Automatic Curvilinear Quality Mesh Generation with Smooth Mesh Boundaries of Medical Images

Jing Xu and Andrey N. Chernikov
Department of Computer Science,
Old Dominion University,
Norfolk, VA, USA,
{jxu,achernik}@cs.odu.edu

Keywords: medical image processing, high-order mesh generation, Bézier polynomial basis, smoothness

Abstract

The development of robust high-order finite element methods requires the curvilinear discretization for complex geometries without user intervention. In this work we present a new technique that allows for the automatic construction of high-order curvilinear meshes with two main features: first, the boundary of the mesh is globally smooth, i.e., it satisfies either the C^1 or the C^2 smoothness requirement; second, all elements are valid as measured by their Jacobians. Example meshes demonstrate the features of the algorithm.

1. INTRODUCTION

High-order finite element methods have been used extensively in direct numerical simulations in the last few decades. The exponential rates of convergence, small dispersion and diffusion solution errors have all motivated the development of high-order finite element techniques which better capture the geometry [1, 2, 11]. How well the geometry is approximated has fundamentally important effects on the accuracy of finite element solutions [6, 10]. Therefore, valid meshes with properly curved elements must be constructed to approximate the curved geometric domain.

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 geometric 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 that capture the features of the geometry.

When a geometric domain is given, the common way to accomplish the generation of a curvilinear mesh 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.

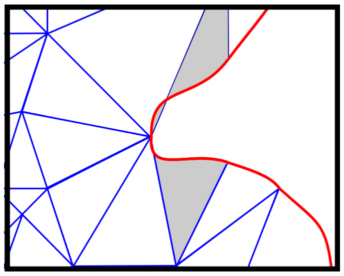


Figure 1: An example of invalid meshes. The red line is the curved mesh boundary, and the blue lines are straight mesh edges in the interior. The curved triangles that are tangled are highlighted in gray.

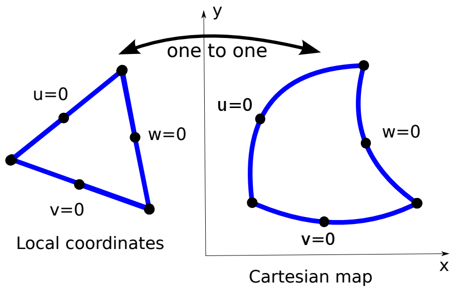


Figure 2: An illustration of the local u, v, w coordinates are distorted into a new, curvilinear set when plotted in global Cartesian x, y space. A general principle for the transformation: a one-to-one correspondence between Cartesian and curvilinear coordinates.

The naive approach does not ensure that all elements of the final curved mesh are valid. The invalid elements are usually caused by curving only the boundary mesh edges while the interior mesh edges remain straight. Fig. 1 gives an example of this critical issue: some of the curvilinear triangular patches have tangled edges. The validity of curved meshes is absolutely crucial to the successful execution of high-order finite element simulations, even one invalid element can ruin the whole simulation. Thus, it is necessary to verify the validity and eliminate all the invalid elements by curving interior mesh edges as a post-processing step once the curved mesh has been constructed.

When elements of the basic types are mapped into distorted forms, a general principle is that a one-to-one correspondence between Cartesian and curvilinear coordinates can be established (illustrated in Fig. 2). If a non-uniqueness occurs, a violent distortion occurs, the mesh would be invalid even if the mesh boundary is an accurate approximation of the geometric domain. The well-known condition for an one-to-one mapping can be decided by the specific property of the *Jacobian* of the transformation that maps a reference element onto each element in the physical domain. A curved element is valid if and only if the sign of the *Jacobian* remains unchanged at all the points of the mapped element, i.e., strictly positive everywhere on this element. To detect invalid elements, one approach is verifying the positiveness by sampling the *Jacobian* at discrete locations [12]. A more accurate way is to calculate a lower bound for the determinant of the Jacobian matrix.

In this paper we build the methodology for automatically generating valid high-order meshes to represent curvilinear domains with smooth global mesh boundaries. Cubic Bézier polynomial basis is selected for the geometric representation of the elements because it provides a convenient framework supporting the smoothing operation and mesh validity verification. We highlight the contributions of this paper:

1. Curved mesh boundary is globally smooth. It satisfies the C^1 or C^2 smoothness requirement.
2. The tight lower bound of the *Jacobian* for each cubic Bézier element is obtained by applying Bézier subdivision algorithm.
3. Our proposed approach is robust in the sense that the invalid elements are eliminated.

