Fidelity and Quality Improvement of Curvilinear Image Meshing on Medical Images

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

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

Abstract

Mesh generation is a useful tool for obtaining discrete descriptors of medical objects represented by images. Different from the conventional meshes with all straight-sided elements, the curvilinear meshes match curved shapes of medical objects that are ubiquitous in nature very well. However, the fidelity (accuracy of the representation) of the mesh boundaries and the quality (measured by Jacobians) of the mesh elements could be deteriorated when transforming straight-sided meshes to curvilinear meshes. In this work we present a technique that allows for the automatic construction of high-order curvilinear meshes with C^1 or C^2 smooth boundaries. By carefully designing the linear mesh generator, the fidelity is improved compared to the corresponding linear mesh. The paper also provides a technique as a post-processing step that corrects all the invalid elements and improves the mesh quality as measured by their Jacobians. The technique is illustrated with examples and data analysis.

1. INTRODUCTION

The use of discretizations for delineating homogeneous spatial zones within objects that can be represented as units for an overall object description is an emerging computing area that requires a quantitative analysis of spatially dependent attributes. 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. It is a useful tool for biomedical applications, for example gene expression pattern analysis [11, 12, 4, 5].

In our previous work [11, 12] we used triangular meshes with straight sides to discretize images of fruit fly embryos. However, the embryos, like most biomedical 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.

Various procedures have also been developed and implemented by other authors to accomplish the generation of a curvilinear mesh. Sherwin and Peiro [9] 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 [7] 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. [6] 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 [8] 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.

In this paper we build the methodology for automatically generating high quality curvilinear meshes with smooth global mesh boundaries to represent curvilinear domains with higher accuracy. 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 two contributions of this paper:

1. The proposed approach is robust in the sense that all the invalid elements are eliminated, and the mesh quality is

much enforced.

2. The method provides higher accuracy compared to the linear discretization.

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

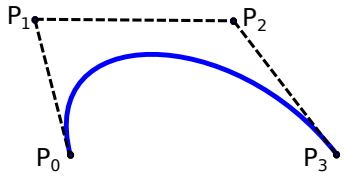


Figure 1. An example of the cubic Bézier curve with its control polygon formed by four control points.

We express Bézier curves in terms of Bernstein polynomials. Let u and v be the barycentric coordinates, $u \in [0,1]$ and $v \in [0,1]$, $u+v=1$, the n-th order Bernstein polynomial is defined explicitly by

$$B_{ij}^n(u,v) = \binom{n}{i} u^i v^j, \quad i = 0,\ldots,n, \quad j = n-i,$$

where the binomial coefficients are given by

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

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

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

where the set of points $P_{ij} \in \mathcal{R}^2$ are called *control points*, and the polygon P formed by points P_{ij} is called *control polygon* of the curve b^n. 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:

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

Fig. 1 gives an example of the cubic Bézier curve with its control polygon.

2.2. Bézier triangles

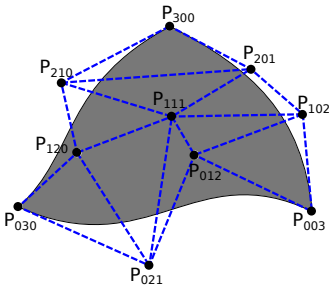


Figure 2. An example of the cubic Bézier triangle with its control net formed by ten control points.

A Bézier triangle of degree n can be defined similarly:

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

where $B_{\vec{i}}^n(\vec{u})$ is a n-th order Bernstein polynomial in the bivariate case and

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

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* $\binom{n}{\vec{i}} = \frac{n!}{i!j!k!}$. The set of points $P_{\vec{i}}$ are *control points*, and the net N formed by points $P_{\vec{i}}$ is called *control net* of the Bézier triangle $\mathcal{T}^n$.

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

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

Fig. 2 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.

Jacobian is the determinant of the Jacobian matrix J which is defined by all first-order partial derivatives of the transformation:

$$J = \begin{bmatrix} \frac{\partial \mathcal{T}_x^n}{\partial \hat{x}} & \frac{\partial \mathcal{T}_x^n}{\partial \hat{y}} \\ \frac{\partial \mathcal{T}_y^n}{\partial \hat{x}} & \frac{\partial \mathcal{T}_y^n}{\partial \hat{y}} \end{bmatrix}.$$

The transformation 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.

