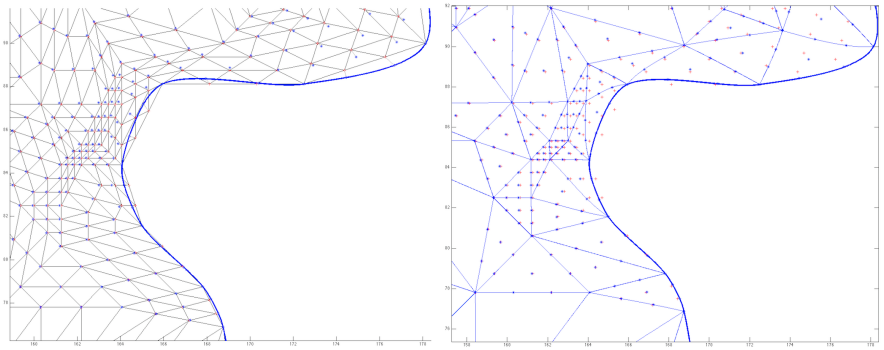

Fig. 4. An illustration of the solid mechanics approach to curved mesh generation.Upper: the bold blue line is curved boundary, the red crosses show the original vertex positions, the blue stars show the new vertex positions after the elements are deformed according to the solution of a nonlinear elasticity problem, the gray segments show the displacements. Lower: elements are deformed according to the equilibrium solution of a nonlinear elasticity problem.

control points which determine the linear edge curving passing through the

points in the new positions can be easily calculated:

$$C_1 = -\frac{5}{6}v_0 + \frac{1}{3}v_1 + 3v_3 - \frac{3}{2}v_4, \tag{9}$$

$$C_2 = \frac{1}{3}v_0 - \frac{5}{6}v_1 - \frac{3}{2}v_3 + 3v_4, \tag{10}$$

where C_1 and C_2 are two middle control points of the four control points of the interior mesh edge, and v_0, v_1, v_2 and v_3 are points the curved edge will pass through.

Validity check is executed after this procedure. In most cases, it can handle this problem successfully. However, in the case that the curvature of the boundary edge is very large, the interior linear edge may not be curved enough to avoid the intersection. Once our validity checking procedure reports that there is an intersected edge, local mesh modifications can be used to correct the shape.

Because each curved edge has two control points in the middle, and each control point determines the curvature of the corresponding half of the edge, we first distinguish which part of the curved edge is intersected. For example, if the left half of the curved edge is intersected, that means the curvature determined by the corresponding control point is not large enough. We enlarge the curvature by rotating the segment formed by the left endpoint and the left control point around the left endpoint by a small angle α. We do it repeatedly until there is no intersection at all.

5 Element validity

A Curvilinear Mesh is valid provided that any two elements do not intersect (except the common vertices and edges). To verify a curved element, we can use explicit intersection checks if the number of elements is small. A cheaper way is detecting the intersection at the element level by evaluating the sign of the determinant of the Jacobian matrix throughout the element. One approach is verifying the positiveness by sampling the Jacobian at discrete locations. A more efficient way is to calculate a lower bound for the determinant of the Jacobian. It is easy to be obtained when the Bézier form is used when mapping a reference element due to its convex hull property [19]. In the case that a positive bound is obtained, it guarantees that the element is valid; on the contrary, when a non-positive bound occurs, the element could be invalid. In this case we need to obtain a tighter bound. This evaluation can be either used to check the validity or to guide the correction of invalid elements.

We rewrite a cubic Bézier triangle $\mathcal{T}^3(u, v, w)$ in the following form:

$$\begin{aligned}\mathcal{T}^3(u,v,w) = {} & P_{300}u^3 + P_{030}v^3 + P_{003}w^3 + 3P_{201}u^2w + 3P_{210}u^2v \\ & + 3P_{120}uv^2 + 3P_{102}uw^2 + 3P_{021}v^2w + 3P_{012}v^2w + 6P_{111}uvw.\end{aligned} \tag{11}$$