The procedure starts with the automatic construction of a linear mesh that simultaneously satisfies the quality (elements do not have arbitrarily small angles) and the fidelity (a reasonably close representation) requirements. The edges of those linear elements which are classified on the boundary are then curved using cubic Bézier polynomials such that these boundary edges constitute a C^1 or C^2 smooth curve. Once the validity verification procedure detects invalid elements, the meshing procedure next curves the interior elements by solving for the equilibrium configuration of an elasticity problem to eliminate the invalid elements.

Various procedures have been developed and implemented to accomplish the generation of a curvilinear mesh. Distinct from our method, Sherwin and Peiro [15] adopted three strategies to alleviate the problem of invalidity: optimization of the surface mesh that accounts for surface curvature, hybrid meshing with prismatic elements near the domain boundaries, curvature driven surface mesh adaption. Persson and Peraire [13] proposed a node relocation strategy for constructing well-shaped curved meshes. They use a nonlinear elasticity analogy, and by solving for the equilibrium configuration, vertices located in the interior are relocated as a result of a prescribed boundary displacement. George and Borouchaki [14] proposed a method for constructing tetrahedral meshes of degree two from a polynomial surface mesh of degree two. *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. Luo et al. [9] isolate singular reentrant model entities, then generate linear elements around those features, and curve them while maintaining the gradation. Modification operations are applied to eliminate invalid elements whenever they are introduced.

The rest of the paper is organized as follows. in Section 2., we review some basic definitions. Section 3. gives a description of the automatic construction of a graded linear mesh and the transformation of the linear mesh into a valid high-order mesh. We present meshing results in Section 4. and conclude in Section 5..

2. PRELIMINARIES

2.1. Bézier curves

We express Bézier curves in terms of Bernstein polynomials. A nth order Bernstein polynomial is defined explicitly by

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

where the binomial coefficients are given by

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

One of the important properties of the Bernstein polynomials is that they satisfy the following recurrence:

$$B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t),$$

with

$$B_0^0(t) \equiv 1, \quad B_j^n(t) \equiv 0 \quad for \quad j \in 0,...,n.$$

Then the Bézier curve of degree n in terms of Bernstein polynomial can be defined recursively as a point-to-point linear combination (linear interpolation) of a pair of corresponding points in two Bézier curves of degree $n-1$. Given a set of points $P_0, P_1, ..., P_n \in E^2$, where E^2 is two-dimensional Euclidean space, and $t \in [0,1]$, set

$$b_i^r(t) = (1-t)b_i^{r-1}(t) + tb_{i+1}^{r-1}(t) \quad \begin{cases} r = 1,...,n \\ i = 0,...,n-r \end{cases}$$

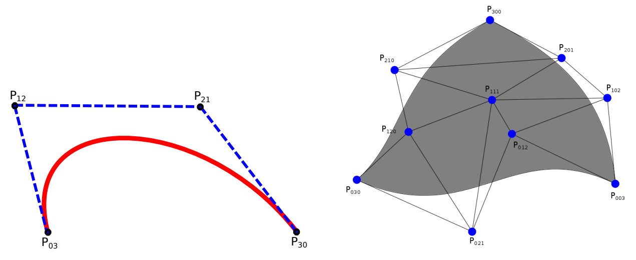


Figure 3: (a) An example of the cubic Bézier curve with its control polygon formed by four control points. (b) An example of the cubic Bézier triangle with its control net formed by ten control points.

and $b_i^0(t) = P_i$. Then $b_0^n(t)$ is the point with parameter value t on the Bézier curve b^n. The set of points $P_0, P_1, ..., P_n$ are called *control points*, and the polygon P formed by points $P_0, P_1, ..., P_n$ is called *control polygon* of the curve b^n.

An explicit form of a n-th order Bézier curve can be defined as

$$b^n(t) = \sum_{i=0}^{n} B_i^n(t) P_i.$$

The barycentric form of Bézier curves demonstrates its symmetry property nicely. Let u and v be the barycentric coordinates, $u \in [0,1]$ and $v \in [0,1]$, $u + v = 1$, then

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

where $B_{ij}^n(u,v) = \frac{n!}{i!j!} u^i v^j$, $P_{ij} \in E^2$ are the control points, and $i + j = n$.