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. The algorithm starts with the automatic construction of a linear mesh with several specified properties. The boundary edges of those linear elements are then curved using cubic Bézier polynomials such that these boundary edges constitute a smooth closed curve. The procedure next curves the interior elements by iteratively solving for the equilibrium configuration of an elasticity problem until all the invalid elements are eliminated and the mesh quality is dramatically improved.

3.1. Linear mesh construction

The linear mesh has to provide an approximation of the object shape, and we measure the closeness by 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 quadtree based image-to-mesh conversion algorithm [3], which satisfies the following requirements:

1. The mesh maintains the topology of the original image.
2. The two-sided Hausdorff distance from the mesh to the image and the image to the mesh is within a user-specified tolerance.
3. The vertices that are on the mesh boundary are all located on the image boundary.
4. 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

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 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 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. For how to construct C^1 and C^2 Bézier curves, please see our previous work [10].

3.3. 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.

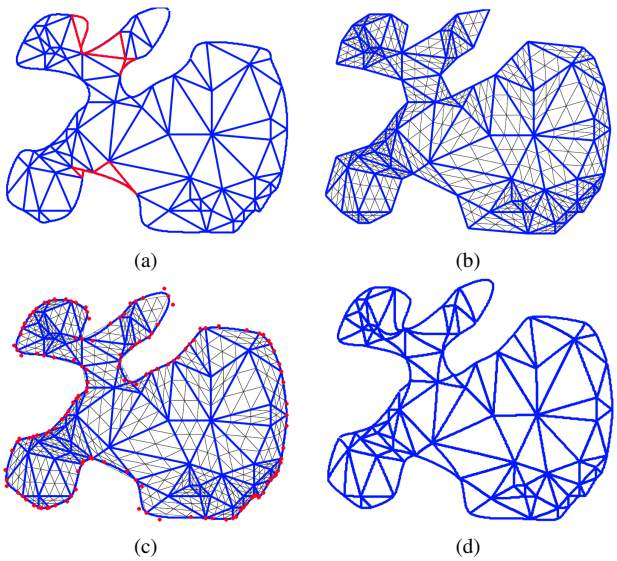


Figure 3. (a) Invalid mesh with red invalid elements. (b) The control nets of the linear mesh elements is the undeformed geometry. (c) The red control points of the smooth curved boundary edges are the external loads. (d) The final configuration is determined by solving for the equilibrium configuration of an elasticity problem.

We relocate the control points of the interior mesh edges using a finite element method [13]. 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 mesh 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. 3b). The external loads are the displacements of the control points (red points in Fig. 3c) 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. 3 illustrates these steps.

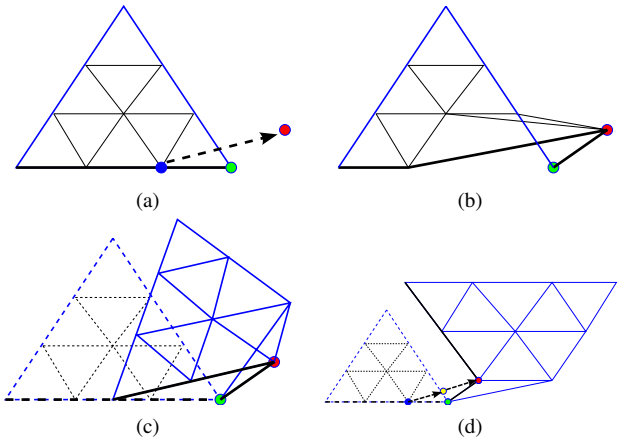


Figure 4. An illustration of the iterative finite element method. (a) The mesh composed of one element. (b) The invalid mesh with twisted control net. (c) The one-step FE method was applied, but the control net is still twisted. (d) The iterative FE method successfully corrected the twisted control net.

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. 4 illustrates the iterative FE method. In this example there is only one element in the mesh, the black border line represents the mesh 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 [10]. 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. 5 shows an example of the comparison of the result of one-step FE method and the result of the iterative FE method.

