Construction of Discrete Descriptions of Biological Shapes through Curvilinear Image Meshing

No Author Given

No Institute Given

Summary. Mesh generation is a useful tool for obtaining discrete descriptors of biological objects represented by images. The generation of meshes with straight sided elements has been fairly well understood. However, in order to match curved shapes that are ubiquitous in nature, meshes with curved (high-order) elements are required. Moreover, for the processing of large data sets, automatic meshing procedures are needed. In this work we present a new technique that allows for the automatic construction of high-order curvilinear meshes. This technique allows for a transformation of straight-sided meshes to curvilinear meshes with C^1 or C^2 smooth boundaries while keeping all elements valid and with good quality as measured by their Jacobians. The technique is illustrated with examples. Experimental results show that the mesh boundaries naturally represent the objects' shapes, and the accuracy of the representation is improved compared to the corresponding linear mesh.

Key words: biomedical image processing, high-order mesh generation, Bézier polynomial, Jacobian, finite element method

1 Introduction

Discretizations of complex shapes into simple elements are widely used in various computing areas that require a quantitative analysis of spatially dependent attributes. One, traditional, area is the finite element analysis [14] which is used to numerically solve partial differential equations derived using solid mechanics and computational fluid dynamics approaches. With this approach one starts with the knowledge of the constitutive physical laws and initial (boundary) conditions and obtains a prediction of the observable properties of objects of interest. Another, emerging, area is the use of discretizations for delineating homogeneous spatial zones within objects that can be represented as units for an overall object description. With this approach one starts with the knowledge of observable object properties and uses statistical methods to infer the processes that govern the formation of the object. Therefore, the

second approach can be viewed as a reversal of the first approach, that still relies on a similar discretization technique. This second approach is a useful tool for bioinformatics applications, for example gene expression pattern analysis [5, 6, 12, 13].

Said discretizations of objects are usually called meshes, and the simple elements that they consist of are either triangles and tetrahedra (in two and three dimensions, respectively), or quadrilaterals and hexahedra. Furthermore, elements can have either straight or curved sides. In previous work [12, 13] the authors used triangular meshes with straight sides to discretize images of fruit fly embryos. However, the embryos, like most biological objects, have curved shapes, and their discretizations with straight-sided elements have limited accuracy. To obtain much higher accuracy one needs to use curved-sided elements that match the curves of object boundaries.

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 smooth operation and mesh validity verification. We highlight the three contributions of this paper:

1. Curved mesh boundary is globally smooth. It satisfies C^1 or C^2 smoothness requirement chosen by the user.
2. The proposed approach is robust in the sense that the invalid elements are eliminated, and the mesh quality is enforced.
3. The method provides higher accuracy compared to the linear discretization.

The procedure starts with the automatic construction of a linear mesh. 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 smooth closed curve. Once our validity verification procedure detects invalid elements, the method next curves the interior elements by iteratively solving for the equilibrium configuration of an elasticity problem until all the invalid elements are eliminated.

Various procedures have also been developed and implemented by other authors to accomplish the generation of a curvilinear mesh. Sherwin and Peiro [11] adopted three strategies to alleviate the problem of invalidity: generating boundary conforming surface meshes that account for curvature; the use of a hybrid mesh with prismatic and tetrahedral elements near the domain boundaries; refining the surface meshes according to the curvature. The mesh spacing is decided by a user defined tolerance ϵ related to the curvature and a threshold to stop excessive refinement. In the present work we develop a method that allows for an all triangle mesh which simplifies and unifies both meshing and analysis. Persson and Peraire [9] proposed a node relocation strategy for constructing well-shaped curved meshes. Compared to our method which iteratively solves for the equilibrium configuration of a linear

elasticity problem, 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. Luo et al. [7] isolate singular reentrant model entities, then generate linear elements around those features, and curve them while maintaining the gradation. Local mesh modifications such as minimizing the deformation, edge or facet deletion, splitting, collapsing, swapping as well as shape manipulation are applied to eliminate invalid elements whenever they are introduced instead of our global node relocation strategy. George and Borouchaki [10] 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. In contrast, our method does not require a starting curved boundary mesh, as well as produces more flexible cubic elements.

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 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 n-th 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

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

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

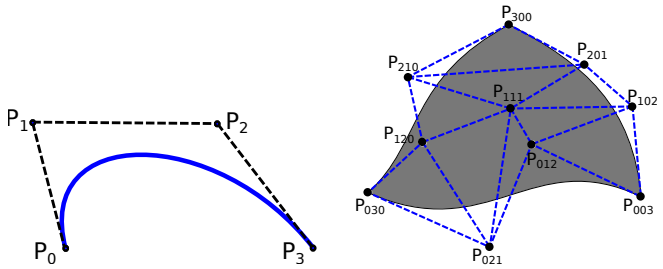
$$B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t),$$

with

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

Now the Bézier curve of degree n can be defined in terms of Bernstein polynomials as

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



(a) (b)

Fig. 1: (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.

where 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.

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

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

where $B^n_{ij}(u,v) = \frac{n!}{i!j!}u^i v^j$, $P_{ij} \in \mathcal{R}^2$ are the control points. Note that this and the following equations are in fact two equations corresponding to the two spatial coordinates.

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

$$b^3(u,v) = \sum_{i+j=3} B^3_{ij}(u,v)P_{ij} = u^3P_{03} + 3u^2vP_{12} + 3uv^2P_{21} + v^3P_{30}.$$

Fig. 1a 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

$$B^n_{\mathbf{i}}(\mathbf{u}) = \binom{n}{\mathbf{i}} u^i v^j w^k,$$

where

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

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

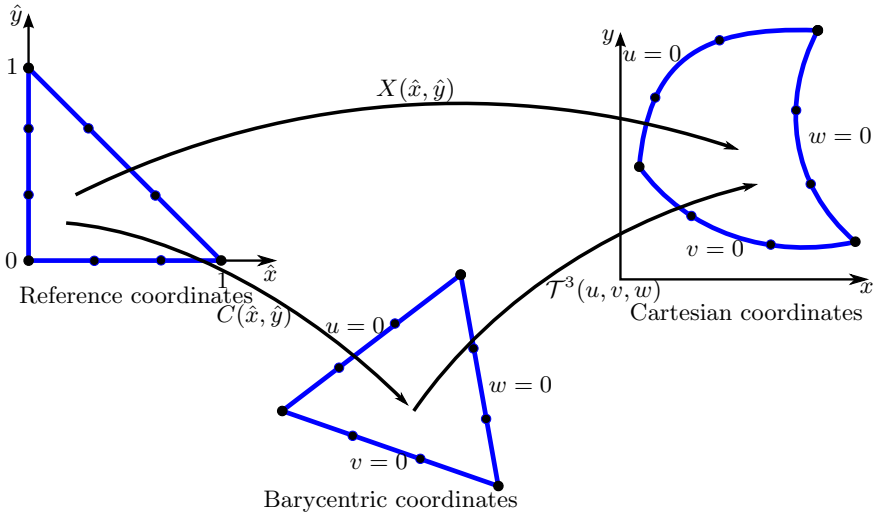


Fig. 2: Reference unit triangle in local coordinates $(\hat{x},\hat{y})$ and the mappings $\mathbf{X}(\hat{x},\hat{y})$, $\mathbf{C}(\hat{x},\hat{y})$ and $\mathcal{T}(u,v,w)$. A general principle for the transformations: an one-to-one correspondence between coordinates systems.