The *Jacobian matrix* of a Bézier triangle can be written as

$$J = \begin{bmatrix} \frac{\partial x}{\partial \hat{x}} & \frac{\partial x}{\partial \hat{y}} \\ \frac{\partial y}{\partial \hat{x}} & \frac{\partial y}{\partial \hat{y}} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{T}}{\partial u} & \frac{\partial \mathcal{T}}{\partial v} & \frac{\partial \mathcal{T}}{\partial w} \end{bmatrix} \begin{bmatrix} \frac{\partial u}{\partial \hat{x}} & \frac{\partial u}{\partial \hat{y}} \\ \frac{\partial v}{\partial \hat{x}} & \frac{\partial v}{\partial \hat{y}} \\ \frac{\partial w}{\partial \hat{x}} & \frac{\partial w}{\partial \hat{y}} \end{bmatrix}, \tag{12}$$

with variable change ($u = 1 - \hat{x} - \hat{y}$, $v = \hat{x}$, $w = \hat{y}$),

$$\begin{bmatrix} \frac{\partial u}{\partial \hat{x}} & \frac{\partial u}{\partial \hat{y}} \\ \frac{\partial v}{\partial \hat{x}} & \frac{\partial v}{\partial \hat{y}} \\ \frac{\partial w}{\partial \hat{x}} & \frac{\partial w}{\partial \hat{y}} \end{bmatrix} = \begin{bmatrix} -1 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{13}$$

therefore,

$$J = \begin{bmatrix} \frac{\partial \mathcal{T}}{\partial u} & \frac{\partial \mathcal{T}}{\partial v} & \frac{\partial \mathcal{T}}{\partial w} \end{bmatrix} \begin{bmatrix} -1 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{T}}{\partial v} - \frac{\partial \mathcal{T}}{\partial u} & \frac{\partial \mathcal{T}}{\partial w} - \frac{\partial \mathcal{T}}{\partial u} \end{bmatrix}. \tag{14}$$

Finally,

$$det(J) = (\frac{\partial \mathcal{T}}{\partial v} - \frac{\partial \mathcal{T}}{\partial u}) \times (\frac{\partial \mathcal{T}}{\partial w} - \frac{\partial \mathcal{T}}{\partial u}) \cdot \mathbf{n}, \tag{15}$$

where $\mathbf{n}$ is the vector $(0, 0, 1)$. Because the derivative of a qth order Bézier function is a $(q-1)th$ order Bézier function and the product of two Bézier functions is also a Bézier function, the resulting determinant of *Jacobian* is a Bézier polynomial function with order $2(q-1)$ [9]. In our case, the determinant of *Jacobian* is a forth order Bézier polynomial with fifteen control points. Specifically,

$$\mathcal{T}^4(u, v, w) = \sum_{i+j+k=4} B^4_{ijk}(u, v, w) P_{ijk}, \tag{16}$$

where P_{ijk} is one of the fifteen control values, $B^4_{ijk}(u, v, w) = \frac{4!}{i!j!k!} u^i v^j w^k$, $u \in [0, 1]$, $v \in [0, 1]$and $w \in [0, 1]$ is the barycentric coordinates and $u + v + w = 1$. Because

$$\frac{\partial \mathcal{T}}{\partial v} - \frac{\partial \mathcal{T}}{\partial u} = 3u^2 a_1 + 3v^2 b_1 + 3w^2 c_1 + 6uwd_1 + 6uve_1 + 6vwf_1 \tag{17}$$

and

$$\frac{\partial \mathcal{T}}{\partial w} - \frac{\partial \mathcal{T}}{\partial u} = 3u^2 a_2 + 3v^2 b_2 + 3w^2 c_2 + 6uwd_2 + 6uve_2 + 6vwf_2, \tag{18}$$

where $a_1 = P_{210} - P_{300}$, $b_1 = P_{030} - P_{120}$, $c_1 = P_{012} - P_{102}$, $d_1 = P_{111} - P_{201}$, $e_1 = P_{120} - P_{210}$, $f_1 = P_{021} - P_{111}$, $a_2 = P_{201} - P_{300}$, $b_2 = P_{021} - P_{120}$, $c_2 = P_{003} - P_{102}$, $d_2 = P_{102} - P_{201}$, $e_2 = P_{111} - P_{210}$, $f_2 = P_{012} - P_{111}$, the fifteen control values can be calculated. They are listed in Table 1.