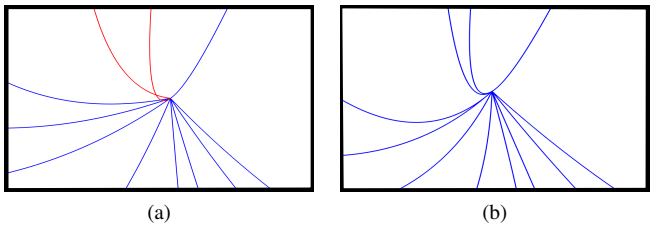


Figure 5. A comparison of the result of one-step FE method and the result of the iterative FE method. (a) After one-step FE method, the two red edges are still tangled together. (b) After eight iterations, the edges are untangled, all the elements are valid.

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 the mouse brain image [2], two slices of the human brain image [2], and the fly embryo image [1]. The mouse brain image (MB) has the size $198 * 169$ pixels; the first slice of the human brain image (HB I) has the size $239 * 233$ pixels; the second slice of the human brain image (HB II) has the size $235 * 283$ pixels; the fly embryo image (FE) 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. 6.

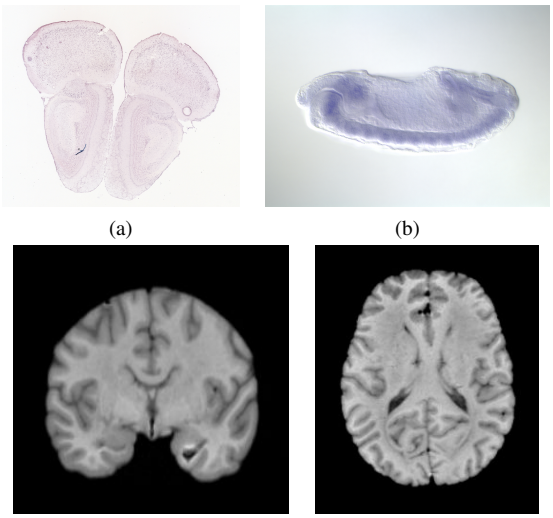


Figure 6. The original images. (a) The mouse brain image. (b) The fly embryo image. (c) The first slice of the human brain image. (d) The second slice of the human brain image.

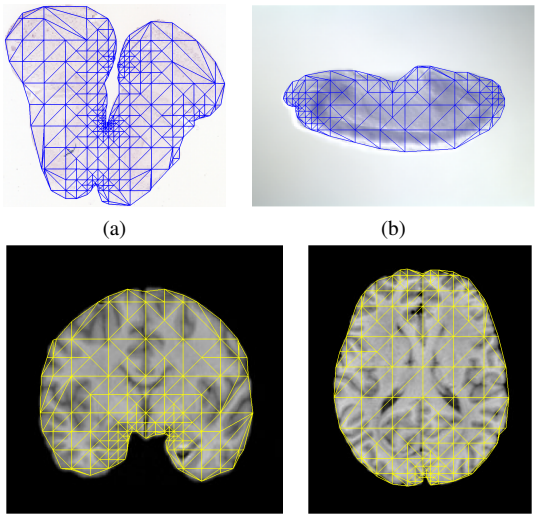


Figure 7. The linear meshes. (a) The linear mesh result for the mouse brain image. (b) The linear mesh result for the fly embryo image. (c) The linear mesh result for the first slice of the human brain image. (d) The linear mesh result for the second slice of the human brain image.

We show the linear mesh results for the original images with different requirements (in Fig. 7). For 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 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 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 mouse brain image is 3.6°; the minimum angle bound of the fly embryo image 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 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* [13]. 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.

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. 8, from left to right for each image, the boundaries are linear boundaries, C^1 boundaries and C^2 boundaries.