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

$$\mathcal{T}^n(\mathbf{u}) = \sum_{i+j+k=n} B^n_{\mathbf{i}}(\mathbf{u}) P_{\mathbf{i}},$$

where the set of points $P_{\mathbf{i}}$ are *control points*, and the net N formed by points $P_{\mathbf{i}}$ is called *control net* of the Bézier triangle $\mathcal{T}^n$.

Specifically, the Bézier triangle of degree three can be written as

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

Fig. 1b gives an example of the cubic triangular patch with its control net formed by its ten control points.

2.3 The Jacobian

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

Because the cubic Bézier triangle is defined in terms of the barycentric coordinates (u,v,w) with the form:

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

the reference triangle is first mapped to a triangle in barycentric coordinates (by the mapping $\mathbf{C}(\hat{x},\hat{y})$) and then mapped to a curved triangle in global (x,y) coordinates (by the mapping $\mathcal{T}(u,v,w)$). This two-step mapping is presented in Fig. 2.

The mapping should be bijective, because there should not be overlapped regions inside the element. This implies that the sign of the *Jacobian* of the transformation has to be strictly positive everywhere on this element. *Jacobian* is the determinant of the Jacobian matrix J which is defined by all first-order partial derivatives of the transformation:

$$J = \begin{bmatrix} \frac{\partial \mathcal{T}^n_x}{\partial \hat{x}} & \frac{\partial \mathcal{T}^n_x}{\partial \hat{y}} \\ \frac{\partial \mathcal{T}^n_y}{\partial \hat{x}} & \frac{\partial \mathcal{T}^n_y}{\partial \hat{y}} \end{bmatrix}.$$

3 Mesh generation for curvilinear domains

Given a bounded curved domain $\Omega \subset \mathcal{R}^2$, the algorithm outputs a curvilinear mesh of the *interior* of Ω with globally smooth boundary. Fig. 3 illustrates the main steps performed by our algorithm. The details are elaborated below.

3.1 Linear mesh construction

The mesh has to provide a close approximation of the object shape, and we measure the closeness by the fidelity tolerance, the two-sided Hausdorff distance from the mesh to the image and the image to the mesh. For image boundary I and mesh boundary M, the one-sided distance from I to M is given by

$$h(I,M) = \max_{i\in I} \min_{m\in M} d(i,m),$$

where $d(\cdot,\cdot)$ is the regular Euclidean distance. The one-sided distance from M to I is given similarly by

$$h(M,I) = \max_{m\in M} \min_{i\in I} d(m,i).$$

The two-sided distance is:

$$H(I,M) = \max\{h(I,M), h(M,I)\}.$$

The initial linear mesh is generated by the modified quad-tree based image-to-mesh conversion algorithm [3], which satisfies the following requirements:

1. It allows for guaranteed fidelity; i.e., the two-sided Hausdorff distance from the mesh to the image and the image to the mesh is within a user-specified fidelity tolerance.

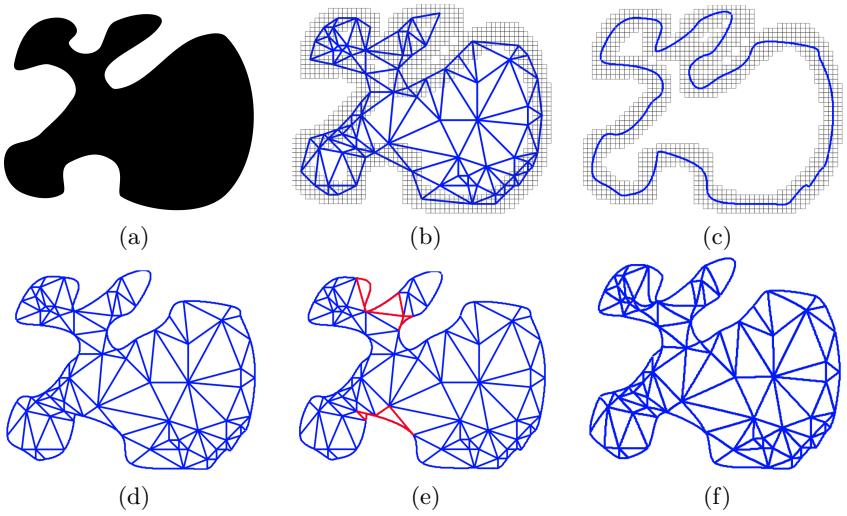


Fig. 3: An illustration of the main steps performed by our algorithm. (a) The input two-dimensional image. (b) The linear mesh for the original image. The shaded regions are the regions within the user specified fidelity tolerance. (c) Curved mesh boundary satisfying C^1 or C^2 smoothness requirement. (d) Smooth curved boundary with linear edges in the interior. (e) The red triangles are invalid elements detected by the verifying procedure. (f) Valid high quality curvilinear mesh obtained by iteratively solving for the equilibrium configuration of an elasticity problem.

2. It allows for topological fidelity; i.e., the mesh maintains the topology of the original image.
3. It can either generate a mesh with almost equal-sized elements inside (except the boundary elements) or coarsen the mesh to a much lower number of elements with gradation in the interior, decided by the application.

3.2 Smooth boundary construction

We aim to find a smooth curve interpolating all the mesh boundary vertices given in order. A curve 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 segments touch at the joint point; (2) C^1: First derivatives are continuous at the joint point; (3) C^2: First and second derivatives are continuous at the joint point.

A smooth C^1 piecewise cubic curve is composed of pieces of different cubic curves glued together, and it has a first derivative everywhere and the derivative is continuous. A Bézier path is C^1 smooth provided that two Bézier curves share a common tangent direction at the joint point. The basic idea is to calculate control points around each endpoint so that they lie in a straight line

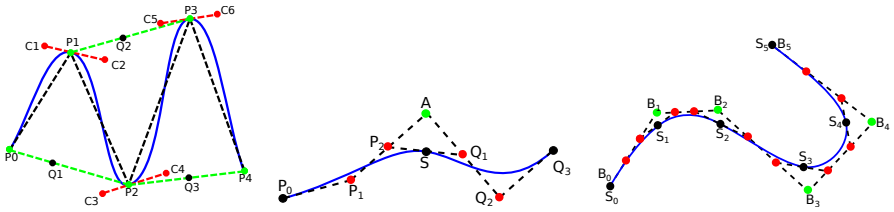


Fig. 4: An illustration of the construction of the C^1 and C^2 smooth curves. (a) An example of finding control points of a smooth C^1 path. (b) If two Bézier curves touch at a joint point, both their first and second derivatives match at the joint point if and only if their control polygons fit an A-frame. (c) The cubic spline curve is constructed with the help of the green B-spline points.