If the element is valid, it means the determinant of the *Jacobian* is positive everywhere in this element. However, if the lower bound of the determinant

Table 1. Fifteen control values for $det(J)$ of a cubic triangle

P_{ijk}	Control Value
P_{400}	$a_1 \times a_2 \cdot \mathbf{n}$
P_{040}	$b_1 \times b_2 \cdot \mathbf{n}$
P_{004}	$c_1 \times c_2 \cdot \mathbf{n}$
P_{220}	$\frac{3}{2}(a_1 \times b_2 \cdot \mathbf{n} + b_1 \times a_2 \cdot \mathbf{n} + 4e_1 \times e_2 \cdot \mathbf{n})$
P_{202}	$\frac{3}{2}(a_1 \times c_2 \cdot \mathbf{n} + c_1 \times a_2 \cdot \mathbf{n} + 4d_1 \times d_2 \cdot \mathbf{n})$
P_{022}	$\frac{3}{2}(b_1 \times c_2 \cdot \mathbf{n} + c_1 \times b_2 \cdot \mathbf{n} + 4f_1 \times f_2 \cdot \mathbf{n})$
P_{301}	$\frac{3}{2}(a_1 \times d_2 \cdot \mathbf{n} + d_1 \times a_2 \cdot \mathbf{n})$
P_{310}	$\frac{3}{2}(a_1 \times e_2 \cdot \mathbf{n} + e_1 \times a_2 \cdot \mathbf{n})$
P_{130}	$\frac{3}{2}(b_1 \times e_2 \cdot \mathbf{n} + e_1 \times b_2 \cdot \mathbf{n})$
P_{031}	$\frac{3}{2}(b_1 \times f_2 \cdot \mathbf{n} + f_1 \times b_2 \cdot \mathbf{n})$
P_{103}	$\frac{3}{2}(c_1 \times d_2 \cdot \mathbf{n} + d_1 \times c_2 \cdot \mathbf{n})$
P_{013}	$\frac{3}{2}(c_1 \times f_2 \cdot \mathbf{n} + f_1 \times c_2 \cdot \mathbf{n})$
P_{211}	$\frac{3}{2}(a_1 \times f_2 \cdot \mathbf{n} + f_1 \times a_2 \cdot \mathbf{n} + 2d_1 \times e_2 \cdot \mathbf{n} + 2e_1 \times d_2 \cdot \mathbf{n})$
P_{121}	$\frac{3}{2}(b_1 \times d_2 \cdot \mathbf{n} + d_1 \times b_2 \cdot \mathbf{n} + 2e_1 \times f_2 \cdot \mathbf{n} + 2f_1 \times e_2 \cdot \mathbf{n})$
P_{112}	$\frac{3}{2}(c_1 \times e_2 \cdot \mathbf{n} + e_1 \times c_2 \cdot \mathbf{n} + 2d_1 \times f_2 \cdot \mathbf{n} + 2f_1 \times d_2 \cdot \mathbf{n})$

of *Jacobian* is non-positive, it does not necessarily mean that the element is invalid. Since it is only a sufficient condition to calculate a lower bound of the determinant of *Jacobian*, sometimes, it is overly-conservative. In the cases that the bound is not tight, the minimum value could be positive whereas the element is reported invalid. To further confirm the answer, we explicitly verify the mesh entities using the Bézier subdivision intersection algorithm. The algorithm relies on the convex hull property and the de Casteljau algorithm [19]. It proceeds by comparing the convex hulls of the two curves. If they do not overlap, it reports immediately that the two curves do not have intersections. If they do overlap, the two curves are subdivided into two in the middle and the two halves of one curve are checked for overlap against the two halves of the other curve. It recursively rejects regions of curves which do not contain intersection points. Once the two curves have been subdivided sufficiently that they can each be approximated by a line segment to within a tolerance, the intersection of the two approximating line segments is found. Other interference checking algorithms such as degree elevation intersection algorithm could also be used, but the Bézier subdivision intersection algorithm is selected here because the convergence of this repeated subdivision process is very fast [20, 21].