Specifically, the cubic Bézier curve can be written in terms of the barycentric coordinates,

$$\begin{aligned} b^3(u,v) &= \sum_{i+j=3} B_{ij}^3(u,v) P_{ij} \\ &= u^3 P_{03} + 3u^2 v P_{12} + 3uv^2 P_{21} + v^3 P_{30}, \end{aligned} \tag{1}$$

Fig. 3a gives an example of the cubic Bézier curve with its control polygon.

2.2. Bézier triangles

Univariate Bernstein polynomials are the terms of the binomial expansion of $[t + (1-t)]^n$. In the bivariate case, a n-th order Bernstein polynomial is defined by

$$B_{\vec{i}}^n(\vec{u}) = \binom{n}{\vec{i}} u^i v^j w^k,$$

where

$$\vec{i} = \{i, j, k\}, \quad |\vec{i}| = n, \quad \vec{u} = \{u, v, w\},$$

$u \in [0,1]$, $v \in [0,1]$ and $w \in [0,1]$ are the barycentric coordinates and $u + v + w = 1$. It follows the standard convention for the *trinomial coefficients* $\binom{n}{\vec{i}} = \frac{n!}{i!j!k!}$.

This leads to a simple definition of a Bézier triangle of degree n

$$\mathcal{T}^n(u,v,w) = \sum_{i+j+k=n} B_{ijk}^n(u,v,w) P_{ijk},$$

where P_{ijk} is a control point. Specifically, the Bézier triangle of degree three can be written as

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

Fig. 3b gives an example of the cubic triangular patch with its control net formed by its ten control points.

The Bézier shapes possess three important properties [7] which are useful to this work: (1) the convex hull property: a Bézier curve, surface or volume is completely contained in the convex hull formed by its control points; (2) all derivatives and products of Bézier functions are Bézier functions; (3) the convex hull can be refined by Bézier degree elevation algorithm or Bézier subdivision algorithm.

2.3. The Jacobian

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

Let's start at the definition of a finite element. A typical finite element e, $e \in R^n$, is defined by a closed subset of K, $K \in R^n$ with a non empty interior, a set of real-valued functions N defined over the set K, and a finite set of local nodes $u_i, 1 \leq i \leq N$. Then the mapping from the set of local coordinates $\hat{x}$, $\hat{y}$ to a corresponding set of global coordinates x, y is:

$$\vec{u} = \sum_a N_a \vec{u}_a = [N_1, N_2, ..., N_N] \begin{bmatrix} u_1 \\ u_2 \\ ... \\ u_N \end{bmatrix}, \quad a = 1, 2, ..., N,$$

where $\vec{u} = u(x,y)$, and $\vec{u}_a = u_a(\hat{x}, \hat{y})$. The functions N_a are called *shape functions* (or basis functions).

By the *chain rule* of partial differentiation we have

$$\begin{bmatrix} \frac{\partial N_a}{\partial \hat{x}} & \frac{\partial N_a}{\partial \hat{y}} \end{bmatrix} = \begin{bmatrix} \frac{\partial N_a}{\partial x} & \frac{\partial N_a}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial x}{\partial \hat{x}} & \frac{\partial x}{\partial \hat{y}} \\ \frac{\partial y}{\partial \hat{x}} & \frac{\partial y}{\partial \hat{y}} \end{bmatrix} = \begin{bmatrix} \frac{\partial N_a}{\partial x} & \frac{\partial N_a}{\partial y} \end{bmatrix} J,$$