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 the cubic Bézier form (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. 4a. 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.

The cubic Bézier form provides enough degrees of freedom to construct a cubic spline curve that satisfies C^2 smoothness requirement. Since the curvature of a point on a curve is a function with respect to the first and second derivative of this point, and if the first and the second derivative are continuous, then the curvature at this point is continuous. We prefer C^2 smooth curve to C^1 smooth curve because the boundary of the biomedical objects usually have continuous curvatures. If two Bézier curves with control points P_0, P_1, P_2, S and S, Q_1, Q_2, Q_3 touch 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. 4b 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. 4c, the junction points S (shown in black) are mesh vertices that are classified on the boundary of the linear mesh. If the B-spline points B (the apexes of the A-frames, shown in green) are known, the control points (shown in red) can be calculated by computing the one third and two thirds positions between the connection of every two adjacent B-spline points. The

B-spline points B and the junction points S satisfy a relationship:

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

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

3.3 Element validity

The invalid elements are usually caused by curving only the boundary mesh edges while the interior mesh edges remain straight. Some of the curvilinear triangular patches may have tangled edges. Thus, it is necessary to verify the validity and to eliminate all the invalid elements by curving interior mesh edges as a post-processing step once the curved mesh has been constructed. One approach to verify the positiveness of the *Jacobian* is sampling the *Jacobian* at discrete locations such as at Gaussian points [8]. A more precise way is to calculate the tight lower bound for the *Jacobian*.

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

P_{ijk}	Control Point
P_{400}	$9(a_1 \times a_2 \cdot \mathbf{n})$
P_{040}	$9(b_1 \times b_2 \cdot \mathbf{n})$
P_{004}	$9(c_1 \times c_2 \cdot \mathbf{n})$
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}	$\frac{9}{2}(a_1 \times d_2 \cdot \mathbf{n} + d_1 \times a_2 \cdot \mathbf{n})$
P_{310}	$\frac{9}{2}(a_1 \times e_2 \cdot \mathbf{n} + e_1 \times a_2 \cdot \mathbf{n})$
P_{130}	$\frac{9}{2}(b_1 \times e_2 \cdot \mathbf{n} + e_1 \times b_2 \cdot \mathbf{n})$
P_{031}	$\frac{9}{2}(b_1 \times f_2 \cdot \mathbf{n} + f_1 \times b_2 \cdot \mathbf{n})$
P_{103}	$\frac{9}{2}(c_1 \times d_2 \cdot \mathbf{n} + d_1 \times c_2 \cdot \mathbf{n})$
P_{013}	$\frac{9}{2}(c_1 \times f_2 \cdot \mathbf{n} + f_1 \times c_2 \cdot \mathbf{n})$
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}	$\frac{3}{2}(c_1 \times e_2 \cdot \mathbf{n} + e_1 \times c_2 \cdot \mathbf{n} + 2d_1 \times f_2 \cdot \mathbf{n} + 2f_1 \times d_2 \cdot \mathbf{n})$

Since the Bézier basis is selected to represent the elements, the tight lower bound can be calculated with the help of its special properties such as the convex hull property and subdivision property [4]. We can write the Jacobian matrix J of a cubic Bézier triangle as:

$$
\begin{aligned}
J &= \begin{bmatrix} \frac{\partial \mathcal{T}_x^3}{\partial u} & \frac{\partial \mathcal{T}_x^3}{\partial v} & \frac{\partial \mathcal{T}_x^3}{\partial w} \\ \frac{\partial \mathcal{T}_y^3}{\partial u} & \frac{\partial \mathcal{T}_y^3}{\partial v} & \frac{\partial \mathcal{T}_y^3}{\partial w} \end{bmatrix} \begin{bmatrix} \frac{\partial u}{\partial \hat{x}} & \frac{\partial u}{\partial \hat{y}} \\ \frac{\partial v}{\partial \hat{x}} & \frac{\partial v}{\partial \hat{y}} \\ \frac{\partial w}{\partial \hat{x}} & \frac{\partial w}{\partial \hat{y}} \end{bmatrix} \\
&= \begin{bmatrix} \frac{\partial \mathcal{T}_x^3}{\partial u} & \frac{\partial \mathcal{T}_x^3}{\partial v} & \frac{\partial \mathcal{T}_x^3}{\partial w} \\ \frac{\partial \mathcal{T}_y^3}{\partial u} & \frac{\partial \mathcal{T}_y^3}{\partial v} & \frac{\partial \mathcal{T}_y^3}{\partial w} \end{bmatrix} \begin{bmatrix} -1 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} \frac{\partial \mathcal{T}_x^3}{\partial v} - \frac{\partial \mathcal{T}_x^3}{\partial u} & \frac{\partial \mathcal{T}_x^3}{\partial w} - \frac{\partial \mathcal{T}_x^3}{\partial u} \\ \frac{\partial \mathcal{T}_y^3}{\partial v} - \frac{\partial \mathcal{T}_y^3}{\partial u} & \frac{\partial \mathcal{T}_y^3}{\partial w} - \frac{\partial \mathcal{T}_y^3}{\partial u} \end{bmatrix},
\end{aligned}
$$

and the determinant of J can be represented as:

$$det(J) = (\frac{\partial \mathcal{T}^3}{\partial v} - \frac{\partial \mathcal{T}^3}{\partial u}) \times (\frac{\partial \mathcal{T}^3}{\partial w} - \frac{\partial \mathcal{T}^3}{\partial u}) \cdot \mathbf{n},$$

where $\mathbf{n}$ is the vector $(0, 0, 1)$. Because the derivative of a cubic Bézier polynomial is a quadratic Bézier polynomial, and the product of two quadratic Bézier polynomials is a fourth order Bézier polynomial, the *Jacobian* is a fourth order Bézier polynomial with fifteen control points. Note that this *Jacobian* function is a scalar-valued function, and the control points are scalar values.

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

where P_{ijk} is one of the fifteen control points, $B_{ijk}^4(u, v, w) = \frac{4!}{i!j!k!} u^i v^j w^k$, $u \in [0, 1]$, $v \in [0, 1]$ and $w \in [0, 1]$ are the barycentric coordinates and $u+v+w = 1$. Therefore, the fifteen control points of the *Jacobian* can be represented by the control points of the cubic Bézier triangle. Because

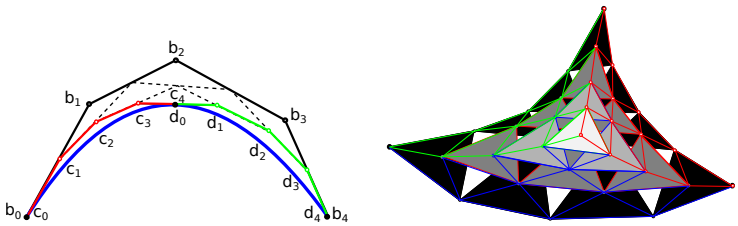
$$\frac{\partial \mathcal{T}}{\partial v} - \frac{\partial \mathcal{T}}{\partial u} = 3u^2 a_1 + 3v^2 b_1 + 3w^2 c_1 + 6uwd_1 + 6uve_1 + 6vwf_1$$

and

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

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 points can be calculated. They are listed in Table 1.