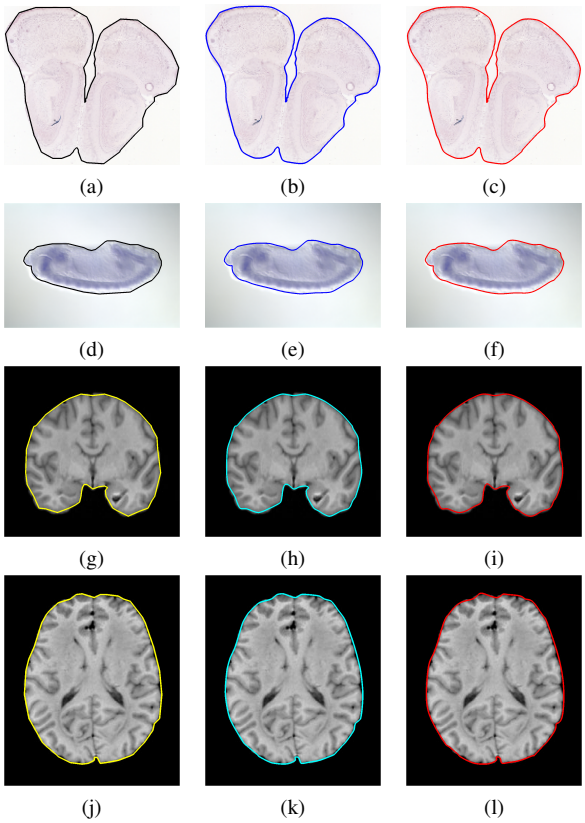


Figure 8. The linear mesh boundaries and curved mesh boundaries with C^1 and C^2 smoothness requirements for the original images.

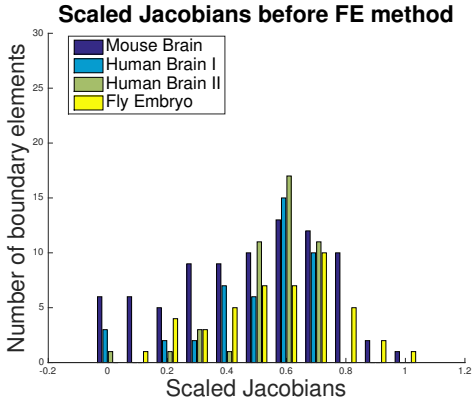
Table 1. Accuracy of the mesh boundaries

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

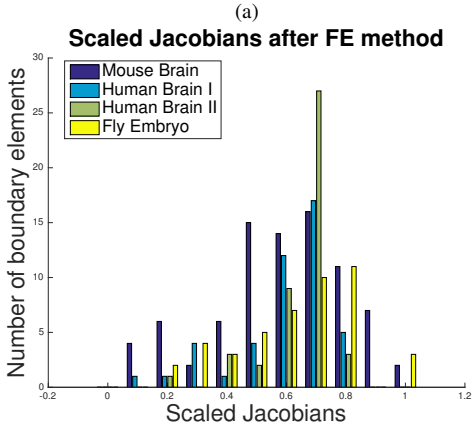
The accuracy was specified by the number of misclassified pixels that composed of background pixels that are inside the mesh and tissue pixels that are outside the mesh. The accuracy of the linear mesh results and the corresponding curvilinear meshes with C^1 and C^2 smoothness requirements are listed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

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 1, 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 mouse brain image is 6, 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. 10a, Fig. 12a and Fig. 12c. The iterative FE method was applied to the invalid meshes. After 6, 5, 5 iterations, all the invalid elements were eliminated for these invalid meshes. For the fly embryo image, there is no invalid element (Fig. 11a). We executed 10 iterations to improve the quality of the elements. The final meshes are shown in Fig. 10b, Fig. 12b, Fig. 12d, and Fig. 11b.



(a)



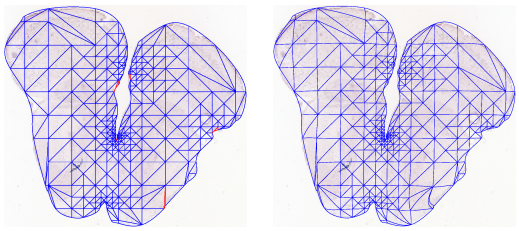
(b)

Figure 9. 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.

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

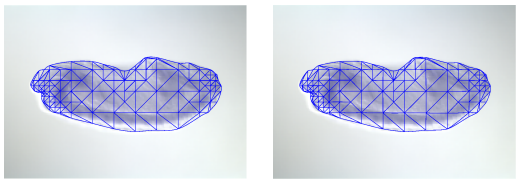
$$I = \frac{min|J|}{max|J|},$$

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. 9, the iterative FE method produced more elements with larger *scaled Jacobian*, thus the bad shaped elements were improved largely.