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

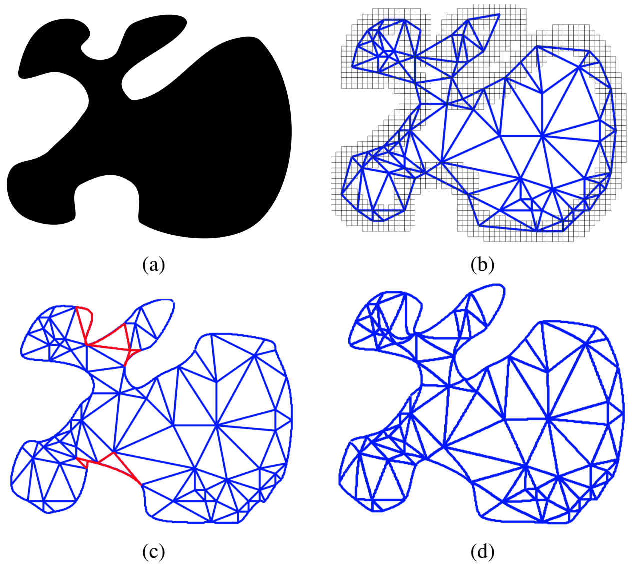


Figure 4: An illustration of the main steps performed by our algorithm. (a) The input two-dimensional image. It shows a curvilinear domain to be meshed. (b) A linear mesh satisfies the user specified quality and fidelity tolerance. The shaded region represents the fidelity tolerance. (c) Those edges that are classified on the linear mesh boundary are curved such that those edges form a smooth curve, and either the C^1 or C^2 smoothness requirement is satisfied. When there are curved edges on the boundary and linear edges in the interior, the mesh validity need to be verified. The red triangles are invalid elements detected by our verifying procedure. (d) To fix these invalid triangles, the interior edges are curved by solving for the equilibrium configuration of an elasticity problem, and a valid mesh is obtained.

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. MESH GENERATION FOR CURVILINEAR DOMAINS

In this section we will describe the proposed algorithm. A comprehensive description of the framework with an extensive evaluation on more data sets is available at [18, 19].

Given a bounded curved domain $\Omega \subset \mathcal{R}^2$, the algorithm outputs a curvilinear mesh of the *interior* of Ω with global smooth boundary. Fig. 4 illustrates the main steps performed by our algorithm. The details are elaborated below.

3.1. Linear mesh construction

To generate the initial linear mesh, we adopt the image-to-mesh conversion algorithm [3], 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 object (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.

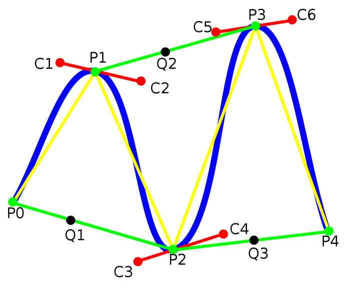


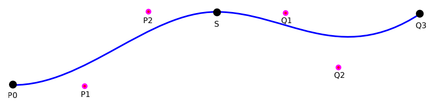
Figure 5: An example of finding control points of a smooth cubic Bézier path. For the curve between P_1 and P_2, we need C_2 and C_3. On segment P_0P_2, find a point Q_1 such that $|P_0Q_1|/|Q_1P_2| = |P_0P_1|/|P_1P_2|$. Translate segment P_0P_2 so that point Q_1 lies on point P_1, and scale the length 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 such that point Q_2 lies on point P_2.

3.2. Smooth boundary construction

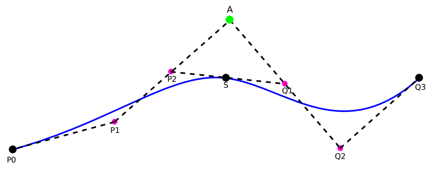
A curve or surface can be described as having C^n continuity, n being the measure of smoothness. Consider the segments on either side of a point on a curve: (1) C^0: The curves touch at the join point; (2) C^1: First derivatives are continuous; (3) C^2: First and second derivatives are continuous.

We aim to find a smooth C^1 curve passing through all the mesh boundary points given in order. A Bézier path is C^1 smooth provided that two Bézier curves share a common tangent direction at the join point. 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. 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 curves at the endpoints. We scale these segments to control the curvature. An example is illustrated in Fig. 5.

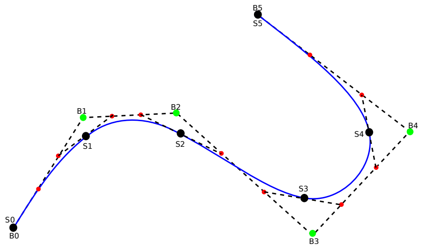
A smooth C^2 curve is a piecewise cubic curve that is composed of pieces of different cubic curves glued together, and it is so smooth that it has a second derivative everywhere and the derivative is continuous. Fig. 6a gives an example of a cu-



(a)



(b)



(c)

Figure 6: An illustration of the construction of the cubic spline curve. (a) An example of a cubic spline curve, formed by two Bézier curves with control points P_0, P_1, P_2, S and S, Q_1, Q_2, Q_3. (b) An A-frame is a structure in which P_2 is the midpoint of $\overline{AP_1}$, Q_1 is the midpoint of $\overline{AQ_2}$ and S is the midpoint of $\overline{P_2Q_1}$. (c) The cubic spline curve constructed with the help of the B-spline points shown in green. The black points are S points, which are the endpoints of the boundary edges of the linear mesh. The red points are the control points need to be calculated to form the cubic spline curve.