6 Geometric and topological fidelity to boundaries

The algorithm we present in this paper offers a mathematical guarantee that the boundary of the high-order mesh it produces is a faithful representation of

the geometric shape within a requested fidelity tolerance. We present proofs of the fact here.

The linear mesh constructed by the method in Sect. 3 provides a faithful representation of the underlying tissues. To measure the distance between the boundaries of the image tissues and the boundaries of the corresponding submesh, we use the two-sided Hausdorff distance. This measure requires that the boundaries of the linear mesh are within the requested tolerance. Below we prove that the curved mesh boundary cannot deviate from the straight mesh boundary by more than a small multiple of the fidelity tolerance, and therefore, for a given value of the fidelity tolerance, we can accommodate both straight and curved deviations. However, the supplied fidelity tolerance must be strictly positive.

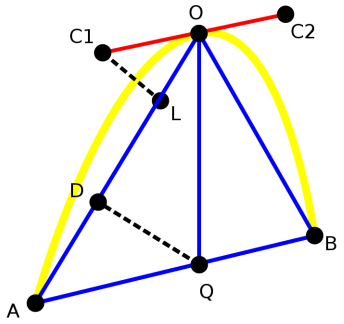

Fig. 5. An illustration of the deviation from the curved edge to the original linear edge. C_1 is a control point of curved edge $\widetilde{AO}$, C_2 is a control point of curved edge $\widetilde{OB}$. Find the point Q such that $|QA|/|QB| = |OA|/|OB|$. $\overline{CL} \perp \overline{OA}$, $\overline{QD} \perp \overline{OA}$.

Since the curved mesh is transformed from the linear mesh, the deviation of the curved edge from the linear edge influences the fidelity. In Fig. 5, let's consider the deviation (say X) of curved edge $\widetilde{AO}$ to linear edge $\overline{AO}$. The length of segment $\overline{C_1C_2}$ controls the curvature of the curve at the Bézier endpoint. Now we need to bound the deviation from the curved edge to the original linear edge. We fix the length of segment $\overline{C_1C_2}$ such that $|\overline{C_1C_2}|$ equals to half of the length of shortest linear edge. Due to the convex hull property, the maximum deviation from the curved edge to the original linear edge is less than the distance from the control point to the linear edge, then we have

$$X < |C_1L| < |C_1O| \tag{19}$$

Therefore, the deviation of the curved edge from the linear edge is bounded, the boundary is completely enclosed by the requested tolerance while maintaining the smoothness.

7 Mesh examples

We apply our algorithm to a variety of examples in the following. For these examples, the input data is a two-dimensional image. The procedure described in Sect. 4.2 was implemented in MATLAB. All the other steps were implemented in C++ for efficiency.

In both of the brain atlas [17] and abdominal atlas [18], the size are 256 * 256 pixels. Each pixel has side lengths of 0.9375 and 0.9375 units in x, y directions, respectively. Table 2 lists the total number of elements in the final meshes of the two examples, number of actual invalid mesh edges and corrected edges. Several figures show the result of each step.

Table 2. Number of detected invalid elements for the two examples below

Image	Elements	Invalid edges	Corrected edges
SPL brain atlas	3034	1	1
SPL abdominal atlas	2025	1	1