Due to the convex hull property, the Bézier polynomial is completely contained in its convex hull formed by the control points, thus, the minimum of the fifteen control points is the lower bound of the *Jacobian*. If the lower bound is positive, then the element is valid. However, if the lower bound is non-positive, it does not necessarily mean that the element is invalid. Since it is only a sufficient condition, sometimes it is overly conservative. In the cases



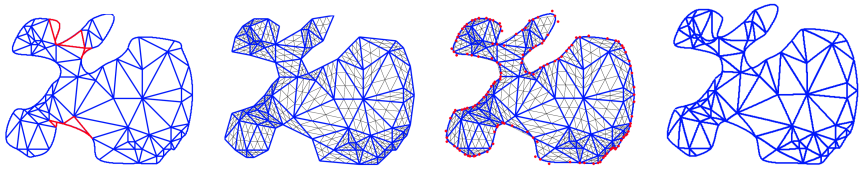
(a) (b)

Fig. 5: An illustration of the recursive subdivision algorithm for Bézier polynomials. (a) The recursive subdivision algorithm for the fourth order Bézier curve. The two new control polygons are shown in red and green. (b) The recursive subdivision algorithm for the fourth order Bézier triangle. The three new control nets are shown in red, green and blue.

that the lower bound is not tight, the minimum value of the *Jacobian* could be positive whereas the calculated lower bound is non-positive. To further confirm the answer, we obtain the tighter bound by refining the convex hull using the Bézier subdivision algorithm [4]. Fig. 5 shows the recursive subdivision algorithm for the fourth order Bézier curve and the fourth order Bézier triangular face. The control points that located on the ends of the new polynomials that obtained by the Bézier subdivision algorithm are always the points on the original polynomial, thus, if one of them is non-positive, the element contains at least one non-positive *Jacobian*. Therefore, if the non-positive minimum of the fifteen control points corresponds to one of the three vertices of the fourth order Bézier triangle (the *Jacobian*), then the element can be reported invalid immediately. If not, and if the non-positive minimum of the fifteen control points corresponds to one of the three nodes on one of the three edges of the fourth order Bézier triangle, then it is necessary to refine this edge. The Bézier subdivision algorithm recursively splits the edge into two sub-edges. Compared to the original convex hull, the two new convex hulls are closer to the original polynomial. If the non-positive minimum of the fifteen control points corresponds to one of the three nodes on the face, then it is necessary to refine this face. The Bézier subdivision algorithm recursively splits the face into three sub-faces. In this way, the three new convex hulls are closer to the original polynomial, and the bound becomes much tighter.

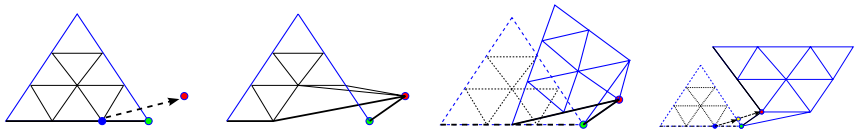
3.4 Mesh untangling

It is usually not enough to curve only the mesh boundary because some control points may be located such that invalid elements occur. In such case, edges in the interior of the mesh should also be curved to eliminate the invalidity or to improve the curved element quality.



(a) (b) (c) (d)

Fig. 6: (a) Invalid mesh with red invalid elements. (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 loads. (d) The final configuration is determined by solving for the equilibrium configuration of an elasticity problem.

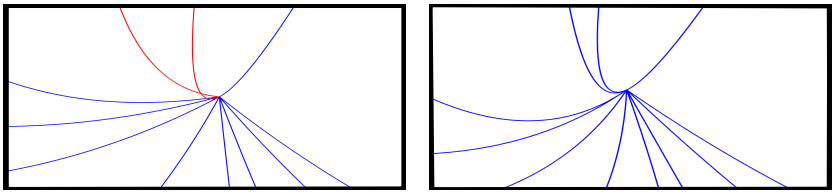


(a) (b) (c) (d)

Fig. 7: An illustration of the iterative finite element method. (a) The mesh composed of one element. (b) The invalid mesh with twisted control net. (c) The one-step FE method was applied, but the control net is still twisted. (d) The iterative FE method successfully corrected the twisted control net.

We move the control points of the interior mesh edges using a finite element method [14]. The geometry of the domain to be meshed is represented as an elastic solid. For each linear mesh edge, the two points which are located in the one third and two thirds ratio of each edge are computed. These points together with the mesh vertices are the original positions of the control points of the edges before deformation. These points form the control nets of the linear mesh elements. The control nets together as a whole is the undeformed geometry (shown in Fig. 6b). The external loads are the displacements of the control points (red points in Fig. 6c) of the smooth curved boundary edges. The control nets are deformed such that when the control points of the boundary edges of the linear mesh moved to the corresponding control points of the curved boundary edge, the new positions of the control points of the interior mesh edges are determined by solving for the equilibrium configuration of an elasticity problem. Fig. 6 illustrates these steps.

In some cases, the one step finite element method can handle this problem successfully. However, in the case that the curvature of the boundary edge is very large, the interior edges may not be able to be curved enough to correct the invalidity. The iterative finite element method successfully solves this problem. Fig. 7 illustrates the iterative FE method. In this example there is only one element in the mesh, the black border line represents the mesh



(a) (b)

Fig. 8: A comparison of the result of one-step FE method and the result of the iterative FE method. (a) After one-step FE method, the two red edges are still tangled together. (b) After eight iterations, the edges are untangled, all the elements are valid.

boundary, the blue point represents one control point of the linear boundary edge. The red point represents the corresponding control point of the curved boundary edge. The green point is one of the mesh vertices on the mesh boundary, thus it has to maintain its position. The control net is invalid because there exists an inverted triangle. When one step FE method was applied, the blue point was directly moved to the red point. After solving for the equilibrium configuration, the control net is still twisted. However, when the yellow point was made the intermediate displacement, the blue point was first moved to the yellow point, then moved to the red point, the two iteration FE method successfully corrected the twisted control net.