bic spline curve. If two Bézier curves with control points P_0, P_1, P_2, S and S, Q_1, Q_2, Q_3 are touched at point S, both their first and second derivatives match at S if and only if their control polygons fit an A-frame, which is a structure in which P_2 is the midpoint of $\overline{AP_1}$, Q_1 is the midpoint of $\overline{AQ_2}$ and S is the midpoint of $\overline{P_2Q_1}$ as Fig. 6b shows. To fit the A-frame in the set of cubic curves, one easy approach is to use B-spline as an intermediate step. In Fig. 6c, the S points (shown in black) are known, they are the endpoints of the boundary edges of the linear mesh. What still needs to be calculated are the red control points. If the B-spline points (the apexes of the A-frames, shown in green) are known, the control points (shown in red) can be easily calculated by computing the one third and two thirds positions between the connection of every two adjacent B-spline points. The B-spline points can be computed by the relationship between S points:

$$6S_i = B_{i-1} + 4B_i + B_{i+1}.$$

By solving a linear system of equations, the coordinates of B-spline points can be obtained.

3.3. Element validity

A curvilinear mesh is valid provided that the intersection of the interiors of two different elements is the null set and any two mesh edges or faces do not intersect each other (except the common vertices or edges). To verify a curved element, one way is detecting the intersection at the element level by evaluating the sign of the *Jacobian* throughout the element. When the Bézier form is used to map a reference element, it is possible to calculate a precise lower bound of the *Jacobian*. Indeed, the *Jacobian* is a Bézier function with order $q = dimension * (degree - 1)$ [8], and its lower bound is easy to be obtained by its convex hull property. In the case that a positive lower bound is obtained, it guarantees that the element is valid; on the contrary, when a non-positive bound occurs, the element may or may not 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.

The Jacobian matrix of a Bézier triangle can be written as

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

with variable change ($u = 1 - \hat{x} - \hat{y}$, $v = \hat{x}$, $w = \hat{y}$) [14],

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

therefore,

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

Finally,

$$det(J) = \left(\frac{\partial \mathcal{T}}{\partial v} - \frac{\partial \mathcal{T}}{\partial u}\right) \times \left(\frac{\partial \mathcal{T}}{\partial w} - \frac{\partial \mathcal{T}}{\partial u}\right) \cdot \vec{n},$$

where $\vec{n}$ 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 *Jacobian* is a Bézier polynomial function with order $2(q-1)$. In our case, the *Jacobian* is a fourth order Bézier polynomial with fifteen control points. Specifically,

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

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

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

P_{ijk}	Control Value
P_{400}	$9(a_1 \times a_2 \cdot \vec{n})$
P_{040}	$9(b_1 \times b_2 \cdot \vec{n})$
P_{004}	$9(c_1 \times c_2 \cdot \vec{n})$
P_{220}	$\frac{3}{2}(a_1 \times b_2 \cdot \vec{n} + b_1 \times a_2 \cdot \vec{n} + 4e_1 \times e_2 \cdot \vec{n})$
P_{202}	$\frac{3}{2}(a_1 \times c_2 \cdot \vec{n} + c_1 \times a_2 \cdot \vec{n} + 4d_1 \times d_2 \cdot \vec{n})$
P_{022}	$\frac{3}{2}(b_1 \times c_2 \cdot \vec{n} + c_1 \times b_2 \cdot \vec{n} + 4f_1 \times f_2 \cdot \vec{n})$
P_{301}	$\frac{9}{2}(a_1 \times d_2 \cdot \vec{n} + d_1 \times a_2 \cdot \vec{n})$
P_{310}	$\frac{9}{2}(a_1 \times e_2 \cdot \vec{n} + e_1 \times a_2 \cdot \vec{n})$
P_{130}	$\frac{9}{2}(b_1 \times e_2 \cdot \vec{n} + e_1 \times b_2 \cdot \vec{n})$
P_{031}	$\frac{9}{2}(b_1 \times f_2 \cdot \vec{n} + f_1 \times b_2 \cdot \vec{n})$
P_{103}	$\frac{9}{2}(c_1 \times d_2 \cdot \vec{n} + d_1 \times c_2 \cdot \vec{n})$
P_{013}	$\frac{9}{2}(c_1 \times f_2 \cdot \vec{n} + f_1 \times c_2 \cdot \vec{n})$
P_{211}	$\frac{3}{2}(a_1 \times f_2 \cdot \vec{n} + f_1 \times a_2 \cdot \vec{n} + 2d_1 \times e_2 \cdot \vec{n} + 2e_1 \times d_2 \cdot \vec{n})$
P_{121}	$\frac{3}{2}(b_1 \times d_2 \cdot \vec{n} + d_1 \times b_2 \cdot \vec{n} + 2e_1 \times f_2 \cdot \vec{n} + 2f_1 \times e_2 \cdot \vec{n})$
P_{112}	$\frac{3}{2}(c_1 \times e_2 \cdot \vec{n} + e_1 \times c_2 \cdot \vec{n} + 2d_1 \times f_2 \cdot \vec{n} + 2f_1 \times d_2 \cdot \vec{n})$