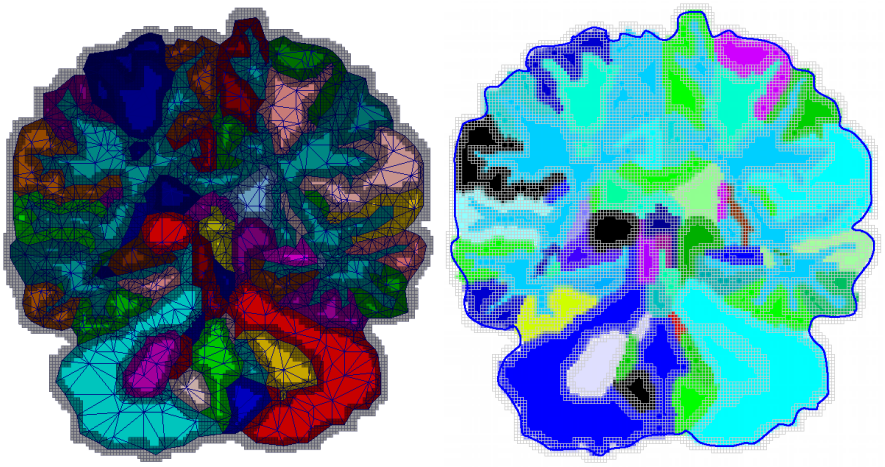

Fig. 6. Results of the first two steps of our algorithm. Left: Linear mesh of a slice of the brain atlas within 2 pixels fidelity tolerance. Right: Smooth curved boundary of a slice of the brain atlas within 2 pixels fidelity tolerance.

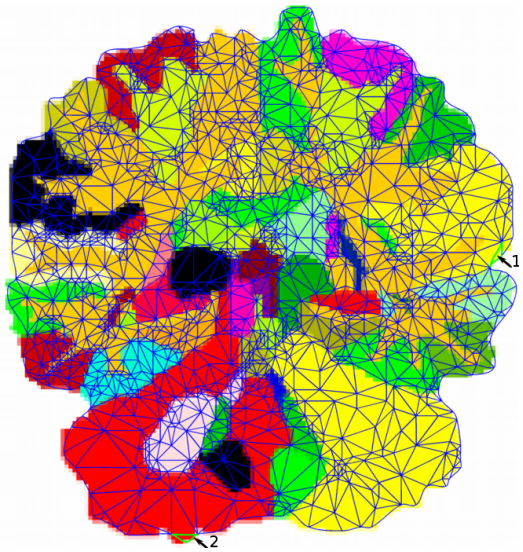

Fig. 7. Result of curving mesh entities in the interior according to the equilibrium solution of a nonlinear elasticity problem. After calculating the lower bound of the determinant of *Jacobian* for each element, two were reported having the non-positive value. The invalid elements are high-lighted in green bold curves.

8 Conclusion

We presented a new approach for automatically constructing a high-order mesh to represent geometry with smooth boundaries with guaranteed fidelity

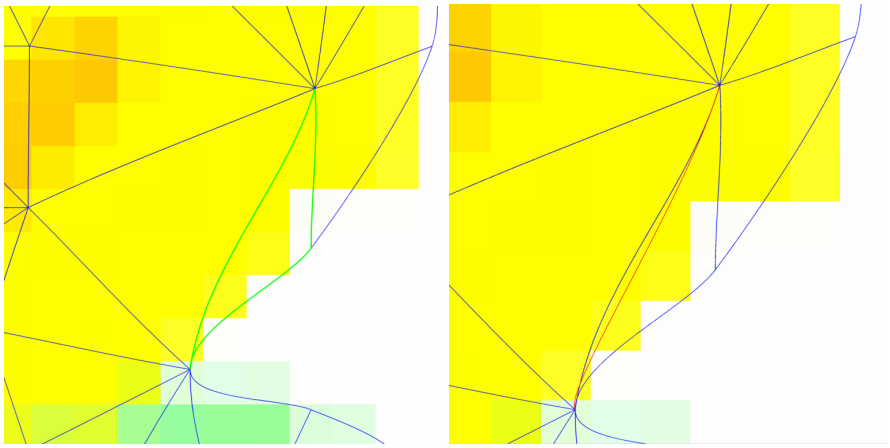

Fig. 8. After the detection of the possible invalid elements, the explicit interference checking procedure reported the intersected mesh edge and correct it accordingly. Left: a zoom in view of an possible invalid element, high-lighted by green bold curves. Right: a zoom in view of the intersected mesh edge corrected by the red curve.