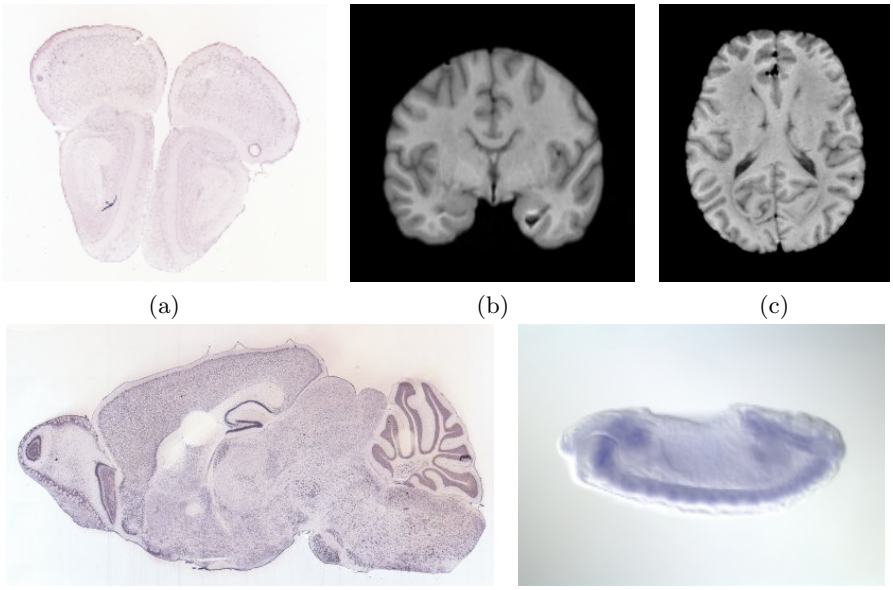
The iterative FE method executes the validity check before each round. When it is reported that an invalid element exists, the procedure divides the segments formed by the control points of the linear boundary edges and the corresponding control points of the curved edges. The procedure takes the endpoints of the subsegments one by one as the intermediate external loadings, and takes the solution of the current external loadings as the undeformed geometry of the next external loadings. The algorithm terminates when all the invalid elements are corrected. Fig. 8 shows an example of the comparison of the result of one-step FE method and the result of the iterative FE method.

4 Mesh examples

The input data to our algorithm is a two-dimensional image. The procedure for mesh untangling and quality improvement was implemented in MATLAB. All the other steps were implemented in C++ for efficiency.

In the following mesh examples, we meshed two slices of the mouse brain image [2], two slices of the human brain image [2], and the fly embryo image [1]. The first slice of the mouse brain image (Mouse Brain I) has the size $198 * 169$ pixels; the second slice of the mouse brain image (Mouse Brain II) has the size $460 * 247$ pixels; the first slice of the human brain image (Human Brain I) has

the size $239 * 233$ pixels; the second slice of the human brain image (Human Brain II) has the size $235 * 283$ pixels; the fly embryo image (Fly Embryo) has the size $182 * 130$ pixels. Each pixel has side lengths of 1 unit in both x, y directions. The original images are listed in Fig. 9.



(a) (b) (c)

(d) (e)

Fig. 9: The original images. (a) The first slice of the mouse brain image. (b) The first slice of the human brain image. (c) The second slice of the human brain image. (d) The second slice of the mouse brain image. (e) The fly embryo image.

We show the linear mesh results for the original images with different requirements (in Fig. 10). For the first slice of the mouse brain image, the fidelity tolerance was specified by 3 pixels; for the second slice of the mouse brain image, the fidelity tolerance was specified by 6 pixels; for the first slice of the human brain image, the fidelity tolerance was specified by 4 pixels; for the second slice of the mouse brain image, the fidelity tolerance was specified by 3 pixels; for the fly embryo image, the fidelity tolerance was specified by 2 pixels. For all the linear mesh results, the mesh vertices that are classified on the mesh boundary were required to be located on the boundary between the background and the tissue of the image. This requirement results in different angle bounds for the linear mesh results: the minimum angle bound of the first slice of the mouse brain image is 3.6°, of the second slice is 2.8°; the minimum angle bound of the first slice of the human brain image is 3.2°, of

the second slice is 5.4°; the minimum angle bound of the fly embryo image is 2.8°. The minimum angle bound is an important measure to the quality of the linear mesh (the higher the better), and it also directly contributes to the quality of the curvilinear mesh. For the curved meshes, the quality can not be measured just simply by calculating the planar angles, however, it can be measured by *scaled Jacobian* [14]. The lower minimum angle bound for the linear mesh could lead to worse *scaled Jacobian* after curving the linear mesh boundary to a smooth closed path, however, the *scaled Jacobian* can be improved by the iterative FE method. For all the linear mesh results, the elements were not coarsened.

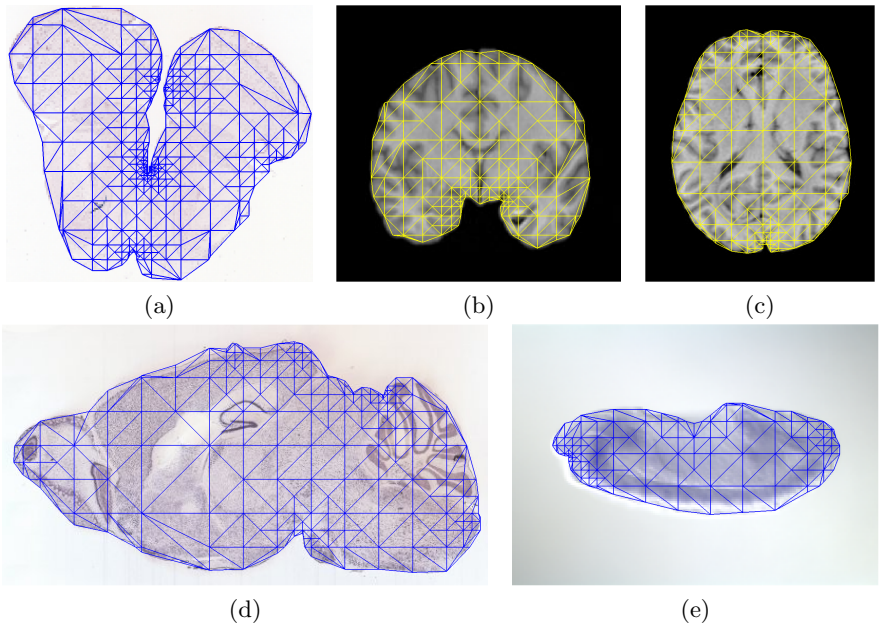


Fig. 10: The linear meshes. (a) The linear mesh result for the first slice of the mouse brain image. (b) The linear mesh result for the first slice of the human brain image. (c) The linear mesh result for the second slice of the human brain image. (d) The linear mesh result for the second slice of the mouse brain image. (e) The linear mesh result for the fly embryo image.

For each of the above linear meshes, we show the linear mesh boundaries and the curved boundaries with both C^1 and C^2 smoothness requirements. In Fig. 11, from left to right for each image, the boundaries are linear boundaries, C^1 boundaries and C^2 boundaries.