If the element is valid, it means the *Jacobian* is positive everywhere in this element. However, if the computed lower bound of the *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 *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 obtain the tighter bound by refining the convex hull using the Bézier subdivision algorithm. The algorithm relies on the convex hull property and the de Casteljau algorithm [7].

Indeed, if the negative minimum of the fifteen control values corresponds to one of the vertices of the element, then the element is invalid. If not, and the negative minimum of the fifteen control values corresponds to one of the three nodes on the edge, then it is necessary to refine this edge. We use the Bézier subdivision algorithm to split the edge into two subedges. If the negative minimum of the fifteen control values corresponds to one of the three nodes on the face, then it is necessary to refine this face. We use the Bézier subdivision algorithm to split the face into three sub-faces. In this way, the new control polygons are closer to the original polynomial, and the bound becomes much tighter. Other algorithms such as degree elevation could also be used, but the Bézier subdivision algorithm is selected here because the convergence of this repeated subdivision process is very fast [4, 5].

3.4. Mesh untangling

It is usually not enough to curve only the mesh boundary because some control points may be located such that element distortions occur in the interior of the mesh. In such case,

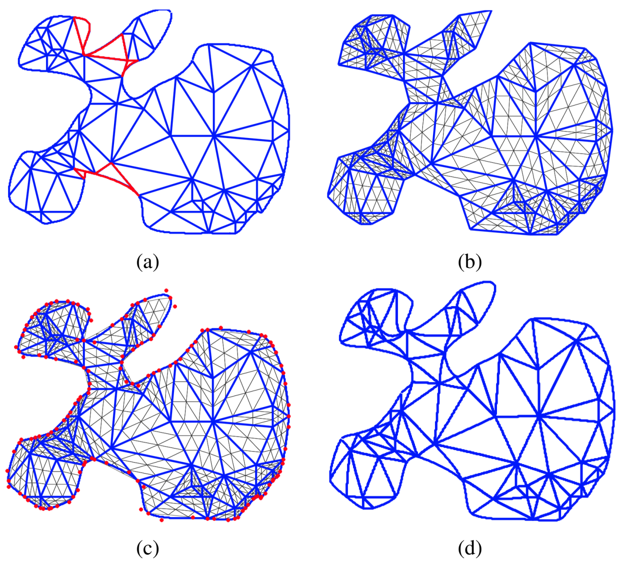


Figure 7: (a) Invalid mesh. (b) The control nets of the linear mesh elements is the undeformed geometry. (c) The red control points of the smooth curved boundary edges are the external loadings. (d) Final configuration.

interior mesh edges should also be curved to eliminate the invalidity or improve the curved element quality.

We move the control points of the interior mesh edges using a finite element method [20]. The geometry of the domain to be meshed is represented as an elastic solid. For each linear mesh edge, the positions of the two points which are located in the one third and two thirds ratio of each edge are computed. These positions are the original positions of the control points of the interior edges before deformation. These points form the control nets of the linear mesh elements. The control nets sticking together as a whole is the undeformed geometry (shown in Fig. 7b). The external loadings are the control points (red points in Fig. 7c) of the smooth curved boundary edges. The control nets are deformed such that the control points of the boundary edges of the linear mesh move to the corresponding control points of the curved boundary edge. By solving for the equilibrium configuration of an elasticity problem, the finial configuration is determined and the new positions of the control points of the interior mesh edges after deformation are obtained. Fig. 7 illustrates these steps.

4. MESH EXAMPLES

We apply our algorithm to two examples in the following. For these examples, the input data is a two-dimensional image. The procedure described in Section 3.4. was implemented in MATLAB. All the other steps were implemented in C++ for efficiency.