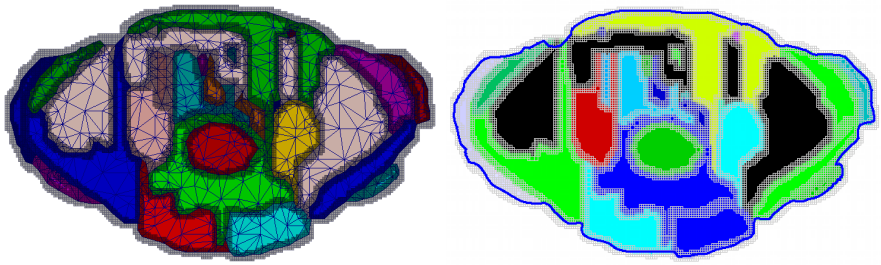

Fig. 9. Results of the first two steps of our algorithm. Left: Linear mesh of a slice of the abdominal atlas within 2 pixels fidelity tolerance. Right: Smooth curved boundary of a slice of the abdominal atlas within 2 pixels fidelity tolerance.

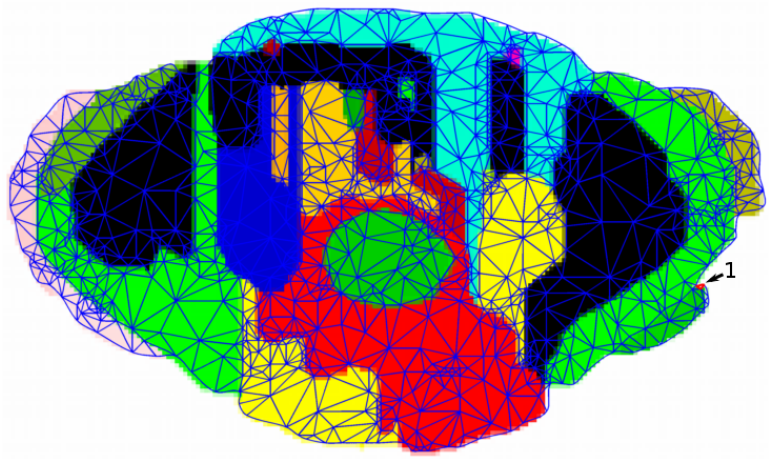

Fig. 10. Result of curving mesh entities in the interior according to the equilibrium solution of a linear elasticity problem. After calculating the lower bound of the determinant of *Jacobian* for each element, only one was reported having the non-positive value. The invalid elements are high-lighted in red bold curves.

and validity. The algorithm we presented is sequential. Our future work includes the development of the corresponding parallel algorithm and the extension to the three-dimensional high-order mesh generation.

9 Acknowledgments

This work was supported (in part) by the Virginia Space Grant Consortium and by the Modeling and Simulation Graduate Research Fellowship Program at the Old Dominion University.

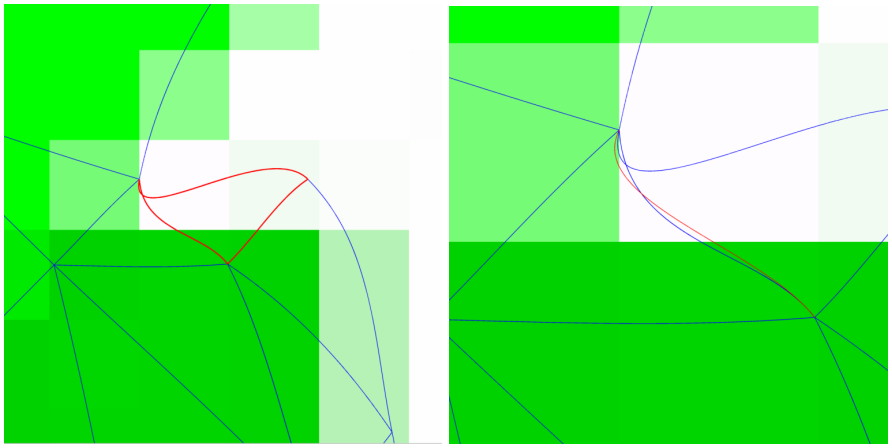

Fig. 11. After the detection of the possible invalid elements, the explicit interference checking procedure reported the intersected mesh edge and correct it accordingly. Left: a zoom in view of an possible invalid element, high-lighted by red bold curves. Right: a zoom in view of the intersected mesh edge corrected by the red curve.