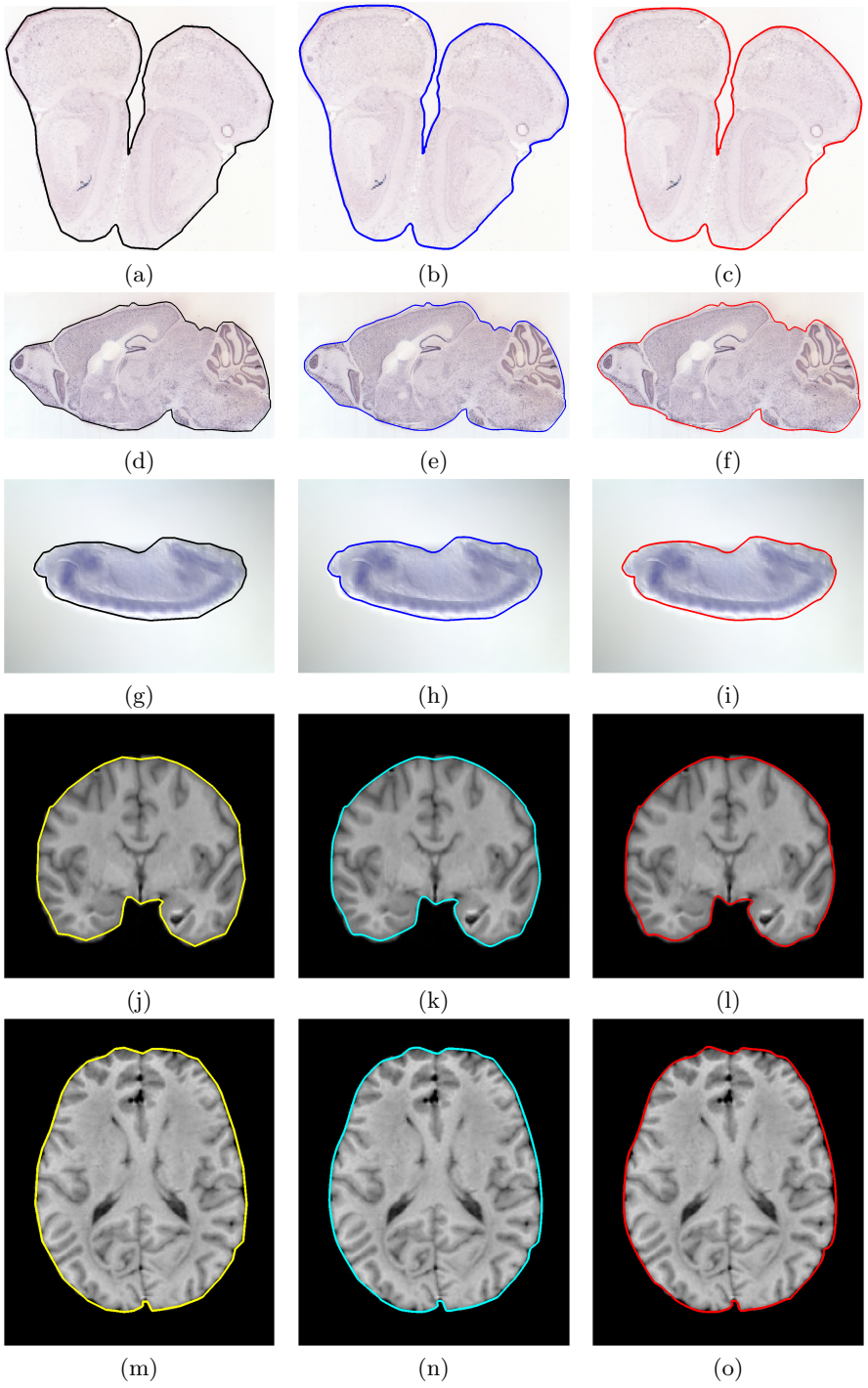


Fig. 11: The linear mesh boundaries and curved mesh boundaries with C^1 and C^2 smoothness requirements for the original images.

The accuracy was specified by the number of misclassified pixels that composed of background pixels that are inside the mesh and tissue pixels that are outside the mesh. The accuracy of the linear mesh results and the corresponding curvilinear meshes with C^1 and C^2 smoothness requirements are listed in Table 2.

Table 2: Accuracy of the mesh boundaries

	Mouse Brain I				
Example	NBPIM	NTPOM	NMP	PNMP (%)	PIA (%)
Linear boundary	73	308	381	1.139	N/A
C2 boundary	128	166	294	0.879	22.835
C1 boundary	95	205	300	0.897	21.260
	Mouse Brain II				
Example	NBPIM	NTPOM	NMP	PNMP (%)	PIA (%)
Linear boundary	293	1073	1366	1.202	N/A
C2 boundary	243	691	934	0.822	31.625
C1 boundary	260	687	947	0.834	30.673
	Human Brain I				
Example	NBPIM	NTPOM	NMP	PNMP (%)	PIA (%)
Linear boundary	70	376	446	0.800	N/A
C2 boundary	138	229	307	0.659	31.166
C1 boundary	111	268	319	0.681	28.475
	Human Brain II				
Example	NBPIM	NTPOM	NMP	PNMP (%)	PIA (%)
Linear boundary	147	314	461	0.693	N/A
C2 boundary	215	216	431	0.648	6.508
C1 boundary	206	201	407	0.648	11.714
	Fly Embryo				
Example	NBPIM	NTPOM	NMP	PNMP (%)	PIA (%)
Linear boundary	39	101	140	0.592	N/A
C2 boundary	59	83	120	0.507	14.286
C1 boundary	52	89	123	0.520	12.143

For each of the original image with linear meshing result and the corresponding C^1 and C^2 smooth boundaries, we list the number of background pixels inside the mesh (NBPIM), the number of tissue pixels outside the mesh (NTPOM), the total number of misclassified pixels (NMP), the percentage for misclassified pixels out of all pixels (PNMP) and the improved accuracy in percentage for both C^1 and C^2 smooth boundaries compared to the linear mesh boundary (PIA). Compare the improved accuracy in percentage (PIA) in Table 2, both C^1 and C^2 smooth boundaries improved the accuracy of the representation. The improved accuracy also relates to the size of the dataset, usually the larger the image, the more improvement its curvilinear mesh obtained. However, if the linear mesh is a very close representation of the image

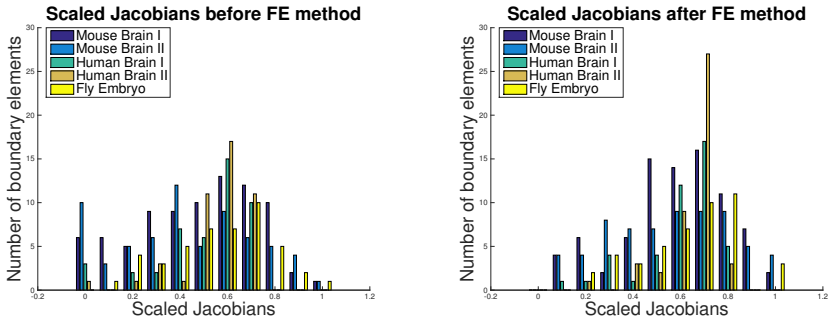
object, after smoothing the mesh boundary, the accuracy can not improve much. Compare the improved accuracy of the meshes that have C^1 smooth boundaries with those of the meshes that have C^2 smooth boundaries, the C^2 smooth boundaries usually have higher accuracy than the C^1 smooth boundaries, but the differences are not large. We chose the results that have better accuracy to construct the final valid high quality meshes.