In both of the brain atlas [17] and abdominal atlas [16], the

Table 2: Number of invalid elements and corrected elements for the examples below

Image	Total	Invalid	Corrected
SPL brain atlas	3034	34	34
SPL abdominal atlas	2025	19	19

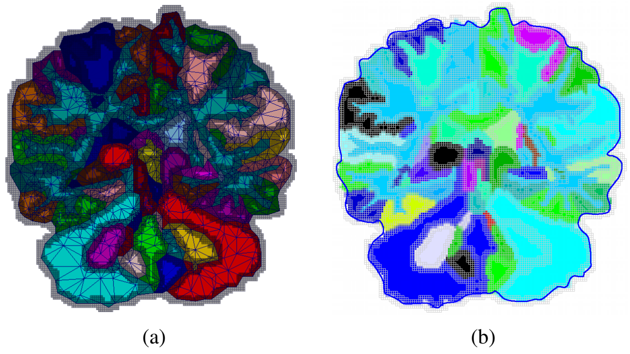

(a) (b)

Figure 8: Results of the first two steps of our algorithm. (a) Linear mesh of a slice of the brain atlas within two pixels fidelity tolerance. (b) Smooth C^2 boundary of a slice of the brain atlas within two pixels fidelity tolerance.

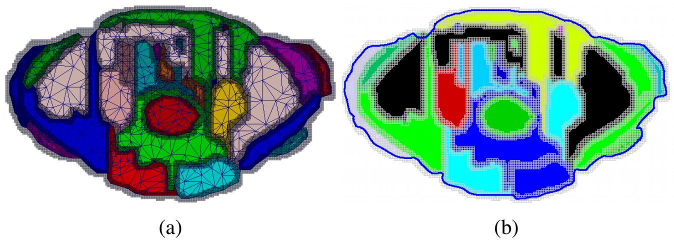
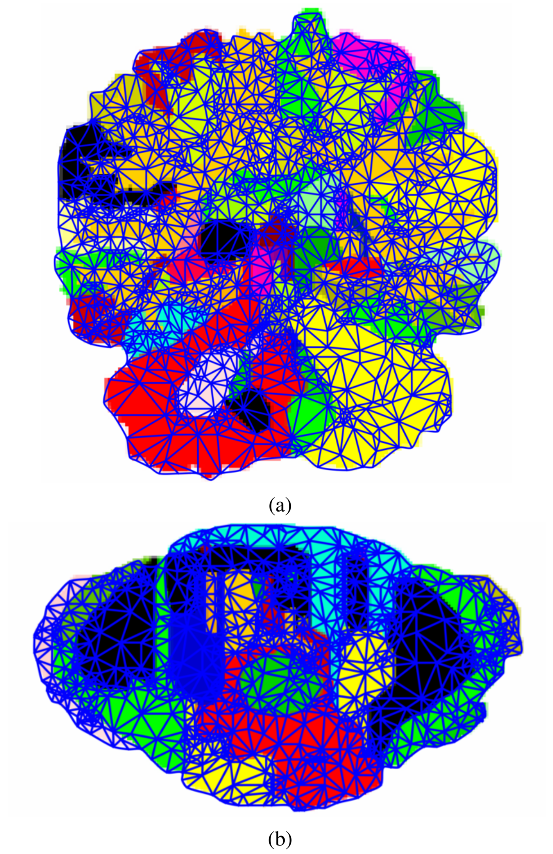

(a) (b)

Figure 9: Results of the first two steps of our algorithm. (a) Linear mesh of a slice of the abdominal atlas within two pixels fidelity tolerance. (b) Smooth C^1 boundary of a slice of the abdominal atlas within two pixels fidelity tolerance.

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, number of actual invalid elements and number of corrected elements in the final meshes of both of the two examples. Several figures show the result of each step.

5. CONCLUSION

We presented a new approach for automatically constructing a guaranteed quality curvilinear mesh to represent geometry with smooth boundaries. 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.


(a)

(b)

Figure 10: Valid high-order meshes for the brain atlas and abdominal atlas.

6. ACKNOWLEDGMENTS

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

REFERENCES

[1] I. Babuska and M. Suri. The p and h-p versions of the finite element method, basic priciples and properties. *SIAM J.Numer*, 36:578–631, 1994.

[2] I. Babuska and B. Szabo. Trends and New Problems in Finite Element Methods. *The Mathematics of Finite Elements and Applications*, pages 1–33, 1997.

[3] Andrey Chernikov and Nikos Chrisochoides. Multitissue tetrahedral image-to-mesh conversion with guaranteed quality and fidelity. *SIAM Journal on Scientific Computing*, 33:3491–3508, 2011.