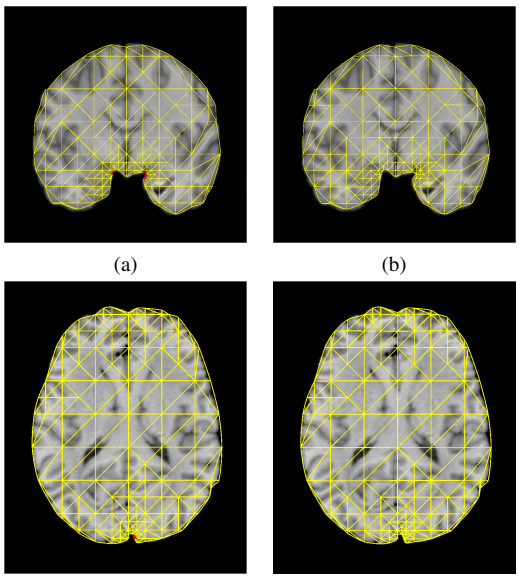
(a) (b)

Figure 10. Invalid meshes and corresponding corrected meshes for the mouse brain image. (a) Invalid curvilinear mesh for the mouse brain image. (b) Valid final curvilinear mesh with quality improvement for the mouse brain image.



(a) (b)

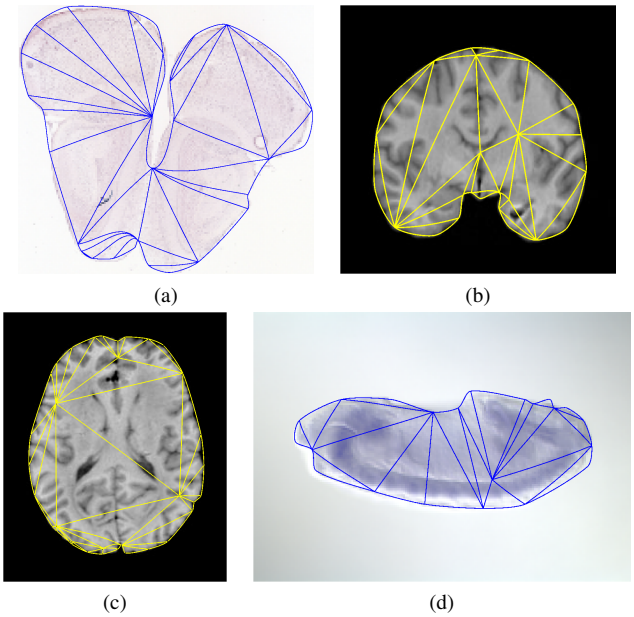
Figure 11. Bad quality curvilinear mesh for the fly embryo image and corresponding mesh with quality improvement. (a) Bad quality curvilinear mesh for the fly embryo image. (b) Improved quality curvilinear mesh for the fly embryo image.



(a) (b)

(c) (d)

Figure 12. 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.



(a) (b)

(c) (d)

Figure 13. curvilinear meshes with coarsened elements for the original images.

The algorithm can also construct curved meshes with coarsened elements inside that have fewer elements. Fig. 13 shows the coarsened curvilinear meshes for the original images.

Table 2. Run time (s) for the eight examples

Example	TNE	NIE	ITRS	RTL (s)	TFE (s)	TRT (s)
MB (fine)	528	6	6	0.169	88.457	89.703
HB I (fine)	251	3	5	0.239	23.890	24.785
HB II (fine)	235	1	5	0.262	23.212	25.691
FE (fine)	213	0	10	0.111	42.373	44.357
MB (coarse)	39	4	10	0.164	10.199	13.122
HB I (coarse)	27	3	12	0.238	16.860	19.079
HB II (coarse)	39	1	80	0.266	40.500	42.391
FE (coarse)	21	3	8	0.119	10.685	14.366

In Table 2, 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 multi-tissue triangular curvilinear mesh construction, the development of the corresponding parallel algorithm and the extension to the three-dimensional high-order mesh generation.

6. ACKNOWLEDGMENTS

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

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] 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.

[5] 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.

[6] 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.

[7] 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.

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

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

[10] Jing Xu and Andrey Chernikov. Automatic curvilinear quality mesh generation with smooth mesh boundaries of medical images. In *Modeling, Simulation, and Visualization Student Capstone Conference*, pages 205–212, Suffolk, VA, April 2015. Virginia Modeling, Analysis and Simulation Center.

[11] 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.

[12] 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.

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