When the linear mesh boundaries were curved to closed smooth paths, and the interior mesh edges remained straight, the invalid elements were created. The number of invalid elements for the first slice of the mouse brain image is 6, for the second is 10. The number of invalid elements for the first slice of the human brain image is 3, for the second is 1. The invalid elements are shown in red in Fig. 13a, Fig. 13c, Fig. 15a and Fig. 15c. The iterative FE method was applied to the invalid meshes. After 6, 25, 5, 5 iterations, all the invalid elements were eliminated for these invalid meshes. For the fly embryo image, there is no invalid element (Fig. 14a). We executed 10 iterations to improve the quality of the elements. The final meshes are shown in Fig. 13b, Fig. 13d, Fig. 15b, Fig. 15d, and Fig. 14b.

The quality of the curvilinear meshes was also improved by the iterative FE method. The measure *scaled Jacobian* is defined by:

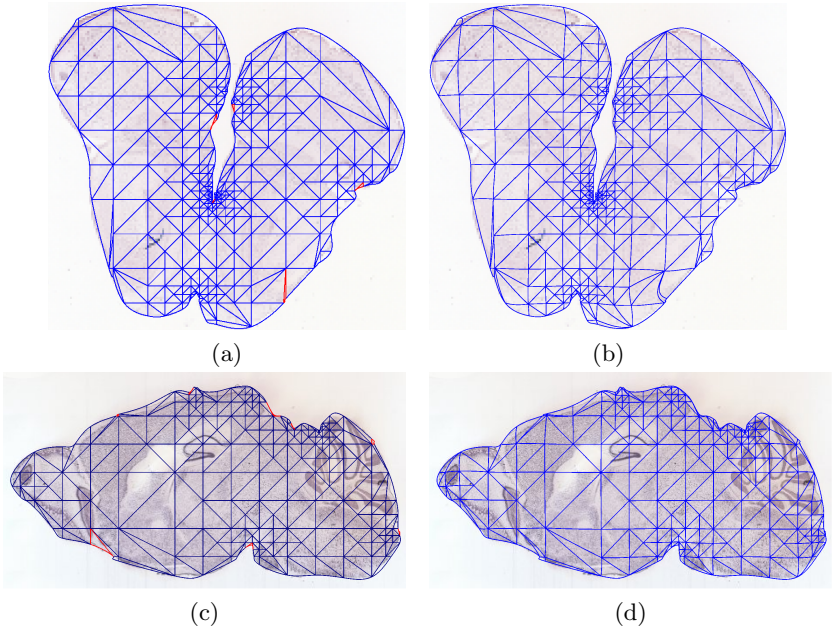
$$I = \frac{min|J|}{max|J|},$$

where $|J|$ is the *Jacobian* of the mapping from the reference coordinates to the physical coordinates. For a straight-sided element, since its *Jacobian* is a constant, $I = 1$; for a curved element, $I \leq 1$. When the curved element is invalid, I is negative; when it gets degenerated, I approaches to 0. From Fig. 12, the iterative FE method produced more elements with larger *scaled Jacobian*, thus the bad shaped elements were improved largely.



(a) (b)

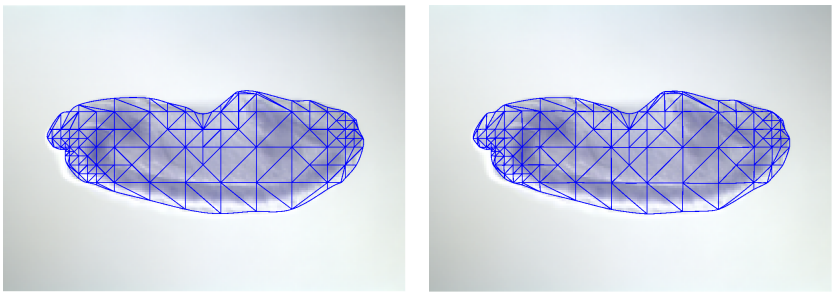
Fig. 12: The comparison of the *scaled Jacobian*. (a) The *scaled Jacobian* before iterative FE method (the negative *scaled Jacobian* were set to be 0 for representation convenience). (b) The *scaled Jacobian* after iterative FE method.



(a) (b)

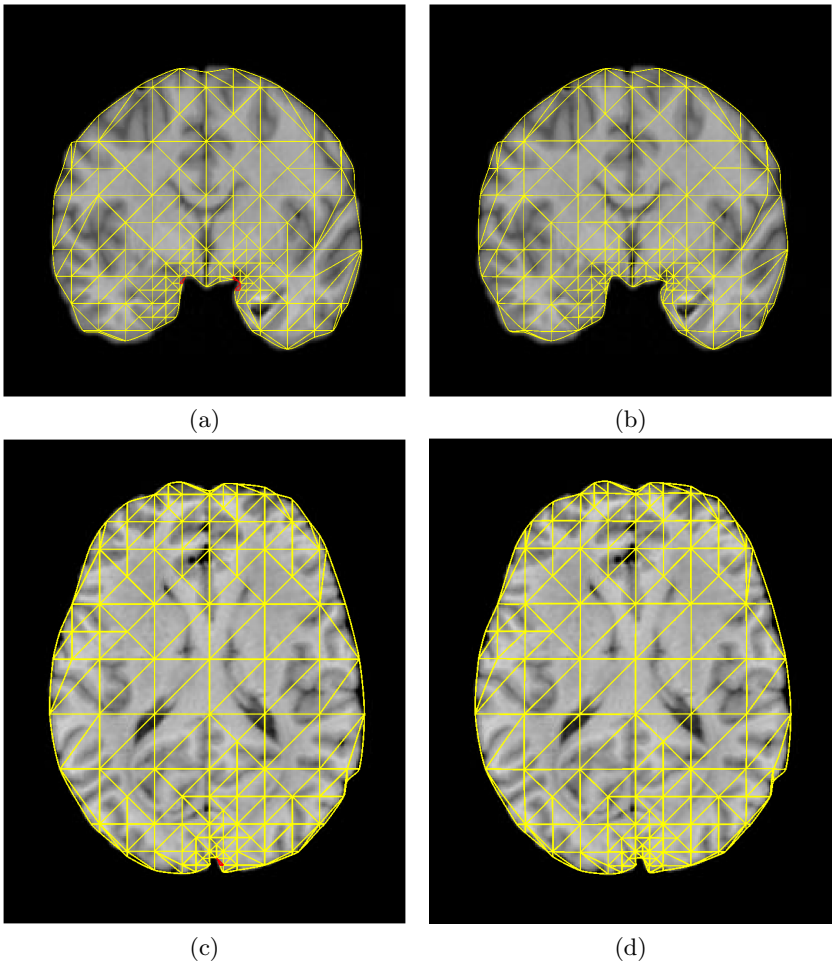
(c) (d)

Fig. 13: Invalid meshes and corresponding corrected meshes for the two slices of the mouse brain image. (a) Invalid curvilinear mesh for the first slice of the mouse brain image. (b) Valid final curvilinear mesh with quality improvement for the first slice of the mouse brain image. (c) Invalid curvilinear mesh for the second slice of the mouse brain image. (d) Valid final curvilinear mesh with quality improvement for the second slice of the mouse brain image.