References

1. C.G. Kim and M. Suri (1993) On the p-version of the finite element method in the presence of numerical integration. Numer. Methods. Partial Differential Equations, 9: 593-629
2. I. Babuska and M. Suri (1994) The p and h-p versions of the finite element method, basic priciples and properties. SIAM J.Numer. Anal., 36(4): 578-631
3. I. Babuska and B. Szabo (1997) Trends and New Problems in Finite Element Methods. The Mathematics of Finite Elements and Applications, J.R. Whiteman, Ed, John Wiley and Sons, Chichester, 1-33.
4. Saikat Dey, Mark S. Shephard and Joseph E. Flaherty (1997) Geometry representation issues associated with p-version finite element computations. Computer Methods in Applied Mechanics and Engineering 150(1-4): 39-55
5. George Em Karniadakis and Spencer J. Sherwin (2004) Spectral/hp Element Methods for CFD, second edition. Oxford Univerisity Press, Great Clarendon Street, Oxford.
6. S.J. Sherwin and J. Peiro (2000) Mesh generation in curvilinear domains using high-order elements. Int. J. Numer. Engng 00:1-6
7. Mark S. Shephard, Joseph E. Flaherty and Kenneth E. Jansen (2005) Adaptive mesh generation for curved domains. Applied Numerical Mathematics 52: 251-271
8. Xiao-juan Luo, Mark S. Shephard, Robert M. O'Bara, Rocco Nastasia and Mark W. Beall (2004) Automatic p-version mesh generation for curved domains. Engineering with Computers 20: 273-285
9. Xiao-juan Luo, Mark S. Shephard, Lie-Quan Lee, Lixin Ge, Cho Ng (2011) Moving curved mesh adaptation for high-order finite element simulations. Engineering with Computers 27: 41-50

10. P.L.George and H.Borouchaki (2012) Construction of tetrahedral meshes of degree two. Int. J. Numer. Mesh. Engng 90: 1156-1182
11. A. Johnen, J. F. Remacle and C. Geuzaine(2013) Geometrical validity of curvilinear finite elements. Journal of Computational Physics 233: 359-372
12. Qiukai Lu, Mark S. Shephard, Saurabh Tendulkar and Mark W. Beall (2014) Parallel mesh adaptation for high-order finite element methods with curved element geometry. Engineering with Computers 30: 271-286
13. Andrey Chernikov and Nikos Chrisochoides (2011) Multitissue tetrahedral image-to-mesh conversion with guaranteed quality and fidelity. SIAM Journal on Scientific Computing 33: 3491-3508
14. S. Dey, R.M.O'Bara and M.S. Shephard (2001) Towards curvilinear meshing in 3D: the case of quadratic simplices. Computer-Aided Design 33: 199-209
15. Per-Olof Persson and Jaime Peraire (2009) Curved Mesh Generation and Mesh Refinement using Lagrangian Solid Mechanics. In: Proceedings of the 47th AIAA Aerospace Sciences Meeting and Exhibit, Orlando (FL), USA, January 5-9, 2009.
16. O. C. Zienkiewicz, R. L. Taylor, J. Z. Zhu (2005) The Finite Element Method: Its Basis and Fundamentals. 6th edition. Oxford: Butterworth-Heinemann.
17. I. Talos, M. Jakab, R. Kikinis, and M. Shenton (2008) SPL-PNL brain atlas. http://www.spl.harvard.edu/publications/item/view/1265.
18. I. Talos, M. Jakab, R. Kikinis, and M. Shenton (2008) SPL-PNL abdominal atlas. http://www.spl.harvard.edu/publications/item/view/1266.
19. Gerald Farin (1997) Curves and Surfaces for Computer-Aided Geometric Design. Academic Press.
20. E. Cohen and L. Schumaker (1985) Rates of convergence of control polygons. Computer Aided Geometric Design 2(1-3): 229-235
21. W. Dahmen (1986) Subdivision algorithm converge quadratically. J. of Computational and Applied Mathematics 16: 145-158
22. O. Sahni, X. J. Luo, K. E. Jansen, M. S. Shephard (2010) Curved boundary layer meshing for adaptive viscous flow simulations. Finite Elements in Analysis and Design 46(1-2): 132-139