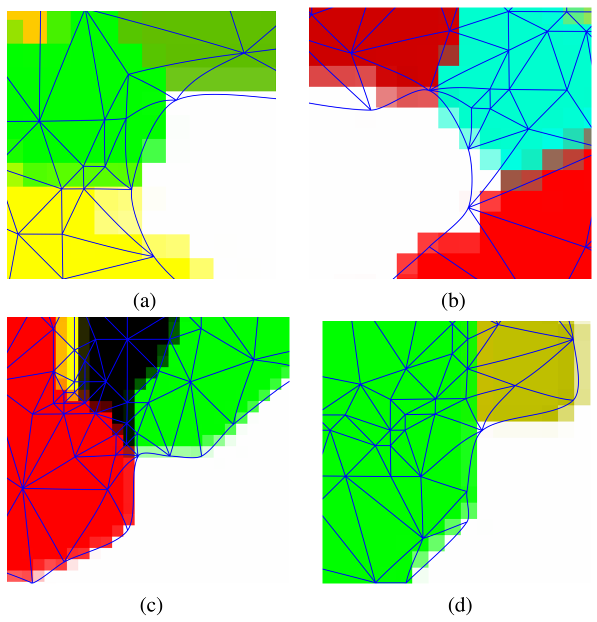


Figure 11: Zoomed-in views of high order valid elements.

[4] E. Cohen and L. Schumaker. Rates of convergence of control polygons. *Computer Aided Geometric Design*, 2:229–235, 1985.

[5] W. Dahmen. Subdivision algorithm converge quadratically. *J. of Computational and Applied Mathematics*, 16:145–158, 1986.

[6] Saikat Dey, Mark S. Shephard, and Joseph E. Geometry representation issues associated with p-version finite element computations. *Computer Methods in Applied Mechanics and Engineering*, 150:39–55, 1997.

[7] Gerald Farin. *Curves and Surfaces for Computer-Aided Geometric Design*. Academic Press, 1997.

[8] Xiao juan Luo, Mark S. Shephard, Lie-Quan Lee, Lixin Ge, and Cho Ng. Moving curved mesh adaptation for high-order finite element simulations. *Engineering with Computers*, 27:41–50, 2011.

[9] Xiao juan Luo, Mark S. Shephard, Robert M. O'Bara, Rocco Nastasia, and Mark W. Beall. Automatic p-version mesh generation for curved domains. *Engineering with Computers*, 20:273–285, 2004.

[10] George Em Karniadakis and Spencer J. Sherwin. *Spectral/hp Element Methods for CFD, second edition*. Oxford Univerisity Press, Great Clarendon Street, Oxford, 2004.

[11] C.G. Kim and M. Suri. On the p-version of the finite element method in the presence of numerical integration. *Numer. Methods*, 9:593–629, 1993.

[12] L. Liu, Y. J. Zhang, T. J. R. Hughes, M. A. Scott, and T. W. Sederberg. Volumetric T-spline construction using Boolean operations. *Engineering with Computers*, 2014.

[13] Per-Olof Persson and Jaime Peraire. Curved Mesh Generation and Mesh Refinement using Lagrangian Solid Mechanics. In *Proceedings of the 47th AIAA Aerospace Sciences Meeting and Exhibit*, Orlando, FL, January 2009.

[14] P.L.George and H.Borouchaki. Construction of tetrahedral meshes of degree two. *Int. J. Numer. Mesh. Engng*, 90:1156–1182, 2012.

[15] S.J. Sherwin and J. Peiro. Mesh generation in curvilinear domains using high-order elements. *Int. J. Numer*, 00:1–6, 2000.

[16] I. Talos, M. Jakab, R. Kikinis, and M. Shenton. Spl abdominal atlas, 2008.

[17] I. Talos, M. Jakab, R. Kikinis, and M. Shenton. Spl-pnl brain atlas, 2008.

[18] Jing Xu and Andrey Chernikov. Automatic curvilinear mesh generation with smooth boundary driven by guaranteed validity and fidelity. In *International Meshing Roundtable*, pages 200–212, London, UK, October 2014. Elsevier.

[19] Jing Xu and Andrey Chernikov. Curvilinear Triangular Discretization of Biomedical Images with Smooth Boundaries. In *International Symposium on Bioinformatics Research and Applications*, Norfolk, VA, July 2015. Springer. To appear.

[20] O. C. Zienkiewicz, R. L. Taylor, and J. Z. Zhu. *The Finite Element Method: Its Basis and Fundamentals, 6th edition*. Oxford: Butterworth-Heinemann, 2005.