(a) (b)

Fig. 14: Bad quality curvilinear mesh for the fly embryo image and corresponding mesh with quality improvement. (a) Bad quality curvilinear mesh for the fly embryo image. (b) Improved quality curvilinear mesh for the fly embryo image.



(a) (b)

(c) (d)

Fig. 15: Invalid meshes and corresponding corrected meshes for the two slices of the human brain image. (a) Invalid curvilinear mesh for the first slice of the human brain image. (b) Valid final curvilinear mesh with quality improvement for the first slice of the human brain image. (c) Invalid curvilinear mesh for the second slice of the human brain image. (d) Valid final curvilinear mesh with quality improvement for the second slice of the human brain image.

The algorithm can also construct curved meshes with coarsened elements inside that have fewer elements. Fig. 16 shows the coarsened curvilinear meshes for the five original images.

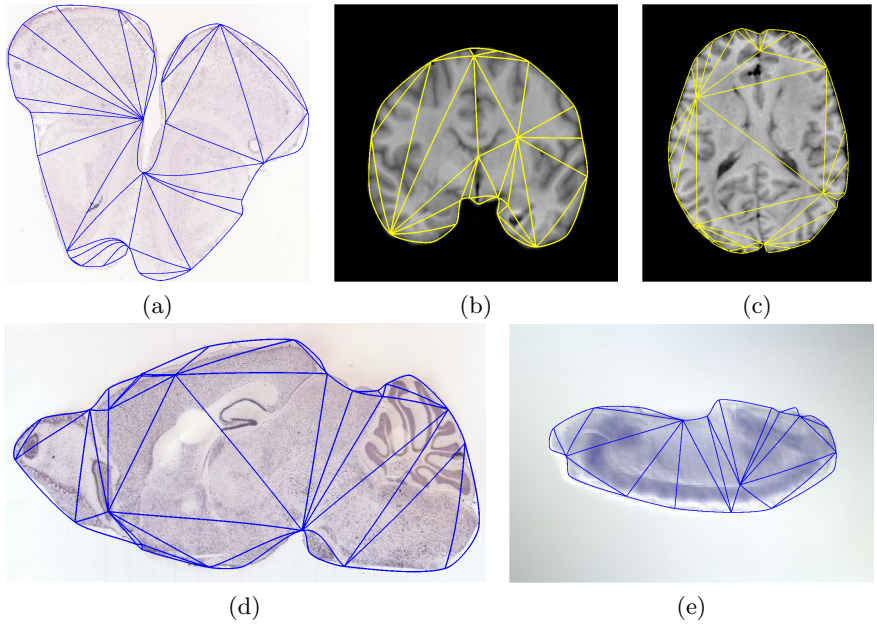


Fig. 16: curvilinear meshes with coarsened elements for the five original images.

In Table 3, we list the total number of elements inside the mesh (TNE), the number of invalid elements (NIE), the iterations needed to improve the quality of the mesh (ITRS), the run time of the linear mesh (RTL), the time spent on FE method (TFE) and the total run time (TRT). The high-order mesh generator is slower, and most of the time was spent on the FEM iterations. The run time is not only decided by the number of elements inside the mesh, but also determined by how many iterations it needs, because when there are highly distorted invalid elements, more iterations are needed to correct them.

5 Conclusion

We presented a new approach for automatically constructing a quality curvilinear mesh to represent geometry with smooth boundaries. The algorithm we presented is sequential. Our future work includes the multitissue triangular curvilinear mesh construction, the development of the corresponding parallel algorithm and the extension to the three-dimensional high-order mesh generation.

Table 3: Run time (s) for the ten examples

Example	TNE	NIE	ITRS	RTL (s)	TFE (s)	TRT (s)
Mouse Brain I (fine)	528	6	6	0.169	88.457	89.703
Mouse Brain II (fine)	373	10	25	0.467	221.336	224.917
Human Brain I (fine)	251	3	5	0.239	23.890	24.785
Human Brain II (fine)	235	1	5	0.262	23.212	25.691
Fly Embryo (fine)	213	0	10	0.111	42.373	44.357
Mouse Brain I (coarse)	39	4	10	0.164	10.199	13.122
Mouse Brain II (coarse)	37	5	16	0.518	12.298	15.445
Human Brain I (coarse)	27	3	12	0.238	16.860	19.079
Human Brain II (coarse)	39	1	80	0.266	40.500	42.391
Fly Embryo (coarse)	21	3	8	0.119	10.685	14.366

References

1. Berkeley drosophila genome project, 2014. http://www.fruitfly.org/.
2. P. Allen. Allen brain atlas, 2014. http://www.brain-map.org.
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.
4. Gerald Farin. *Curves and Surfaces for Computer-Aided Geometric Design*. Academic Press, 1997.
5. Erwin Frise, Ann S Hammonds, and Susan E Celniker. Systematic image-driven analysis of the spatial drosophila embryonic expression landscape. *Molecular systems biology*, 6(1), 2010.
6. Manjunatha Jagalur, Chris Pal, Erik Learned-Miller, R Thomas Zoeller, and David Kulp. Analyzing in situ gene expression in the mouse brain with image registration, feature extraction and block clustering. *BMC bioinformatics*, 8(Suppl 10):S5, 2007.
7. 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.
8. 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.
9. 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.
10. P.L.George and H.Borouchaki. Construction of tetrahedral meshes of degree two. *Int. J. Numer. Mesh. Engng*, 90:1156–1182, 2012.
11. S.J. Sherwin and J. Peiro. Mesh generation in curvilinear domains using high-order elements. *Int. J. Numer*, 00:1–6, 2000.
12. Wenlu Zhang, Daming Feng, Rongjian Li, Andrey Chernikov, Nikos Chrisochoides, Christopher Osgood, Charlotte Konikoff, Stuart Newfeld, Sudhir Kumar, and Shuiwang Ji. A mesh generation and machine learning framework for Drosophila gene expression pattern image analysis. *BMC Bioinformatics*, 14:372, 2013.

13. Wenlu Zhang, Rongjian Li, Daming Feng, Andrey Chernikov, Nikos Chrisochoides, Christopher Osgood, and Shuiwang Ji. Evolutionary soft co-clustering: formulations, algorithms, and applications. *Data Mining and Knowledge Discovery*, pages 1–27, 2014.
14. O. C. Zienkiewicz, R. L. Taylor, and J. Z. Zhu. *The Finite Element Method: Its Basis and Fundamentals, 6th edition.* Oxford: Butterworth-Heinemann, 2005.